SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 556–578

ANALYSIS OF THE PARAREAL TIME-PARALLEL TIME-INTEGRATION METHOD*

MARTIN J. GANDER† AND STEFAN VANDEWALLE‡

Abstract. The parareal algorithm is a method to solve time-dependent problems parallel in time: it approximates parts of the solution later in time simultaneously to parts of the solution earlier in time. In this paper the relation of the parareal algorithm to space-time multigrid and multiple shooting methods is first briefly discussed. The focus of the paper is on new convergence results that show superlinear convergence of the algorithm when used on bounded time intervals, and linear convergence for unbounded intervals.

Key words. time-parallel time-integration, parareal, convergence analysis, shooting, multigrid, deferred correction

AMS subject classifications. 65R20, 45L05, 65L20

DOI. 10.1137/05064607X

1. Introduction.

The parareal algorithm was presented by Lions, Maday, and Turinici in [19] as a numerical method to solve evolution problems in parallel. The name was chosen to indicate that the algorithm is well suited for parallel real time computations of evolution problems whose solution cannot be obtained in real time using one processor only. The method approximates successfully the solution later in time before having fully accurate approximations from earlier times. The algorithm has received a lot of attention over the past few years, especially in the domain decomposition literature [18]. Extensive experiments can be found for fluid and structure problems in [8], for the Navier–Stokes equations in [9], and for reservoir simulation in [10]. Several variants of the method have been proposed, e.g., in [11, 8]. The algorithm has been further analyzed in [21, 22], and its stability is investigated in [29, 2].

Parareal is not the first algorithm to propose the solution of evolution problems in a time-parallel fashion. Already in 1964, Nievergelt suggested a parallel time-integration algorithm [24], which eventually developed into the multiple shooting method for boundary value problems; see [16]. The idea is to decompose the time-integration interval into subintervals, to solve an initial value problem on each subinterval concurrently, and to force continuity of the solution branches on successive intervals by means of a Newton procedure. This approach has also been proposed at a fine grain level, where the shooting intervals correspond to the step size of a one-step method [4]. Parallel multiple shooting for initial value problems was further studied in [6, 17].

In 1967, Miranker and Liniger [23] proposed a family of predictor-corrector methods, in which the prediction and correction steps can be performed in parallel over a number of time steps. Their idea was to "widen the computational front," i.e., to allow processors to compute solution values on several time steps concurrently. A

*Received by the editors November 25, 2005; accepted for publication (in revised form) October 2, 2006; published electronically March 22, 2007.
http://www.siam.org/journals/sisc/29-2/64607.html
†Section de Mathématiques, University of Geneva, 2-4 rue du Lièvre, CP 64 CH-1211 Geneva, Switzerland (Martin.Gander@math.unige.ch).
‡Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Leuven, Belgium (Stefan.Vandewalle@cs.kuleuven.be).

similar motivation led to the block time-integration methods by Shampine and Watts [27]. More recently, [26] and [33] considered the time-parallel application of iterative methods to the system of equations derived with implicit time-integration schemes applied to parabolic partial differential equations (PDEs). Instead of iterating until convergence over each time step before moving on to the next, they showed that it is possible to iterate over a number of time steps at once. Thus a different processor can be assigned to each time step and they all iterate simultaneously. This class of methods was analyzed for its parallel potential in [7]. The authors of that paper showed that the increase in parallel efficiency is often offset by an increased amount of computational work, because of slow information propagation forward in time. The acceleration of such methods by means of a multigrid technique led to the class of parabolic multigrid methods, as introduced by Hackbusch in [12]. The time-parallel multigrid method [13], the multigrid waveform relaxation method [20, 32], and the space-time multigrid method [14] also belong to that class. In [31], a time-parallel variant was shown to achieve excellent speedups on a computer with 512 processors; while run as a sequential algorithm the method is comparable to the best classical time marching schemes. Experiments with time-parallel parabolic multigrid methods on a system with 2^{14} processors are reported in [15].

Our paper is organized as follows. In section 2, we present a new and natural derivation of the parareal algorithm and some of its variants. It is shown that these techniques can be seen as practical implementations of the multiple shooting method. In section 3, we show that the algorithm also fits into the framework of time-multigrid methods. In particular the method is identified as a two-level full approximation scheme with an aggressive time-coarsening. In section 4, we develop new convergence results for scalar linear model problems, on bounded and unbounded time intervals. A number of practically relevant parareal methods are discussed in detail. In section 5, these convergence results are extended to two model PDEs: the heat equation and the pure advection equation. Numerical experiments that illustrate and clarify the theory are reported in section 6. We end in section 7 with some concluding remarks.

2. Parareal as a multiple shooting method. The parareal algorithm is a parallel method for computing the numerical solution for general systems of ordinary differential equations (ODEs) of the form

$$\boldsymbol{u}' = \boldsymbol{f}(\boldsymbol{u}), \quad \boldsymbol{u}(0) = \boldsymbol{u}_0, \quad t \in [0, T]. \tag{2.1}$$

The parareal algorithm is defined using two propagation operators. The operator $G(t_2, t_1, \boldsymbol{u}_1)$ provides a rough approximation to $\boldsymbol{u}(t_2)$ of the solution of (2.1) with initial condition $\boldsymbol{u}(t_1) = \boldsymbol{u}_1$, whereas operator $F(t_2, t_1, \boldsymbol{u}_1)$ provides a more accurate approximation of $\boldsymbol{u}(t_2)$. The algorithm starts with an initial approximation $\boldsymbol{U}_n^0$, $n = 0, 1, \ldots, N$, at time $t_0, t_1, \ldots, t_N$ given, for example, by the sequential computation of $\boldsymbol{U}_{n+1}^0 = G(t_{n+1}, t_n, \boldsymbol{U}_n^0)$, with $\boldsymbol{U}_0^0 = \boldsymbol{u}_0$, and then performs for $k = 0, 1, 2, \ldots$ the correction iteration

$$\boldsymbol{U}_{n+1}^{k+1} = G(t_{n+1}, t_n, \boldsymbol{U}_n^{k+1}) + F(t_{n+1}, t_n, \boldsymbol{U}_n^k) - G(t_{n+1}, t_n, \boldsymbol{U}_n^k). \tag{2.2}$$

Note that for $k \to \infty$ the parareal algorithm (2.2) will upon convergence generate a series of values $\boldsymbol{U}_n$ that satisfy $\boldsymbol{U}_{n+1} = F(t_{n+1}, t_n, \boldsymbol{U}_n)$. That is, the approximation at the time-points t_n will have achieved the accuracy of the F-propagator.

The algorithm can be interpreted most naturally as a classical deferred correction method [28]. In such a method, the solution to a "hard" problem $A(\boldsymbol{u}) = \boldsymbol{0}$ is

computed iteratively by solving a series of "easier" problems of the form $B(\boldsymbol{u}) = \boldsymbol{g}$. More precisely, starting with some approximation $\boldsymbol{u}^0$ to $\boldsymbol{u}$, one solves

$$B(\boldsymbol{u}^{k+1}) = B(\boldsymbol{u}^k) - A(\boldsymbol{u}^k) \quad \text{for } k = 0, 1, \ldots. \tag{2.3}$$

This iteration can be identified with (2.2) if $\boldsymbol{u} = (\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_N)^T$ and if $A(\boldsymbol{u})$ and $B(\boldsymbol{u})$ are vectors of length N times the size of system (2.1), whose components $A(\boldsymbol{u})_{n+1}$ and $B(\boldsymbol{u})_{n+1}$, for $n = 0, \ldots, N-1$, are vectors given, respectively, by

$$A(\boldsymbol{u})_{n+1} = \boldsymbol{U}_{n+1} - F(t_{n+1}, t_n, \boldsymbol{U}_n) \quad \text{and} \quad B(\boldsymbol{u})_{n+1} = \boldsymbol{U}_{n+1} - G(t_{n+1}, t_n, \boldsymbol{U}_n).$$

The parareal method first appeared in [19] for a linear scalar model problem, and it was applied to a linear PDE, and to a nonlinear one by linearization. A different formulation for the nonlinear case appeared first in [3] and then in the mathematically equivalent but simplified form (2.2) in [1]. The original linearized formulation of [19] was rederived in a more general setting, analyzed, and tested on realistic problems in [8]. Here, we will not recall how the different variants have been derived in the literature. Instead, we will present a new, simple, and unified approach for their derivation based on the multiple shooting method applied to (2.1).

