Nonlinear Convergence Analysis for the Parareal Algorithm

Martin J. Gander and Ernst Hairer[1]

Section de Mathématiques, Université de Genève, CP 64, 1211 Genève, Switzerland, martin.gander@math.unige.ch, ernst.hairer@math.unige.ch

1 Introduction

Time domain decomposition methods have a long history: already Nievergelt [1964] made the following visionary statement:

> "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."

Nievergelt proposed a parallel algorithm based on a decomposition of the time direction for the solution of ordinary differential equations. While his idea targeted large scale parallelism, Miranker and Liniger [1967] proposed a little later a family of naturally parallel Runge Kutta methods for small scale parallelism:

> "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."

Waveform relaxation methods, introduced by Lelarasmee et al. [1982] for the large scale simulation of VLSI design, are another fundamental way to introduce time parallelism into the solution of evolution problems. For an up to date historical review and further references, see Gander and Vandewalle [2006].

The present research was motivated by the introduction of the parareal algorithm by Lions et al. [2001]. We show in this paper a general superlinear convergence result for the parareal algorithm applied to a nonlinear system of ordinary differential equations.

2 Derivation of the Parareal Algorithm

The parareal algorithm is a time parallel algorithm for the solution of the general nonlinear system of ordinary differential equations

$$\boldsymbol{u}'(t) = \boldsymbol{f}(\boldsymbol{u}(t)), \quad t \in (0,T), \quad \boldsymbol{u}(0) = \boldsymbol{u}^0, \tag{1}$$

where $\boldsymbol{f} : \mathbb{R}^M \longrightarrow \mathbb{R}^M$ and $\boldsymbol{u} : \mathbb{R} \longrightarrow \mathbb{R}^M$.

To obtain a time parallel algorithm for (1), we decompose the time domain $\Omega = (0,T)$ into N time subdomains $\Omega_n = (T_n, T_{n+1})$, $n = 0, 1, \ldots N-1$, with $0 = T_0 < T_1 < \ldots < T_{N-1} < T_N = T$, and $\Delta T_n := T_{n+1} - T_n$, and consider on each time subdomain the evolution problem

$$\boldsymbol{u}_n'(t) = \boldsymbol{f}(\boldsymbol{u}_n(t)),\ t \in (T_n, T_{n+1}),\ \boldsymbol{u}_n(T_n) = \boldsymbol{U}_n,\ n = 0, 1, \ldots, N-1, \tag{2}$$

where the initial values $\boldsymbol{U}_n$ need to be determined such that the solutions on the time subdomains Ω_n coincide with the restriction of the solution of (1) to Ω_n, i.e. the $\boldsymbol{U}_n$ need to satisfy the system of equations

$$\boldsymbol{U}_0 = \boldsymbol{u}^0, \quad \boldsymbol{U}_n = \boldsymbol{\varphi}_{\Delta T_{n-1}}(\boldsymbol{U}_{n-1}), \quad n = 1, \ldots, N-1, \tag{3}$$

where $\boldsymbol{\varphi}_{\Delta T_n}(\boldsymbol{U})$ denotes the solution of (1) with initial condition $\boldsymbol{U}$ after time ΔT_n. This time decomposition method is nothing else than a multiple shooting method for (1), see Chartier and Philippe [1993]. Letting $\boldsymbol{U} = (\boldsymbol{U}_0^T, \ldots \boldsymbol{U}_{N-1}^T)^T$, the system (3) can be written in the form

$$\boldsymbol{F}(\boldsymbol{U}) = \begin{pmatrix} \boldsymbol{U}_0 - \boldsymbol{u}^0 \\ \boldsymbol{U}_1 - \boldsymbol{\varphi}_{\Delta T_0}(\boldsymbol{U}_0) \\ \vdots \\ \boldsymbol{U}_{N-1} - \boldsymbol{\varphi}_{\Delta T_{N-2}}(\boldsymbol{U}_{N-2}) \end{pmatrix} = \boldsymbol{0}, \tag{4}$$

where $\boldsymbol{F} : \mathbb{R}^{M \cdot N} \longrightarrow \mathbb{R}^{M \cdot N}$. System (4) defines the unknown initial values $\boldsymbol{U}_n$ for each time subdomain, and needs to be solved, in general, by an iterative method. For a direct method in the case where (1) is linear and the system (4) can be formed explicitly, see Amodio and Brugnano [1997].

Applying Newtons method to (4) leads after a short calculation to

$$\begin{aligned} \boldsymbol{U}_0^{k+1} &= \boldsymbol{u}^0, \\ \boldsymbol{U}_n^{k+1} &= \boldsymbol{\varphi}_{\Delta T_{n-1}}(\boldsymbol{U}_{n-1}^k) + \frac{\partial \boldsymbol{\varphi}_{\Delta T_{n-1}}}{\partial \boldsymbol{U}}(\boldsymbol{U}_{n-1}^k)(\boldsymbol{U}_{n-1}^{k+1} - \boldsymbol{U}_{n-1}^k), \end{aligned} \tag{5}$$

where $n = 1, \ldots, N-1$. Chartier and Philippe [1993] showed that the method (5) converges quadratically, once the approximations are close enough to the solution. However in general, it is too expensive to compute the Jacobian terms in (5) exactly. An interesting recent approximation is the parareal algorithm, which uses two approximations with different accuracy: let $\boldsymbol{F}(T_n, T_{n-1}, \boldsymbol{U}_{n-1})$

be an accurate approximation to the solution $\boldsymbol{\varphi}_{\Delta T_{n-1}}(\boldsymbol{U}_{n-1})$ on time subdomain Ω_{n-1}, and let $\boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{U}_{n-1})$ be a less accurate approximation, for example on a coarser grid, or a lower order method, or even an approximation using a simpler model than (1). Then, approximating the time subdomain solves in (5) by $\boldsymbol{\varphi}_{\Delta T_{n-1}}(\boldsymbol{U}^k_{n-1}) \approx \boldsymbol{F}(T_n, T_{n-1}, \boldsymbol{U}^k_{n-1})$, and the Jacobian term by

$$\frac{\partial \boldsymbol{\varphi}_{\Delta T_{n-1}}}{\partial \boldsymbol{U}_{n-1}}(\boldsymbol{U}^k_{n-1})(\boldsymbol{U}^{k+1}_{n-1} - \boldsymbol{U}^k_{n-1}) \approx \boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{U}^{k+1}_{n-1}) - \boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{U}^k_{n-1}),$$

we obtain as approximation to (5)

$$\begin{aligned} \boldsymbol{U}^{k+1}_0 &= \boldsymbol{u}^0, \\ \boldsymbol{U}^{k+1}_n &= \boldsymbol{F}(T_n, T_{n-1}, \boldsymbol{U}^k_{n-1}) + \boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{U}^{k+1}_{n-1}) - \boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{U}^k_{n-1}), \end{aligned} \tag{6}$$

which is the parareal algorithm, see Lions et al. [2001] for a linear model problem, and Baffico et al. [2002] for the formulation (6). A natural initial guess is the coarse solution, i.e. $\boldsymbol{U}^0_n = G(T_n, T_{n-1}, \boldsymbol{U}^0_{n-1})$.

3 Convergence Analysis

To simplify the exposition, we assume in this section that all the time subdomains are of the same size, $\Delta T_n = \Delta T := \frac{T}{N}$, $n = 0, 1, \ldots, N-1$, and that $\boldsymbol{F}$ is the exact solution, i.e. $\boldsymbol{F}(T_n, T_{n-1}, \boldsymbol{U}^k_{n-1}) = \boldsymbol{\varphi}_{\Delta T_{n-1}}(\boldsymbol{U}^k_{n-1})$. We also assume that the difference between the approximate solution given by $\boldsymbol{G}$ and the exact solution can be expanded for ΔT small,

$$\boldsymbol{F}(T_n, T_{n-1}, x) - \boldsymbol{G}(T_n, T_{n-1}, x) = c_{p+1}(x)\Delta T^{p+1} + c_{p+2}(x)\Delta T^{p+2} + \ldots, \tag{7}$$

which is possible if the right hand side function $\boldsymbol{f}$ in (1) is smooth enough, and $\boldsymbol{G}$ is a Runge Kutta method for example. We finally assume that G satisfies the Lipschitz condition

$$\|\boldsymbol{G}(t + \Delta T, t, \boldsymbol{x}) - \boldsymbol{G}(t + \Delta T, t, \boldsymbol{y})\| \leq (1 + C_2 \Delta T)\|\boldsymbol{x} - \boldsymbol{y}\|. \tag{8}$$