In the multiple shooting method the time interval $[0, T]$ is partitioned into N subintervals, not necessarily of equal length, determined by the time-points $0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = T$. A system of N separate initial value problems is posed,

$$\left\{ \begin{array}{rclrcl} \boldsymbol{u}_0' &=& \boldsymbol{f}(\boldsymbol{u}_0), & \boldsymbol{u}_0(0) &=& \boldsymbol{U}_0, \\ \boldsymbol{u}_1' &=& \boldsymbol{f}(\boldsymbol{u}_1), & \boldsymbol{u}_1(t_1) &=& \boldsymbol{U}_1, \\ & \vdots & & & \vdots & \\ \boldsymbol{u}_{N-1}' &=& \boldsymbol{f}(\boldsymbol{u}_{N-1}), & \boldsymbol{u}_{N-1}(t_{N-1}) &=& \boldsymbol{U}_{N-1}, \end{array} \right. \tag{2.4}$$

together with the matching conditions

$$\boldsymbol{U}_0 - \boldsymbol{u}_0 = 0, \quad \boldsymbol{U}_1 - \boldsymbol{u}_0(t_1, \boldsymbol{U}_0) = 0, \quad \ldots, \quad \boldsymbol{U}_N - \boldsymbol{u}_{N-1}(T, \boldsymbol{U}_{N-1}) = 0. \tag{2.5}$$

These matching conditions form a nonlinear system of equations

$$\boldsymbol{F}(\boldsymbol{U}) = \boldsymbol{0}, \quad \boldsymbol{U} = (\boldsymbol{U}_0, \boldsymbol{U}_1, \ldots, \boldsymbol{U}_N)^T. \tag{2.6}$$

Solving this system with Newton's method leads to

$$\boldsymbol{U}^{k+1} = \boldsymbol{U}^k - J_F^{-1}(\boldsymbol{U}^k)\boldsymbol{F}(\boldsymbol{U}^k), \tag{2.7}$$

where J_F denotes the Jacobian of $\boldsymbol{F}$. Using the particular structure of the function $\boldsymbol{F}$ in (2.5), we can write the Newton update $J_F^{-1}(\boldsymbol{U}^k)\boldsymbol{F}(\boldsymbol{U}^k)$ in the form

$$\begin{bmatrix} I & & & & \\ -\frac{\partial u_0}{\partial U_0}(t_1, U_0^k) & I & & & \\ & -\frac{\partial u_1}{\partial U_1}(t_2, U_1^k) & I & & \\ & & \ddots & \ddots & \\ & & & -\frac{\partial u_{N-1}}{\partial U_{N-1}}(t_{n-1}, U_{N-1}^k) & I \end{bmatrix}^{-1} \begin{pmatrix} U_0^k - u_0 \\ U_1^k - u_1(t_1, U_0^k) \\ U_2^k - u_1(t_2, U_1^k) \\ \vdots \\ U_N^k - u_{N-1}(T, U_1^k) \end{pmatrix}.$$

Rearranging (2.7) by multiplying through by the Jacobian, we find the basic recurrence of the multiple shooting method applied to initial value problems,

$$\left\{ \begin{array}{rcl} \boldsymbol{U}_0^{k+1} &=& \boldsymbol{u}_0, \\ \boldsymbol{U}_{n+1}^{k+1} &=& \boldsymbol{u}_n(t_{n+1}, \boldsymbol{U}_n^k) + \frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}(t_{n+1}, \boldsymbol{U}_n^k)(\boldsymbol{U}_n^{k+1} - \boldsymbol{U}_n^k). \end{array} \right. \tag{2.8}$$

While the formulation of (2.8) is entirely at the continuous level, to apply the multiple shooting algorithm one usually needs to discretize the differential equations on each subinterval. That is, one needs to select a numerical method for computing $\boldsymbol{u}_n(t_{n+1}, \boldsymbol{U}_n^k)$. In addition, one has to decide on how to compute or approximate the terms $\frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}(t_{n+1}, \boldsymbol{U}_n^k)$, or the action of those terms on the difference $\boldsymbol{U}_n^{k+1} - \boldsymbol{U}_n^k$.

We will show in a series of remarks that different choices of these approximations have led to different parareal versions.

Remark 2.1. If in the multiple shooting method one approximates the subinterval solves by $\boldsymbol{u}_n(t_{n+1}, \boldsymbol{U}_n^k) = F(t_{n+1}, t_n, \boldsymbol{U}_n^k)$, and if one approximates the second term in the right-hand side of (2.8) in a finite difference way using the G-propagator,

$$\frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}(t_{n+1}, \boldsymbol{U}_n^k)(\boldsymbol{U}_n^{k+1} - \boldsymbol{U}_n^k) = G(t_{n+1}, t_n, \boldsymbol{U}_n^{k+1}) - G(t_{n+1}, t_n, \boldsymbol{U}_n^k),$$

then the multiple shooting algorithm (2.8) and the parareal algorithm (2.2) coincide.

Remark 2.2. In [4], Bellen and Zennaro describe a parallel solver for the computation of a general first order recurrence relation of the form $y_{n+1} = F_{n+1}(y_n)$. The function F_{n+1} represents, for example, a one-step numerical discretization of an initial value problem. The method of [4] can be recovered from the multiple shooting algorithm (2.8) if the computation of $\boldsymbol{u}_n(t_{k+1}, \boldsymbol{U}_n^k)$ is approximated as $F_{n+1}(\boldsymbol{U}_n^k)$ and if the columns of the matrix $\frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}(t_{n+1}, \boldsymbol{U}_n^k)$ are approximated individually by a finite difference method. As the authors point out, these columns can be approximated concurrently on d processors, where d is the dimension of (2.1), by applying the function F_{n+1} to d vectors that are each single component perturbations of $\boldsymbol{U}_n^k$.

Remark 2.3. In shooting methods, the Jacobian may be computed by solving the so-called variational equations, which, in our case, are given by

$$\left(\frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}\right)' = \boldsymbol{f}'(\boldsymbol{u}_n)\frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}, \quad \frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}(t_n) = I, \qquad n = 0, 1, \ldots, N-1. \tag{2.9}$$

Here, I denotes the identity matrix. These linear equations are in general matrix equations, where the matrix size corresponds to the size of the system of equations (2.1). They are classically solved simultaneously with the shooting equations (2.4) using the same method, since the trajectory values $\boldsymbol{u}_n$ are needed in (2.9). If one approximates these equations on the coarse mesh t_n, $n = 0, 1, \ldots, N$, only, then one obtains the original version of the parareal algorithm proposed for nonlinear problems in [19]. Since (2.9) is a matrix equation and one needs only the action of the Jacobian on the vector of differences in (2.8), it is computationally more effective to approximate on the coarse grid the vector valued equation

$$\boldsymbol{v}_n' = \boldsymbol{f}'(\boldsymbol{u}_n)\boldsymbol{v}_n, \quad \boldsymbol{v}_n(t_n) = \boldsymbol{U}_n^{k+1} - \boldsymbol{U}_n^k, \qquad n = 0, 1, \ldots, N-1, \tag{2.10}$$

to obtain $\boldsymbol{v}_n(t_{n+1}) = \frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}(t_{n+1}, \boldsymbol{U}_n^k)(\boldsymbol{U}_n^{k+1} - \boldsymbol{U}_n^k)$. This is done in the so-called PITA algorithm for nonlinear problems proposed in [8] and is mathematically equivalent to approximating (2.9) on the coarse mesh, before applying the matrix valued solution to $\boldsymbol{U}_n^{k+1} - \boldsymbol{U}_n^k$. For linear problems, this approach coincides with formulation (2.2) of the parareal algorithm.

3. Parareal as a time-multigrid method. We will show in this section that iteration (2.2) can be cast into the classical framework of nonlinear multigrid methods. Consider a nonlinear system of equations of the form

$$A_h(\boldsymbol{u}) = \boldsymbol{b}. \tag{3.1}$$

Typically, this system is derived by discretization of a PDE on a mesh Ω_h. The two-grid variant of the so-called full approximation storage (FAS) multigrid method for this problem can be written as follows (see [5]):

$$\left\{\begin{array}{rcl} \tilde{\boldsymbol{u}}^k &=& S(\boldsymbol{u}^k, \boldsymbol{b}), \\ A_H(\boldsymbol{U}^{k+1}) &=& I_h^H(\boldsymbol{b} - A_h(\tilde{\boldsymbol{u}}^k)) + A_H(I_h^H \tilde{\boldsymbol{u}}^k), \\ \boldsymbol{u}^{k+1} &=& \tilde{\boldsymbol{u}}^k + I_H^h(\boldsymbol{U}^{k+1} - I_h^H \tilde{\boldsymbol{u}}^k). \end{array}\right. \tag{3.2}$$

The algorithm is used to compute a series of approximations $\boldsymbol{u}^k$ for $k = 1, 2, \ldots$, starting from a given initial estimate $\boldsymbol{u}^0$, to the solution of problem (3.1). The first step in the two-grid algorithm is called the smoothing step and is fully defined by the smoothing operator S. The second equation in (3.2) defines a coarse grid problem and is characterized by the operator A_H. This operator is usually a discretization of a partial differential operator on a coarse mesh Ω_H. Operator I_h^H is called the restriction operator and transfers discrete functions from the fine mesh Ω_h to the coarse one. Finally, the third step of the algorithm is the correction step. There, the approximation $\tilde{\boldsymbol{u}}^k$ generated by the smoother is updated by transferring a correction from the coarse mesh to the fine mesh by using a prolongation operator I_H^h.

We will show that the FAS components can be selected in such a way that the iteration defined by (3.2) is identical to the parareal iteration given in (2.2). First, we consider the operators A_h and A_H. They are defined by a discretization of the possibly nonlinear evolution equation (2.1) on a fine mesh and on a coarse mesh, respectively. For notational convenience, we will assume those meshes to be regular, with step size h for Ω_h and with step size Mh for Ω_H for some integer $M > 1$. We introduce two arbitrary one-step methods: for the fine grid approximation,

$$\boldsymbol{u}_m = \phi_m(\boldsymbol{u}_{m-1}) + \boldsymbol{b}_m, \quad \boldsymbol{u}_m \approx \boldsymbol{u}(mh), \quad m = 1, 2, \ldots, MN, \tag{3.3}$$

and for the coarse grid approximation,

$$\boldsymbol{U}_n = \Phi_n(\boldsymbol{U}_{n-1}) + \boldsymbol{B}_n, \quad \boldsymbol{U}_n \approx \boldsymbol{u}(nH), \quad n = 1, 2, \ldots, N. \tag{3.4}$$

In the case of a linear problem, the function $\phi_m(\cdot)$ corresponds to a matrix-vector product, with a matrix which we shall denote as ϕ_m. The vector $\boldsymbol{b}_m$ is a vector that collects known, solution independent values. We will extend the sets of equations (3.3) and (3.4) with an additional equation of the form $\boldsymbol{u}_0 = \boldsymbol{b}_0$, or $\boldsymbol{U}_0 = \boldsymbol{B}_0$, where both $\boldsymbol{b}_0$ and $\boldsymbol{B}_0$ are equal to the initial condition. Collecting the vectors of the fine grid approximation $\boldsymbol{u}_m$ in one large vector $\boldsymbol{u}$ and the vectors of the coarse grid approximation in one large vector $\boldsymbol{U}$, we can write the recurrence relation for the fine and coarse grid approximations in the form $A_h(\boldsymbol{u}) = \boldsymbol{b}$ and $A_H(\boldsymbol{u}) = \boldsymbol{B}$, respectively. In the case of a linear problem, we can write the former as a linear matrix equation of the form $A_h \boldsymbol{u} = \boldsymbol{b}$, with

$$A_h := \begin{bmatrix} I & & & \\ -\phi_1 & I & & \\ & \ddots & \ddots & \\ & & -\phi_{MN} & I \end{bmatrix}, \quad \boldsymbol{u} = \begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{NM} \end{pmatrix}, \quad \text{and } \boldsymbol{b} = \begin{pmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_{NM} \end{pmatrix}. \tag{3.5}$$

A similar notation could be used for the coarse mesh problem $A_H(\boldsymbol{U}) = \boldsymbol{B}$.

We will now demonstrate that parareal coincides with a time-multigrid method for a particular—be it somewhat unusual—choice of smoother, restriction, and interpolation operators. The smoother is defined recursively as follows:

$$\begin{cases} S(\boldsymbol{u}^k, \boldsymbol{b})_{nM} := \boldsymbol{u}^k_{nM} \quad \text{for } n = 0, 1, \ldots, N, \\ S(\boldsymbol{u}^k, \boldsymbol{b})_{nM+m} := \phi_{nM+m}(S(\boldsymbol{u}^k, \boldsymbol{b})_{nM+m-1}) + \boldsymbol{b}_{nM+m} \\ \qquad\qquad \text{for } m = 1, \ldots, M-1 \text{ and } n = 0, 1, \ldots, N-1. \end{cases} \tag{3.6}$$

That is, the smoother leaves the components of $\boldsymbol{u}^k$ with indices nM for $n = 0, 1, \ldots, N$ unchanged and fills out the intermediate components by using the recurrence (3.3). In the linear case, this smoother can be written in a more classical matrix formulation. To this end, we define a block Jacobi splitting of A_h,

$$A_h = M_{jac} + N_{jac}, \tag{3.7}$$

where the blocks are of size $M \times M$, except for the first block, which is of size $(M+1) \times (M+1)$, since it also includes the initial condition. We also introduce the matrix E, which is the identity of the same size as A_h, except for zeros on the diagonal at positions $0, M, 2M, \ldots, NM$. With this notation, we have

$$S(\boldsymbol{u}^k, b) := \boldsymbol{u}^k + EM^{-1}_{jac}(\boldsymbol{b} - A_h \boldsymbol{u}^k). \tag{3.8}$$

The method is based on a classical block Jacobi method, but the update of the $\boldsymbol{u}^k_{nM}$ values is prevented by means of the multiplication with the matrix E. Alternatively, the smoother can be interpreted as one phase of a classical two-color scheme, e.g., a red-black method. Let the mesh points be partitioned into a set of "black" points, in our case the points with indices nM, and let the remaining points be in the set of "red" points. One iteration of a standard red-black relaxation method would consist of an update of the red points, followed by an update of the black points. The smoother as defined in (3.6) corresponds to the first, i.e., red, step only.