Theorem 1. *Let $\boldsymbol{F}(T_n, T_{n-1}, \boldsymbol{U}^k_{n-1}) = \boldsymbol{\varphi}_{\Delta T_{n-1}}(\boldsymbol{U}^k_{n-1})$ be the exact solution on time subdomain Ω_{n-1}, and let $\boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{U}^k_{n-1})$ be an approximate solution with local truncation error bounded by $C_3 \Delta T^{p+1}$, and satisfying (7), where the c_j, $j = p+1, p+2, \ldots$ are continuously differentiable, and assume that G satisfies the Lipschitz condition (8). Then, at iteration k of the parareal algorithm (6), we have the bound*

$$\begin{aligned} \|\boldsymbol{u}(T_n) - \boldsymbol{U}^k_n\| &\leq \frac{C_3}{C_1} \frac{(C_1 \Delta T^{p+1})^{k+1}}{(k+1)!} (1 + C_2 \Delta T)^{n-k-1} \prod_{j=0}^{k} (n-j) \\ &\leq \frac{C_3}{C_1} \frac{(C_1 T_n)^{k+1}}{(k+1)!} e^{C_2(T_n - T_{k+1})} \Delta T^{p(k+1)}. \end{aligned}$$

Proof. From the definition of the parareal algorithm (6), we obtain, using that $\boldsymbol{F}$ is the exact solution and adding and subtracting $\boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{u}(T_{n-1}))$

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

Now using expansion (7) for the first two terms on the right hand side, and (8) on the last one, we obtain on taking norms

$$\|\boldsymbol{u}(T_n) - \boldsymbol{U}_n^{k+1}\| \le C_1 \Delta T^{p+1} \|\boldsymbol{u}(T_{n-1}) - \boldsymbol{U}_{n-1}^k\| + (1 + C_2 \Delta T) \|\boldsymbol{u}(T_{n-1}) - \boldsymbol{U}_{n-1}^{k+1}\|.$$

This motivates to study the recurrence relation

$$e_n^{k+1} = \alpha e_{n-1}^k + \beta e_{n-1}^{k+1}, \quad e_n^0 = \gamma + \beta e_{n-1}^0, \tag{9}$$

where $\alpha = C_1 \Delta T^{p+1}$, $\beta = 1 + C_2 \Delta T$ and $\gamma = C_3 \Delta T^{p+1}$, since e_n^k is then an upper bound on $\|\boldsymbol{u}(T_n) - \boldsymbol{U}_n^k\|$. Multiplying (9) by ζ^n and summing over n, we find that the generating function $\rho_k(\zeta) := \sum_{n \ge 1} e_n^k \zeta^n$ satisfies the recurrence relation

$$\rho_{k+1}(\zeta) = \alpha\zeta\rho_k(\zeta) + \beta\zeta\rho_{k+1}(\zeta), \quad \rho_0(\zeta) = \gamma \frac{\zeta}{1-\zeta} + \beta\zeta\rho_0(\zeta).$$

Solving for $\rho_k(\zeta)$, we obtain after induction

$$\rho_k(\zeta) = \gamma\alpha^k \frac{\zeta^{k+1}}{(1-\zeta)} \frac{1}{(1-\beta\zeta)^{k+1}}.$$

Replacing the factor $1 - \zeta$ in the denominator by $1 - \beta\zeta$ only increases the coefficients in the power series of $\rho_k(\zeta)$. Using now the binomial series expansion

$$\frac{1}{(1-\beta\zeta)^{k+2}} = \sum_{j \ge 0} \binom{k+1+j}{j} \beta^j \zeta^j,$$

we obtain for the n-th coefficient e_n^k the bound

$$e_n^k \le \gamma\alpha^k \beta^{n-k-1} \binom{n}{k+1},$$

which concludes the proof.

4 Numerical Experiments

We show now several numerical experiments, first for small systems of ordinary differential equations, where the only potential for parallelization lies in the time direction, and then also for a partial differential equation, namely the viscous Burgers equation.

4.1 Brusselator

The brusselator system of ordinary differential equations models a chain of chemical reactions and is given by

$$\dot{x} = A + x^2 y - (B+1)x, \qquad \dot{y} = Bx - x^2 y.$$

We chose for the parameters $A = 1$ and $B = 3$, and since $B > A^2 + 1$, the system will form a limit cycle, see Hairer et al. [1993]. We start the simulation with the initial conditions $x(0) = 0$, $y(0) = 1$, and compute an approximate solution over the time interval $t \in [0, T = 12]$ using the classical fourth order Runge Kutta method with coarse time step $\Delta T = \frac{T}{32}$, and fine time step $\Delta t = \frac{T}{640}$, which gives a solution with an accuracy of $5.62e-6$. In Figure 1, we show the initial guess from the coarse solver, and the first five iterates of the parareal algorithm in the phase plane, and also the difference between the parareal approximation and the complete fine approximation as a function of time. The red dot in the phase plane, and the vertical red line in the error