For the analogy with the parareal scheme to go through, we need the restriction and prolongation operators to be, respectively, the standard injection operator and its transpose. More precisely,

$$(I^H_h \boldsymbol{u})_n := \boldsymbol{u}_{nM} \text{ and } (I^h_H \boldsymbol{U})_{nM+m} := \begin{cases} \boldsymbol{U}_n & \text{for } m = 0, \\ 0 & \text{for } m = 1, 2, \ldots, M-1. \end{cases} \tag{3.9}$$

That is, the restriction selects the entries at positions $0, M, \ldots, NM$ in the vector to which it is applied. The prolongation operator applied to a vector of length $N+1$ copies its entries into a vector of length $NM+1$ at the positions $0, M, \ldots, NM$, and fills in zero values in between. With all of the operators of the FAS method defined, we can now specify the components of the parareal algorithm. We consider $F(t_{n+1}, t_n, \cdot)$ to be composed of a sequence of steps of the form (3.3). More precisely,

$$\begin{cases} F(t_{n+1}, t_n, \boldsymbol{U}_n) := \boldsymbol{u}_{(n+1)M}, \text{ with } \quad \boldsymbol{u}_{nM} = \boldsymbol{U}_n, \\ \text{and } \quad \boldsymbol{u}_m = \phi_m(\boldsymbol{u}_{m-1}) + \boldsymbol{b}_m \text{ for } \quad m = nM+1, \ldots, (n+1)M. \end{cases} \tag{3.10}$$

In a similar way, the propagator $G(t_{n+1}, t_n, \cdot)$ is defined as a single step of (3.4), i.e.,

$$G(t_{n+1}, t_n, \boldsymbol{U}_n) := \Phi_{n+1}(\boldsymbol{U}_n) + \boldsymbol{B}_{n+1}. \tag{3.11}$$

THEOREM 3.1. *Let the F- and G-propagators of the parareal algorithm be defined as in* (3.10) *and* (3.11). *Let the smoother, restriction, and prolongation operators of the multigrid method be defined as in* (3.8) *and* (3.9), *and let the operators $A_h(\cdot)$ and $A_H(\cdot)$ be constructed from* (3.3) *and* (3.4). *If the initial approximation $\boldsymbol{u}^0$ satisfies $\boldsymbol{u}^0_{nM} = \boldsymbol{U}^0_n$, for $n = 0, 1, \ldots, N$, then the parareal algorithm* (2.2) *for solving* (2.1) *coincides with the two-grid method* (3.2) *for solving* (3.1), *in the sense that $\boldsymbol{U}^k_n = \boldsymbol{u}^k_{nM}$ for $n = 0, 1, \ldots, N$ and for all $k \geq 0$.*

Proof. We shall prove the result by induction on k; i.e., we show that $\boldsymbol{U}^{k+1}_n = \boldsymbol{u}^{k+1}_{nM}$ if $\boldsymbol{U}^k_n = \boldsymbol{u}^k_{nM}$. For $k = 0$, the equality holds by the assumption of the theorem.

We rewrite each of the terms in the second equation of the FAS scheme (3.2). For the nth element of the left-hand side, we find

$$\Big(A_H(\boldsymbol{U}^{k+1})\Big)_n = -\Phi_n(\boldsymbol{U}^{k+1}_{n-1}) + \boldsymbol{U}^{k+1}_n.$$

The first term on the right-hand side leads to

$$\Big(I^H_h(\boldsymbol{b} - A_h(\tilde{\boldsymbol{u}}^k))\Big)_n = \Big(\boldsymbol{b} - A_h(\tilde{\boldsymbol{u}}^k)\Big)_{nM} = \boldsymbol{b}_{nM} + \phi_{nM}(\tilde{\boldsymbol{u}}^k_{nM-1}) - \tilde{\boldsymbol{u}}^k_{nM}.$$

For the second term on the right-hand side we find

$$\Big(A_H(I^H_h\tilde{\boldsymbol{u}}^k)\Big)_n = -\Phi_n((I^H_h\tilde{\boldsymbol{u}}^k)_{n-1}) + (I^H_h\tilde{\boldsymbol{u}}^k)_n = -\Phi_n(\tilde{\boldsymbol{u}}^k_{(n-1)M}) + \tilde{\boldsymbol{u}}^k_{nM}.$$

Hence, the second equation of (3.2) can be rewritten componentwise as

$$-\Phi_n(\boldsymbol{U}^{k+1}_{n-1}) + \boldsymbol{U}^{k+1}_n = \boldsymbol{b}_{nM} + \phi_{nM}(\tilde{\boldsymbol{u}}^k_{nM-1}) - \Phi_n(\boldsymbol{u}^k_{(n-1)M}),$$

where we have used the property that the smoother leaves the elements at positions nM, for $n = 0, 1, \ldots, N$, unchanged. Adding and subtracting $\boldsymbol{B}_n$, we obtain, after rearranging terms,

$$\boldsymbol{U}^{k+1}_n = \Phi_n(\boldsymbol{U}^{k+1}_{n-1}) + \boldsymbol{B}_n + \phi_{nM}(\tilde{\boldsymbol{u}}^k_{nM-1}) + \boldsymbol{b}_{nM} - \Phi_n(\boldsymbol{u}^k_{(n-1)M}) - \boldsymbol{B}_n.$$

By using the induction hypothesis, i.e., $\boldsymbol{U}^k_n = \boldsymbol{u}^k_{nM}$, and the definition of the F- and G-propagators and the smoother, we get

$$\boldsymbol{U}^{k+1}_n = G(t_n, t_{n-1}, \boldsymbol{U}^{k+1}_{n-1}) + F(t_n, t_{n-1}, \boldsymbol{U}^k_{n-1}) - G(t_n, t_{n-1}, \boldsymbol{U}^k_{n-1}).$$

This proves that the coarse grid FAS solution is precisely the solution generated by the parareal algorithm (2.2). Finally, we consider the last equation of (3.2),

$$\boldsymbol{u}^{k+1}_{nM} = \Big(\tilde{\boldsymbol{u}}^k + I^h_H(\boldsymbol{U}^{k+1} - I^H_h\tilde{\boldsymbol{u}}^k)\Big)_{nM} = (I^h_H\boldsymbol{U}^{k+1})_{nM} = \boldsymbol{U}^{k+1}_n,$$

where the second equality follows from the fact that the operator $I^h_H I^H_h$ acts as the identity on the components $\boldsymbol{u}_{nM}$ of any vector $\boldsymbol{u}$ to which it is applied. This proves the theorem. ∎

Remark 3.2. The coarse grid problem defined in the second equation of (3.2) is of the same type as the fine grid problem $A_h(\boldsymbol{u}) = \boldsymbol{b}$. Hence, it could be solved by a recursive application of the same method, as in classical multigrid.

Remark 3.3. Note that the smoother (3.6) is not convergent (but not divergent either). Actually, in the linear case, it can easily be shown that the spectral radius of

the iteration matrix is equal to one. This type of smoother, which operates only on the fine grid points and leaves the coarse points unchanged, is sometimes called an F-smoother; see [30].

Remark 3.4. With the multigrid components as defined above, the iteration (3.2) is consistent with the fine grid problem. That is, the solution of $A_h(\boldsymbol{u}) = \boldsymbol{b}$ is a fixed point of the iteration. The convergence of the algorithm towards that solution is proved in the next two sections.

4. Convergence analysis: The scalar ODE case. In this section, we first recall a convergence result presented earlier in the literature and then derive two new convergence theorems. The first result is valid on bounded time intervals, $T < \infty$, whereas the second one also holds for unbounded time intervals.

4.1. Some results from the literature. The purpose of our study is to derive the conditions under which the algorithm will converge and to derive an expression for the convergence speed. Our results are different from those in the literature, where typically the number of iterations of (2.2) is fixed to a finite value, and the convergence is studied as ΔT goes to zero. In the fixed number of iterations case, (2.2) defines a new time-integration scheme. The accuracy of the $\boldsymbol{U}_n^k$ values is characterized in a proposition from the first publication on the parareal algorithm [19], which applies for a scalar linear problem of the form

$$u' = au, \quad u(0) = u_0, \quad t \in [0, T] \ \text{ with } \ a \in \mathbb{C}. \tag{4.1}$$

PROPOSITION 4.1. *Let $\Delta T = T/N$, $t_n = n\Delta T$ for $n = 0, 1, \ldots, N$. Consider (4.1) with $a \in \mathbb{R}$. Let $F(t_{n+1}, t_n, U_n^k)$ be the exact solution at t_{n+1} of (4.1) with $u(t_n) = U_n^k$, and let $G(t_{n+1}, t_n, U_n^k)$ be the corresponding backward Euler approximation with time step ΔT. Then,*

$$\max_{1 \le n \le N} |u(t_n) - U_n^k| \le C_k \Delta T^{k+1}. \tag{4.2}$$

Hence, for a fixed iteration step k, the algorithm behaves in ΔT like an $O(\Delta T^{k+1})$ method. This theorem was extended to more general and, in particular, higher order time-integration schemes in [3, 2]. In those papers it was shown that parareal is a method of $O(\Delta T^{p(k+1)})$ when a method of order p is used as the coarse propagator, combined with an exact solver or a sufficiently accurate solver as the fine propagator.

In our analysis we will fix ΔT and study the algorithm's behavior as k goes to infinity. This case is not covered by the above theorem, because the constant C_k in (4.1) grows with k in the estimate of the proof in [19].

4.2. Some preparatory lemmas. In our analysis an important role will be played by a strictly lower triangular Toeplitz matrix $M := M(\beta)$ of size N. Its elements are fully defined by the values of the elements in its first column,

$$M_{i1} = \begin{cases} 0 & \text{if } \ i = 1, \\ \beta^{i-2} & \text{if } \ 2 \le i \le N \end{cases} \tag{4.3}$$

for some $\beta \in \mathbb{C}$. The analysis will involve the powers of the matrix M. Those matrices maintain the strictly lower triangular Toeplitz structure of M. The elements of those matrices, their norms, and a bound on their norms are identified in the lemmas below.

LEMMA 4.2. *The ith element in the first column of the kth power of M is*

$$M^k_{i,1} = \begin{cases} 0 & \text{if } 1 \le i \le k, \\ \binom{i-2}{k-1}\beta^{i-1-k} & \text{if } k+1 \le i \le N. \end{cases} \tag{4.4}$$

Proof. The proof is based on induction on k. The matrix M is strictly lower triangular with a nonzero first subdiagonal. Hence, the multiplication of M^{k-1} with M generates exactly one additional zero subdiagonal. This leads to a total of k zero subdiagonals for the matrix M^k, which proves the first part of the lemma.

The ith element in the first column of M^k, for $i \ge k+1$, can be computed as follows:

$$M^k_{i,1} = \sum_{j=1}^{N} M^{k-1}_{i,j} \cdot M_{j,1} = \sum_{j=2}^{i-k+1} M^{k-1}_{i+1-j,1} \cdot M_{j,1} = \sum_{j=0}^{i-k-1} M^{k-1}_{i-1-j,1} \cdot M_{j+2,1},$$

where we have used the zero structure of M^{k-1} and M, and the Toeplitz nature of M^{k-1}. Using the induction hypothesis and (4.3), we can assign values to the matrix elements in the above sum; then we reverse the summation order to find

$$M^k_{i,1} = \sum_{j=0}^{i-k-1} \binom{i-3-j}{k-2} \beta^{i-k-1} = \sum_{J=0}^{i-k-1} \binom{J+k-2}{k-2} \beta^{i-k-1}.$$

The result now follows by applying the well-known summation formula

$$\sum_{s=0}^{r} \binom{n+s}{n} = \binom{n+r+1}{n+1}, \tag{4.5}$$

with $s = J$, $r = i-k-1$, and $n = k-2$. ∎

LEMMA 4.3. *The infinity norm of the kth power of M is given by*

$$||M^k||_\infty = \begin{cases} \binom{N-1}{k} & \text{if } |\beta| = 1, \\ \dfrac{1}{(k-1)!} \dfrac{d^{k-1}}{dz^{k-1}} \left(\dfrac{z^{N-1}-1}{z-1} \right) \text{ for } z = |\beta| & \text{if } |\beta| \ne 1. \end{cases} \tag{4.6}$$

Proof. The largest row sum of the entries in modulus, i.e., $||M^k||_\infty$, is the row sum in modulus of the last row, since all of the other rows contain less of the same entries than the last one by the structure of M^k. Because of the strictly lower triangular Toeplitz structure, we actually have $||M^k||_\infty = ||M^k||_1$. Hence,

$$||M^k||_\infty = \sum_{i=k+1}^{N} |M^k_{i,1}| = \sum_{i=k+1}^{N} \binom{i-2}{k-1} |\beta|^{i-1-k} = \sum_{i=0}^{N-k-1} \binom{i+k-1}{k-1} |\beta|^i \,. \tag{4.7}$$

The result for $|\beta| = 1$ now follows by applying (4.5) with $n = k-1$ and $r = N-k-1$. If $|\beta| \ne 1$, we proceed as follows:

$$||M^k||_\infty = \frac{1}{(k-1)!} \sum_{i=0}^{N-k-1} (i+k-1)(i+k-2)\dots(i+1)|\beta|^i$$

$$= \frac{1}{(k-1)!} \sum_{i=0}^{N-k-1} \frac{d^{k-1}}{dz^{k-1}} z^{i+k-1} \quad \text{for } z = |\beta|$$

$$= \frac{1}{(k-1)!} \frac{d^{k-1}}{dz^{k-1}} \sum_{i=0}^{N-k-1} z^{i+k-1} \quad \text{for } z = |\beta|$$

$$= \frac{1}{(k-1)!} \frac{d^{k-1}}{dz^{k-1}} \sum_{I=0}^{N-2} z^{I} \quad \text{for } z = |\beta|.$$

Summing the geometric series leads to the desired result. ∎

LEMMA 4.4. *The infinity norm of the kth power of M is bounded as follows:*

$$||M^k||_\infty \le \begin{cases} \min\left\{ \left(\dfrac{1-|\beta|^{N-1}}{1-|\beta|} \right)^k, \dbinom{N-1}{k} \right\} & \text{if } |\beta| < 1, \\ |\beta|^{N-k-1} \dbinom{N-1}{k} & \text{if } |\beta| \ge 1. \end{cases} \tag{4.8}$$

Proof. The bound for $|\beta| \ge 1$ follows from (4.7) by replacing $|\beta|^i$ by the larger value $|\beta|^{N-k-1}$ and by using (4.5) to evaluate the summation.

Part of the bound for $|\beta| < 1$ follows from (4.7) by replacing $|\beta|^i$ by its upper bound 1. The other part can be derived as follows:

$$||M^k||_\infty \le ||M||_\infty^k = \left(\sum_{i=0}^{N-2} |\beta|^i \right)^k = \left(\frac{1-|\beta|^{N-1}}{1-|\beta|} \right)^k. \quad \blacksquare$$

4.3. A superlinear convergence result on bounded intervals. We derive the conditions for convergence of the parareal algorithm and an estimate for the convergence speed. We consider the classical model problem (4.1) and, initially, assume that the F-propagator is an exact solver. This restriction will be removed in section 4.5, where we consider the use of an approximate F-propagator. The coarse G-propagator is assumed to be a one-step method $G(t_{n+1}, t_n, U_n^k) = R(a\Delta T)U_n^k$, characterized by its stability function $R(z)$. For example, for a Runge–Kutta method with the Butcher tableau coefficients contained in the matrix $A = [a_{ij}]$ and the vector $\boldsymbol{b} = [b_j]$, we have

$$R(z) = 1 + z\boldsymbol{b}^T(I - zA)^{-1}\mathbf{1},$$

where I denotes the identity matrix, and $\mathbf{1}$ the vector containing all 1's.