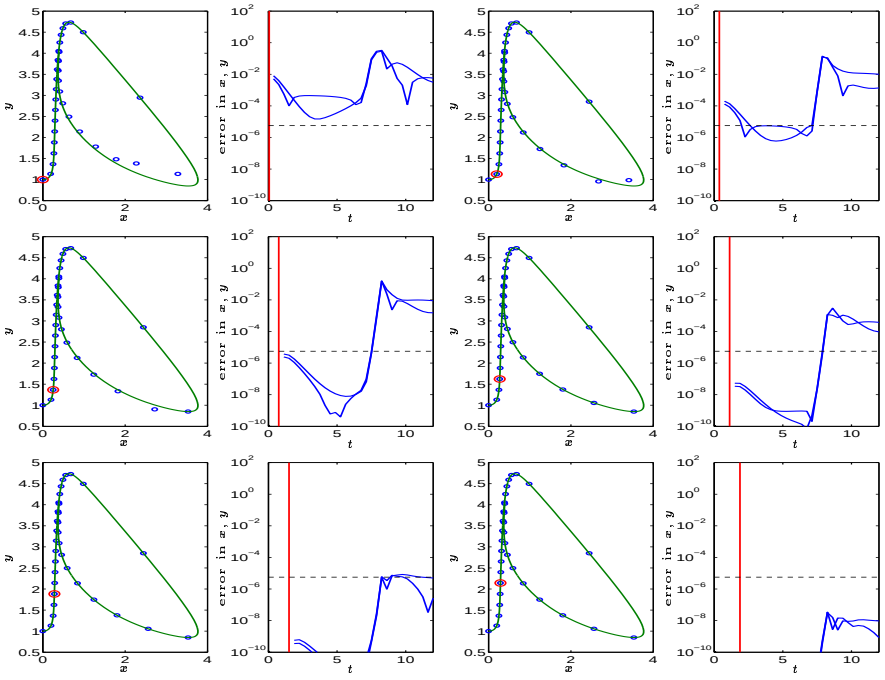

Fig. 1. Parareal approximation of the solution of the Brusselator problem.

plots indicate how far one could have computed the fine solution sequentially in the same computation time, neglecting the cost of the coarse solve. The fine dashed line indicates the accuracy of the fine grid solution. Clearly there

is a parallel speedup with this type of time parallelization: with 32 processors, one could have computed the numerical approximation to the same accuracy of $5.62e-6$ about eight times faster than with one processor.

4.2 Arenstorf orbit

Arenstorf orbits are closed orbits of a light object (e.g. a satellite) moving under the influence of gravity of two heavy objects (e.g. planets, moons). The equations of motion for the example of two heavy objects are

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

where D_j, $j = 1, 2$ are function of x and y,

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

If the parameters are $a = 0.012277471$ and $b = 1-a$, and the initial conditions are chosen to be $x(0) = 0.994$, $\dot{x} = 0$, $y(0) = 0$, $\dot{y}(0) = -2.00158510637908$, then the solution is a nice closed orbit with period $T = 17.06521656015796$, see Hairer et al. [1993]. There have been earlier attempts to compute planetary orbits in parallel, see Saha et al. [1997], where a multiple shooting method was developed. We use again the parareal algorithm to compute the Arenstorf orbit in parallel, with the classical fourth order Runge Kutta method and coarse time step $\Delta T = \frac{T}{250}$, and fine time step $\Delta t = \frac{T}{80000}$, such that the fine trajectory has an accuracy of $9.98e-6$. We show in Figure 2 the initial guess and the first five iterations of the parareal algorithm, as in the case of the brusselator problem. While the initial guess is completely off, and simply spirals outward, the first iteration already reveals the shape of the Arenstorf orbit, and the algorithm has converged to the precision of the fine time step approximation after four iterations. Neglecting the cost of the coarse grid solve, one could have computed this trajectory with 250 processors about 62 times faster in parallel, than with one processor sequentially. The fact that the initial guess is so off is due to the tremendous sensitivity of the solution to the initial conditions, so it would be better to use an adaptive method here. We are currently studying the use of adaptivity in the context of the parareal algorithm.