THEOREM 4.5. *Let $T < \infty$, $\Delta T = T/N$, and $t_n = n\Delta T$ for $n = 0, 1, \ldots, N$. Let $F(t_{n+1}, t_n, U_n^k)$ be the exact solution at t_{n+1} of (4.1) with $u(t_n) = U_n^k$, and let $G(t_{n+1}, t_n, U_n^k) = R(a\Delta T)U_n^k$ be a one-step method. Then, with $M(\beta)$ defined in (4.3), we have the bound*

$$\begin{aligned} &\max_{1\le n\le N} |u(t_n) - U_n^k| \\ &\qquad \le |e^{a\Delta T} - R(a\Delta T)|^k \|M^k(R(a\Delta T))\|_\infty \max_{1\le n\le N} |u(t_n) - U_n^0|. \end{aligned} \tag{4.9}$$

Proof. We denote by e_n^k the error at iteration step k of the parareal algorithm at time t_n; i.e., $e_n^k := u(t_n) - U_n^k$. With (2.2) and an induction argument on n, this error

satisfies

$$e_n^{k+1} = R(a\Delta T)e_{n-1}^{k+1} + (e^{a\Delta T} - R(a\Delta T))e_{n-1}^k$$
$$= (e^{a\Delta T} - R(a\Delta T)) \sum_{j=1}^{n-1} R(a\Delta T)^{n-j-1} e_j^k.$$

This relation can be written in matrix form by collecting the error components of the kth iterate e_n^k in the vector $\boldsymbol{e}^k = (e_1^k, e_2^k \ldots, e_N^k)^T$, which leads to

$$\boldsymbol{e}^{k+1} = (e^{a\Delta T} - R(a\Delta T))M(R(a\Delta T))\boldsymbol{e}^k. \tag{4.10}$$

By induction on k in (4.10), we obtain

$$\boldsymbol{e}^k = (e^{a\Delta T} - R(a\Delta T))^k \left(M(R(a\Delta T))\right)^k \boldsymbol{e}^0, \tag{4.11}$$

and taking norms, the result follows. ∎

The bound in (4.9) can be computed by replacing the norm of $M^k(R(a\Delta T))$ by its value as given in Lemma 4.3 or by the bound given in Lemma 4.4. Of particular interest is the case of a one-step method used in its region of absolute stability, i.e., with $|R(a\Delta T)| < 1$.

COROLLARY 4.6. *Let $T < \infty$, $\Delta T = T/N$, and $t_n = n\Delta T$ for $n = 0, 1, \ldots, N$. Let $F(t_{n+1}, t_n, U_n^k)$ be the exact solution at t_{n+1} of (4.1) with $u(t_n) = U_n^k$, and let $G(t_{n+1}, t_n, U_n^k) = R(a\Delta T)U_n^k$ be a one-step method in its region of absolute stability. Then, we have the bound*

$$\max_{1\le n\le N} |u(t_n) - U_n^k| \le \frac{|e^{a\Delta T} - R(a\Delta T)|^k}{k!} \prod_{j=1}^{k} (N-j) \max_{1\le n\le N} |u(t_n) - U_n^0|. \tag{4.12}$$

If the local truncation error of G is bounded by $C\Delta T^{p+1}$, with $p > 0$ and C a constant, then we have, for ΔT small enough,

$$\max_{1\le n\le N} |u(t_n) - U_n^k| \le \frac{(CT)^k}{k!} \Delta T^{pk} \max_{1\le n\le N} |u(t_n) - U_n^0|. \tag{4.13}$$

Proof. The first result follows from (4.9) and the norm estimate in Lemma 4.4 for $\beta := R(a\Delta T)$, considering that $|\beta| < 1$. The bound (4.13) follows from the bound on the local truncation error together with a simple estimate of the product,

$$\frac{|e^{a\Delta T} - R(a\Delta T)|^k}{k!} \prod_{j=1}^{k} (N-j) \le \frac{C^k \Delta T^{(p+1)k}}{k!} N^k = \frac{(CT)^k}{k!} \Delta T^{pk}. \qquad ∎$$

Remark 4.7. The product term in (4.12) shows that the parareal algorithm terminates with a converged solution for any ΔT on any bounded time interval in at most $N-1$ steps. This property actually holds without restriction on the magnitude of $R(a\Delta T)$ or the quality of the G-operator. Typically, however, one would like to stop the iteration with a converged or sufficiently accurate solution well before $N-1$ iteration steps, since otherwise there is no speedup in the parallel process.

Remark 4.8. Observe that the algorithm converges superlinearly, as the division by $k!$ in (4.12) shows. This follows also from the value of the spectral radius of the error propagation operator in (4.10), which is zero.

4.4. A linear convergence result on long time intervals. Next we consider the parareal algorithm as a time-integration method on an infinitely long time interval. In the first theorem of this section we show that the convergence is not worse than linear, and we construct an upper bound for the iteration error. In the second theorem we show that the bound is asymptotically sharp. That is, the asymptotic convergence rate is exactly linear.

THEOREM 4.9. *Let ΔT be given, and $t_n = n\Delta T$ for $n = 0, 1, \ldots$. Let $F(t_{n+1}, t_n, U_n^k)$ be the exact solution at t_{n+1} of (4.1) with $u(t_n) = U_n^k$, and let $G(t_{n+1}, t_n, U_n^k) = R(a\Delta T)U_n^k$ be a one-step method in its region of absolute stability. Then,*

$$\sup_{n>0} |u(t_n) - U_n^k| \leq \left(\frac{|e^{a\Delta T} - R(a\Delta T)|}{1 - |R(a\Delta T)|} \right)^k \sup_{n>0} |u(t_n) - U_n^0|. \tag{4.14}$$

If the local truncation error of G is bounded by $C\Delta T^{p+1}$, with $p > 0$ and C a constant, then we have for ΔT small enough

$$\sup_{n>0} |u(t_n) - U_n^k| \leq \left(\frac{C\Delta T^p}{\Re(-a) + O(\Delta T)} \right)^k \sup_{n>0} |u(t_n) - U_n^0|. \tag{4.15}$$

Proof. In the present case M, as defined in (4.3), is an infinite dimensional Toeplitz operator. Its infinity norm is given by

$$||M(a\Delta T)||_\infty = \sum_{j=0}^{\infty} |R(a\Delta T)|^j = \frac{1}{1 - |R(a\Delta T)|},$$

where we have used that $|R(a\Delta T)| < 1$; i.e., G operates in its region of absolute stability. Using (4.10), we obtain for the error vectors $\boldsymbol{e}^k$ of infinite length the relation

$$||\boldsymbol{e}^k||_\infty \leq |e^{a\Delta T} - R(a\Delta T)|^k ||M||_\infty^k ||\boldsymbol{e}^0||_\infty = \left(\frac{|e^{a\Delta T} - R(a\Delta T)|}{1 - |R(a\Delta T)|} \right)^k ||\boldsymbol{e}^0||_\infty,$$

which proves the first result.

Next, we can use the bound on the local truncation error in order to estimate the numerator of (4.14), i.e., $|e^{a\Delta T} - R(a\Delta T)| \leq C\Delta T^{p+1}$. This bound implies for a strictly positive integer p that $R(a\Delta T) = 1 + a\Delta T + O(\Delta T^2)$. From this, we find

$$\begin{aligned} |R(a\Delta T)| &= R(a\Delta T)^{1/2} \overline{R(a\Delta T)}^{1/2} \\ &= (1 + 1/2 a\Delta T + O(\Delta T^2))(1 + 1/2 \bar{a}\Delta T + O(\Delta T^2)), \end{aligned}$$

which implies $1 - |R(a\Delta T)| = \Re(-a)\Delta T + O(\Delta T^2)$. Thus, (4.15) follows. ∎

The parareal algorithm can be seen as an iteration which maps infinite sequences into infinite sequences. More precisely, it maps the discrete sequence $U^k := \{U_n^k\}_{n=0}^\infty$ into the discrete sequence $U^{k+1} := \{U_n^{k+1}\}_{n=0}^\infty$. In operator notation we can write the iteration compactly as $U^{k+1} = \mathcal{K}(a\Delta T)U^k$, with $\mathcal{K}(a\Delta T)$ the infinite dimensional iteration operator. When applied to model problem (4.1), the parareal iteration operator becomes an infinite dimensional triangular Toeplitz operator or, equivalently, a discrete convolution operator. Such an operator is fully characterized by the elements of its first column a_i, $i = 1, 2, \ldots$. The spectral radius ρ of a Toeplitz operator $\mathcal{T}$ is

known to be related to the operator *symbol* $T(z) := \sum_{i=1}^{\infty} a_i z^{i-1}$. In particular, when $\sum_{i=1}^{\infty} |a_i| < \infty$, we have that

$$\rho(\mathcal{T}) = \max_{|z|\le 1} | \, T(z) \, | = \max_{|z|=1} | \, T(z) \, | \, ; \tag{4.16}$$

see, e.g., [25, Theorem 2.1] for a proof.

THEOREM 4.10. *Let ΔT be given, and $t_n = n\Delta T$ for $n = 0, 1, \ldots$. Let $F(t_{n+1}, t_n, U_n^k)$ be the exact solution at t_{n+1} of (4.1) with $u(t_n) = U_n^k$, and let $G(t_{n+1}, t_n, U_n^k) = R(a\Delta T)U_n^k$ be a one-step method in its region of absolute stability. Then, the asymptotic convergence factor of the parareal algorithm is given by*

$$\rho(\mathcal{K}(a\Delta T)) = \frac{|e^{a\Delta T} - R(a\Delta T)|}{1 - |R(a\Delta T)|}. \tag{4.17}$$

Proof. The parareal iteration operator is characterized by the elements a_i in its first column. From (4.10) we have

$$a_i = (e^{a\Delta T} - R(a\Delta T))M_{i,1}(a\Delta T), \ i = 1, 2, \ldots .$$

Hence, using (4.3) and (4.16) we find

$$\begin{aligned}
\rho(\mathcal{K}(a\Delta T)) &= \max_{|z|=1} \left| \, (e^{a\Delta T} - R(a\Delta T)) \sum_{i=2}^{\infty} R(a\Delta T)^{i-2} z^{i-1} \right| \\
&= \max_{|z|=1} \frac{|e^{a\Delta T} - R(a\Delta T)|}{|1 - R(a\Delta T)z|} \\
&= \max_{\theta\in[0,2\pi]} \frac{|e^{a\Delta T} - R(a\Delta T)|}{|1 - |R(a\Delta T)|e^{i(\arg R(a\Delta T)+\theta)}|}.
\end{aligned}$$

The maximum is found for $\theta = -\arg R(a\Delta T)$, which proves the result. ∎

Remark 4.11. Note that (4.17) is not necessarily smaller than 1. We have that $\rho(\mathcal{K}(a\Delta T)) < 1$ if and only if $|e^{a\Delta T} - R(a\Delta T)| + |R(a\Delta T)| < 1$ for $a \in \mathbb{R}_0^-$. This is equivalent to the condition

$$\frac{e^{a\Delta T} - 1}{2} < R(a\Delta T) < \frac{e^{a\Delta T} + 1}{2}. \tag{4.18}$$

Note that (4.18) is essentially equivalent to the stability condition derived in [29].

4.5. Discussion of the fully discrete case. We consider the case where instead of an exact solution on the subintervals an approximate solver is used as the F-propagator. Typically, this would be a fine grid approximation obtained by repeated application of the G-propagator on a series of small time steps. Alternatively, it could also be the application of one or more steps of a method of higher order than the G-propagator. In those cases the proofs of the theorems derived in sections 4.3 and 4.4 remain valid with some minor modifications. It suffices to replace the term $e^{a\Delta T}$ representing the exact solution by a term of the form $R_f(a\Delta T)$. Here, the function $R_f(z)$ denotes the stability function of the F-propagator as a solver for (4.1) over an interval in time of length ΔT. For example, if the approximation is obtained by M steps of the fine solver, we have $R_f(a\Delta T) = R(a\Delta T/M)^M$.

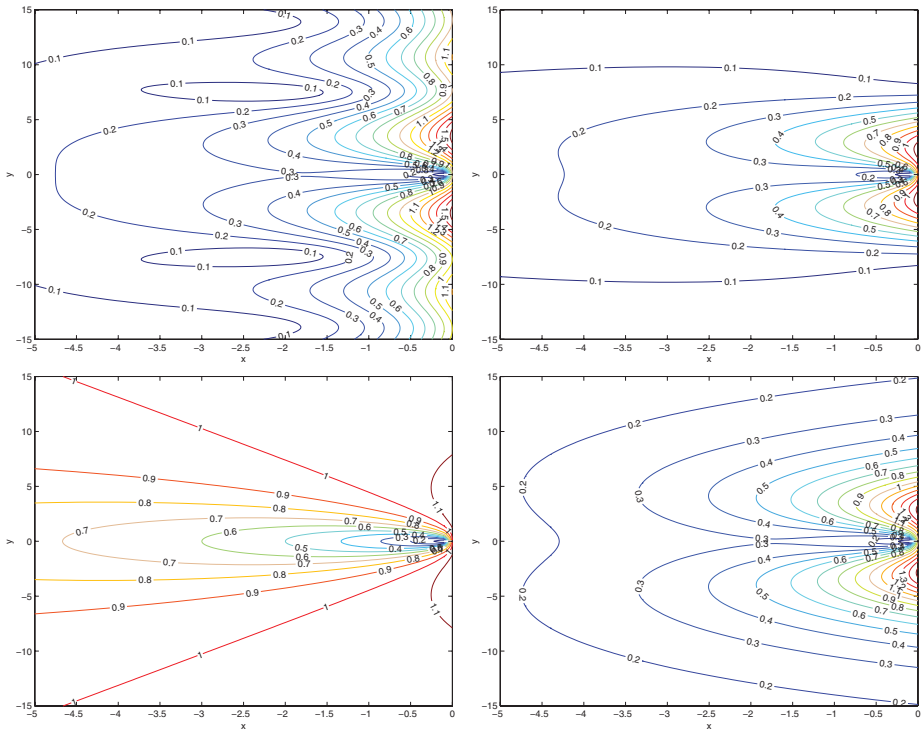

FIG. 4.1. *Contour plots of the convergence factor* $\rho(\mathcal{K}(z))$ *with* $z := a\Delta T \in \mathbb{C}^-$, *for* 4 *different F-propagator/G-propagator combinations: exact solution/backward Euler (top left), backward Euler* ($M = 10$)*/backward Euler (top right), trapezoidal rule* ($M = 1$)*/backward Euler (bottom left), and Radau* IIA ($M = 1$)*/backward Euler (bottom right).*

In Figure 4.1 on the top left, we show a contour plot of the function $\rho(\mathcal{K}(z))$ for $z := a\Delta T \in \mathbb{C}^-$, as defined in (4.17), with $R(z)$ the stability function of the backward Euler method. Contour plots of the linear convergence factor of the fully discrete parareal operator, i.e.,

$$\rho(\mathcal{K}(z)) = \frac{|R_f(z) - R(z)|}{1 - |R(z)|}, \tag{4.19}$$

as a function of $z \in \mathbb{C}^-$, are given in the other three parts of the figure. Each of these illustrates a combination of the backward Euler scheme as the G-propagator with a different F-propagator. On the top right, the effect of using an F-propagator consisting of ten backward Euler steps is shown. On the bottom left, the case of a single step of the trapezoidal rule is illustrated, while on the bottom right, the case of a single step of the three-stage Radau IIA method is considered. Note that when the number M is increased and, hence, the F-propagator becomes more accurate, the value of the fully discrete convergence factor will approach that of (4.17).