4.3 Lorenz Equations

Weather prediction could be an important application of the parareal algorithm, since predictions have to be made in real time. If a large scale parallel computer is available, and the parallelization in space of the partial differential equation modeling the evolution of the weather is already saturated, the only way to speed up the computation is to try to parallelize the time

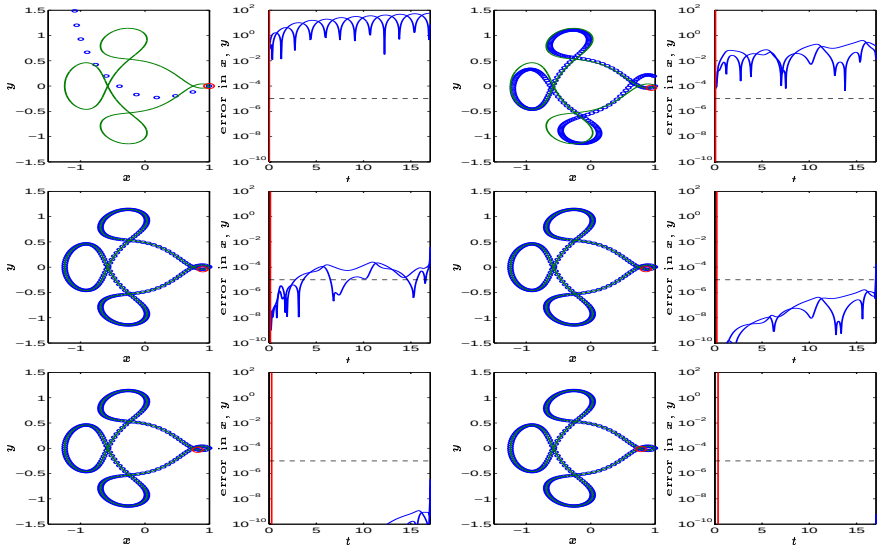

Fig. 2. Parareal approximation of the Arenstorf orbit.

direction. A very simple model for weather prediction is the model given by the Lorenz equations,

$$\dot{x} = -\sigma x + \sigma y, \quad \dot{y} = -xz + rx - y, \quad \dot{z} = xy - bz.$$

These equations were first studied by Lorenz [1978], who discovered that in certain cases approximations to their solution are very sensitive to small changes in the initial data (he noticed this when he interrupted a computation and wrote the current position of the solution down by hand to continue the next day, but his notes included only the first four digits, and not the full precision). A legend then says that looking at the solution of his equations, which looks on the attractor like a butterfly, Lorenz concluded that the wings of a butterfly in Europe could create a thunderstorm in the US.

We chose for the parameters in the Lorenz equations $\sigma = 10$, $r = 28$ and $b = \frac{8}{3}$, such that the system is in the chaotic regime, and trajectories converge to the butterfly attractor. We start with the initial conditions $(x, y, z)(0) = (20, 5, -5)$, and compute with the parareal algorithm an approximate solution on the time interval $[0, T = 10]$, using again the classical fourth order Runge Kutta method with coarse time step $\Delta T = \frac{T}{180}$, and fine time step $\Delta t = \frac{T}{14400}$, which leads to an accuracy in the fine trajectory of $2.4e-6$. We show in Figure 3 the initial guess and the first five iterations of the parareal algorithm, together with error curves for the coordinates, as a function of time. One can see that for the first two iterations, the approximate parareal trajectory is not in the same wing of the butterfly attractor as the converged trajectory. At iteration three, the situation changes and the parareal approximation follows now the converged trajectory. From this iteration on, the algorithm converges

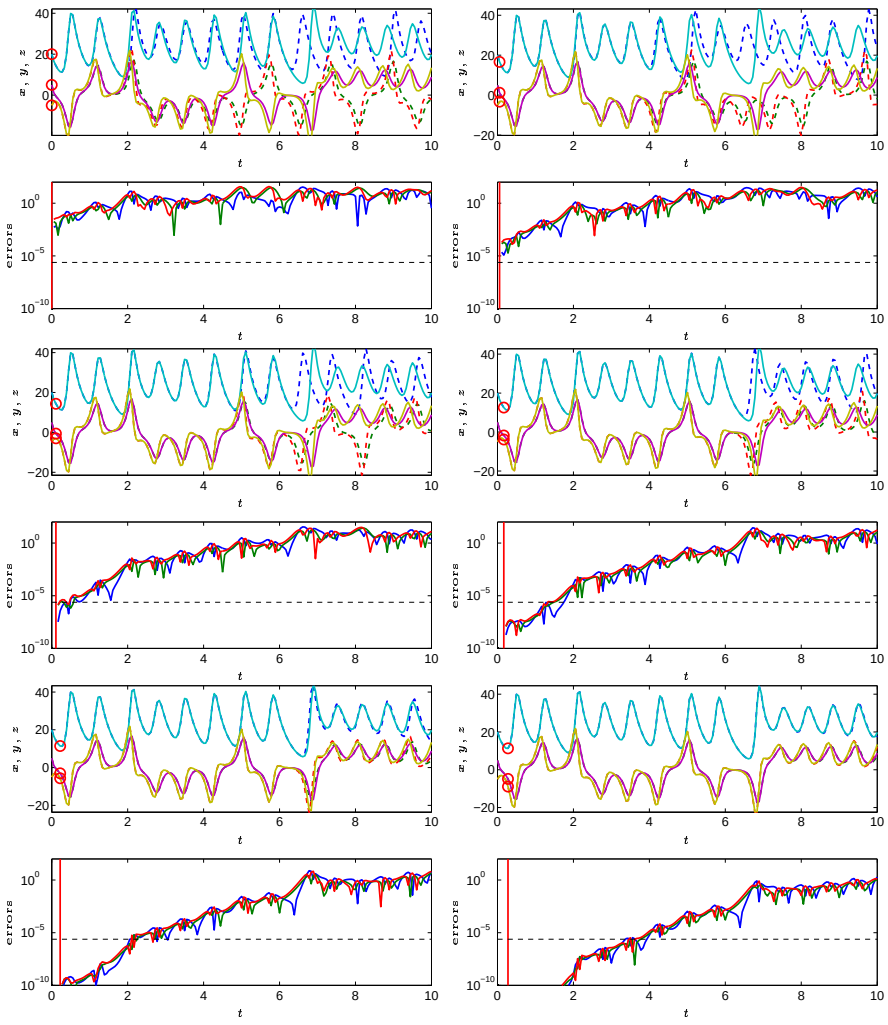

Fig. 3. Initial guess and first five parareal approximations of the solution of the Lorenz equations.

on the entire time interval, as one can see in Figure 4, where we show iteration six to eleven. At iteration ten, an overall accuracy of $1e-6$, which corresponds to the fine grid solution accuracy, is achieved. Neglecting the cost of the coarse solver, one could therefore have computed a fine grid accurate solution with 180 processors about 18 times faster than sequentially, as indicated by the colored dots and the red vertical line on the graphs.

In Figure 5 on the left, we show how the difference of the parareal approximation and the converged solution, measured in the L^2-norm in space,

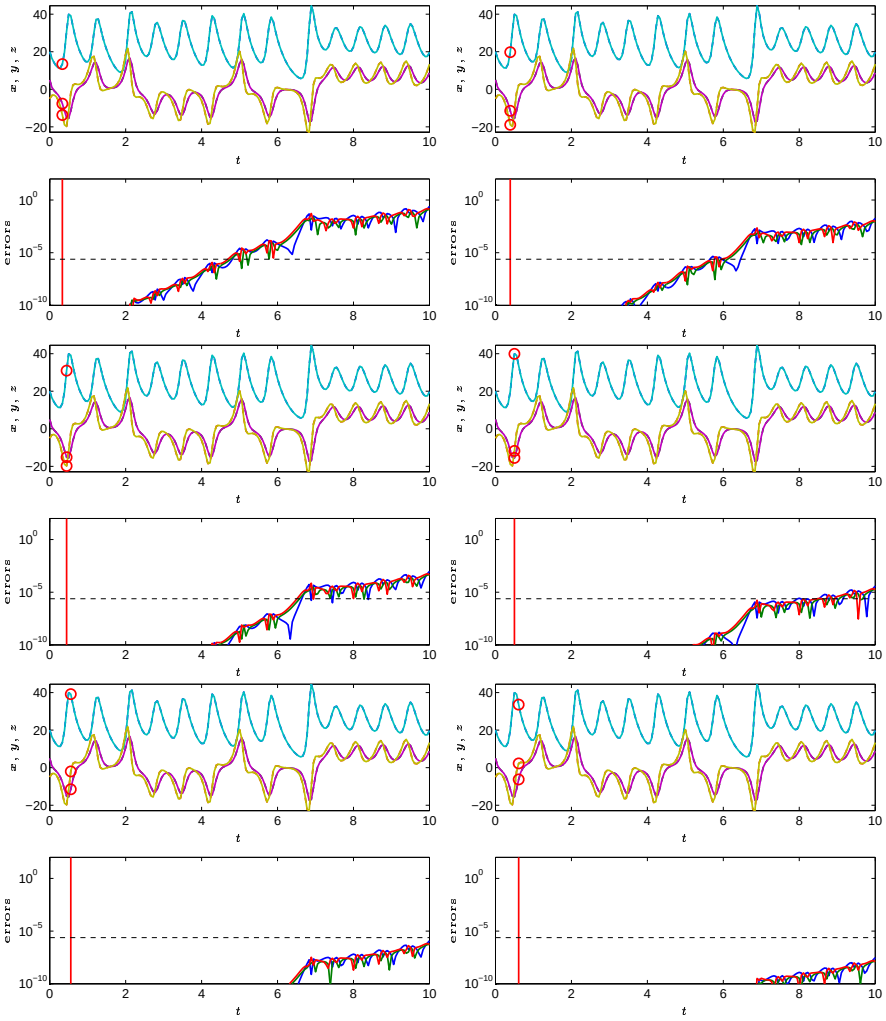

Fig. 4. Sixth to eleventh parareal approximation of the solution of the Lorenz equations.

and in the L^∞-norm in time, diminishes as a function of the iterations of the parareal algorithm. One can clearly see that the convergence is superlinear.

In the context of the Lorenz equations, it is interesting to investigate the behavior of the parareal algorithm with respect to the chaotic nature of the system. In Figure 5, we show on the right the convergence behavior of the parareal algorithm for an implementation with variable precision arithmetic, using 16, 32 and 48 digits of accuracy. One can see that the theoretical result of superlinear convergence stops at a certain level before the numerical precision has been reached, and the algorithm stagnates, or in other words,

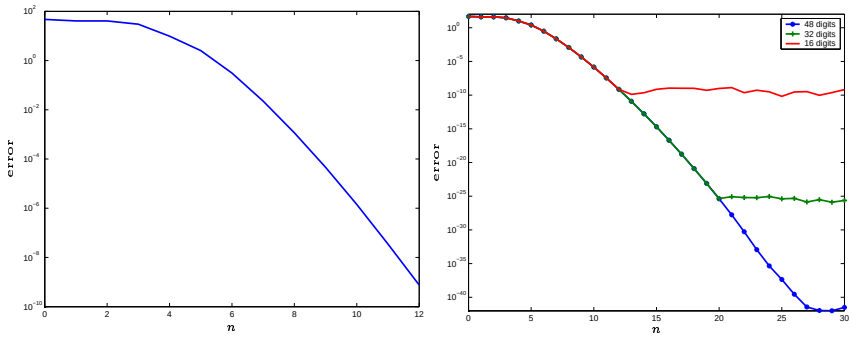

Fig. 5. Convergence behavior of the parareal algorithm applied to the Lorenz equations.

the trajectory has converged to a different solution from the one computed sequentially, due to roundoff errors.

4.4 Viscous Burgers equation

We finally show numerical experiments for a non-linear partial differential equation, the viscous Burgers equation,

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

with homogeneous boundary data, such that the solution forms Friedrich's N-wave. We chose for the viscosity parameter $\nu = \frac{1}{50}$, used a centered finite difference discretization with spatial step $\Delta x = \frac{1}{50}$, and a backward Euler discretization in time. We only parallelized the solution in time using the parareal algorithm, with coarse time step $\Delta T = \frac{1}{10}$, and fine time step $\Delta t = \frac{1}{100}$, which gives a numerical accuracy of $4e-2$. We show in Figure 6 on the left the converged solution over a short time interval, $[0, T = 0.1]$, where one can see how the N-wave is forming, and on the right the same solution over a longer time interval, $[0, T = 1]$. In Figure 7, we show on the left the convergence behavior of the parareal algorithm applied to the Burgers equation, when the problem is posed over time intervals of various length. Again we measure the error in the L^2-norm in space, and the L^∞-norm in time. Over short time intervals, the convergence of the parareal algorithm is faster than over long time intervals. In the case of $T = 0.1$, the algorithm converges at step two to the accuracy of the discretization error, and one could therefore, neglecting the coarse solve, compute this approximation with ten processors five times faster in parallel than with one processor. Note also that as one continues to iterate, the algorithm converges further toward the fine grid solution, until the roundoff error accuracy is reached at step 10, as indicated by Theorem 1. Over longer time intervals, for example $T = 1$, with the same parareal configuration, the accuracy of the discretization error is reached at iteration

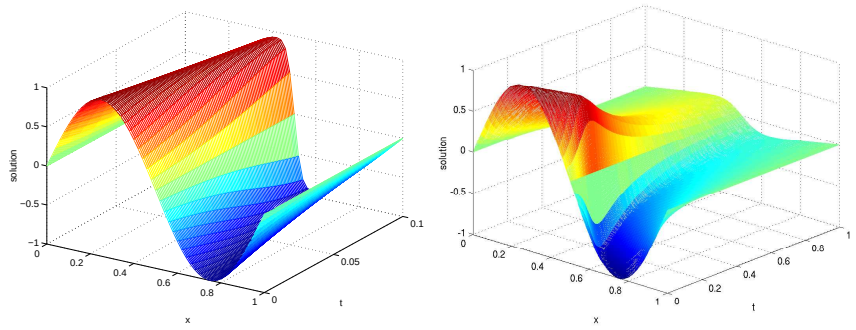

Fig. 6. Converged approximate solution for the Burgers equation over a short and long time interval.

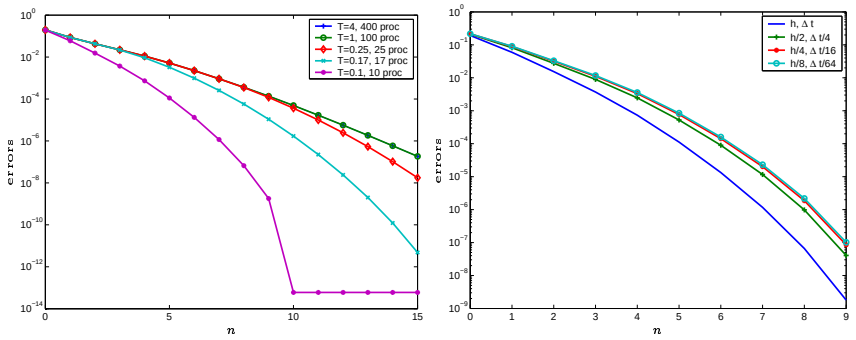

Fig. 7. Convergence behavior of the parareal algorithm applied to Burgers equation, on the left for various lengths of the time interval, and on the right when the accuracy of the discretization is increased.

four. Computing with one hundred processors in parallel, this solution could have been obtained 25 times faster than sequentially on one processor.

In Figure 7 on the right, we show how the discretization error affects the parareal algorithm. For $T = 0.1$, we computed more and more refined solutions, both in space and time, with truncation error $4e-2$, $1e-2$, $2.5e-3$ and $6.2e-4$, using the parareal algorithm with 10 coarse time intervals. The convergence plot shows that the convergence rate becomes independent of the mesh parameters, as it was proved for the linear case in Gander and Vandewalle [2006].

5 Conclusions

We showed that the parareal algorithm applied to a nonlinear system of ordinary differential equations converges superlinearly on any bounded time interval. We illustrated this result with four non-linear examples coming from

chemical reactions, planetary orbits, weather forecast and fluid flow problems. These examples show that parallel speedup in time is possible, although not at the same level as in space, where one often asks for perfect speedup, i.e. the computation with one hundred processors should be one hundred times faster. For time parallelization with the parareal algorithm, one has to be satisfied with less, but if this is the only option left to speedup the solution time, it might be worthwhile considering it.

References

Pierluigi Amodio and Luigi Brugnano. Parallel implementation of block boundary value methods for ODEs. *J. Comp. Appl. Math.*, 78:197–211, 1997.

Leonardo Baffico, Stephane Bernard, Yvon Maday, Gabriel Turinici, and Gilles Zérah. Parallel-in-time molecular-dynamics simulations. *Physical Review E*, 66:057706–1–4, 2002.

Philippe Chartier and Bernard Philippe. A parallel shooting technique for solving dissipative ODEs. *Computing*, 51:209–236, 1993.

Martin J. Gander and Stefan Vandewalle. Analysis of the parareal time-parallel time-integration method. *SIAM J. Sci. Comput.*, 2006. in print.

Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems.* Springer-Verlag, Second Revised Edition, 1993.

Ekachai Lelarasmee, Albert E. Ruehli, and Alberto L. Sangiovanni-Vincentelli. The waveform relaxation method for time-domain analysis of large scale integrated circuits. *IEEE Trans. on CAD of IC and Syst.*, 1:131–145, 1982.

Jacques-Louis Lions, Yvon Maday, and Gabriel Turinici. A parareal in time discretization of pde's. *C.R. Acad. Sci. Paris, Serie I*, 332:661–668, 2001. URL `http://www.elsevier.nl/gej-ng/10/37/18/47/27/35/article.pdf`.

Edward N. Lorenz. On the prevalence of aperiodicity in simple systems. In M. Grmela and J.E. Marsden, editors, *Global Analysis*, volume 755, pages 53–75, Calgary, 1978. Lecture Notes in Mathematics.

Willard L. Miranker and Werner Liniger. Parallel methods for the numerical integration of ordinary differential equations. *Math. Comp.*, 91:303–320, 1967.

Jörg Nievergelt. Parallel methods for integration ordinary differential equations. *Comm. ACM*, 7:731–733, 1964.

Prasenjit Saha, Joachim Stadel, and Scott Tremaine. A parallel integration method for solar system dynamics. *Astron. J.*, 114:409–415, 1997.