Of special interest is the case $a \in \mathbb{R}_0^-$. For this case, four combinations of propagators are considered in Figure 4.2. The top left picture illustrates the use of backward Euler as a coarse solver, and M steps of backward Euler as a fine solver for $M = 2, \ldots, 10$. The top curve in that picture corresponds to an exact fine solve. On the top right, we show the same plot with Radau IIA as a fine solver. On the bottom left, we show the trapezoidal rule as fine solver. On the bottom right, we illustrate

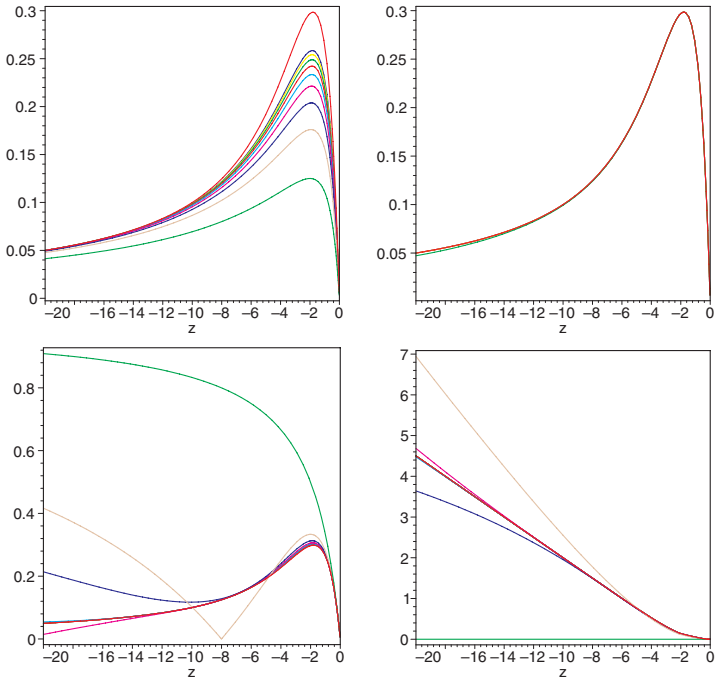

FIG. 4.2. *Value of the fully discrete convergence factor* $\rho(\mathcal{K}(z))$ *with* $z := a\Delta T \in \mathbb{R}_0^-$ *for* 4 *different F-propagator/G-propagator combinations: backward Euler/backward Euler (top left), Radau* IIA*/backward Euler (top right), trapezoidal rule/backward Euler (bottom left), and trapezoidal rule/trapezoidal rule (bottom right).*

the case where the trapezoidal rule is used for both the coarse and fine solve.

Remark 4.12. These experiments indicate that for certain combinations, the continuous result derived in the earlier sections is an upper bound for the fully discretized algorithm. This can be shown rigorously in certain cases, but does not hold in general. For example, for the backward Euler scheme we have that $R(a\Delta T) = 1/(1 - a\Delta T)$. For $a \in \mathbb{R}_0^-$ it can easily be verified that

$$\frac{|R(a\Delta T/M)^M - R(a\Delta T)|}{1 - |R(a\Delta T)|} \leq \frac{|e^{a\Delta T} - R(a\Delta T)|}{1 - |R(a\Delta T)|}.$$

Hence, in that case the convergence rate of the fully discrete parareal operator is bounded by the convergence rate of the parareal solver with an exact solve as F-propagator. This is illustrated in the top left picture of Figure 4.2. This does not, however, continue to hold for all $a \in \mathbb{C}^-$.

Remark 4.13. The stability function of the trapezoidal rule is known to be $R(z) = \frac{1+z/2}{1-z/2}$. From this, it is readily seen that the convergence rate of the combination trapezoidal rule/backward Euler tends to 1 when $z \to -\infty$. For a fixed value of $a\Delta T$, however, the convergence rate tends to the continuous limit with increasing m. This can be observed in the bottom left picture of Figure 4.2. For the trapezoidal rule/trapezoidal rule combination in the bottom right picture of Figure 4.2, the convergence rate actually becomes bigger than 1 for z sufficiently negative.

Remark 4.14. The necessary and sufficient condition for convergence on infinite length time windows as given in Remark 4.11 can be generalized to the fully discrete

case. With (4.19), we see that convergence is guaranteed for $a \in \mathbb{R}_0^-$ if and only if $|R_f(a\Delta T) - R(a\Delta T)| + |R(a\Delta T)| < 1$. This, in turn, is equivalent to the conditions

$$-1 < R_f(a\Delta T) < 1 \quad \text{and} \quad \frac{R_f(a\Delta T) - 1}{2} < R(a\Delta T) < \frac{R_f(a\Delta T) + 1}{2}. \tag{4.20}$$

The first condition is equivalent to a requirement of stability for the F-propagator and will naturally be satisfied.

5. Convergence analysis for partial differential equations. We now use the results derived in section 4 to investigate the performance of the parareal algorithm on PDEs. We consider two model problems, a diffusion problem and an advection problem.

5.1. Convergence analysis for a diffusion equation. For the diffusion case, we consider the heat equation, without loss of generality in one dimension,

$$u_t = u_{xx} \quad \text{in } \Omega = \mathbb{R}, \quad u(0,x) \in L^2(\Omega). \tag{5.1}$$

Using a Fourier transform in space, this equation becomes a system of decoupled ODEs for each Fourier mode ω,

$$\hat{u}_t = -\omega^2 \hat{u}, \tag{5.2}$$

and hence the convergence results for scalar ODEs can be directly applied. In the theorem below, we assume that the heat equation is discretized in time using a one-step method that is stable on the negative real axis (A_0-stable). In the present case, this is a minimal necessary requirement for any time-integrator to produce meaningful results.

THEOREM 5.1. *Let $F(t_{n+1}, t_n, U_n^k)$ be the exact solution at t_{n+1} of (5.1) with $u(t_n) = U_n^k$, and let $G(t_{n+1}, t_n, U_n^k)$ be an A_0-stable one-step method. Then, we have a superlinear bound on the convergence on bounded time intervals,*

$$\max_{1 \le n \le N} ||u(t_n) - U_n^k||_2 \le \frac{\gamma_s^k}{k!} \prod_{j=1}^{k} (N-j) \max_{1 \le n \le N} ||u(t_n) - U_n^0||_2, \tag{5.3}$$

where $||\cdot||_2$ denotes the spectral norm in space and the constant γ_s is universal for each one-step method. Assume that, additionally, the negative real axis belongs to the interior of the stability region of the one-step method and that $|R(-\infty)| = c < 1$. Then, we have a linear bound on the convergence on unbounded time intervals,

$$\sup_{n>0} ||u(t_n) - U_n^k||_2 \le \gamma_l^k \sup_{n>0} ||u(t_n) - U_n^0||_2, \tag{5.4}$$

with a constant γ_l depending only on the one-step method.

Proof. The numerator in the superlinear bound (4.12) when applied to (5.2) is a continuous function of $z := -\omega^2 \Delta T$. It is zero at $z = 0$ and, because of the A_0-stability assumption, it is bounded for $z \to -\infty$ along the real axis. Hence, the numerator is uniformly bounded, and we can define its supremum as

$$\gamma_s := \sup_{z \in \mathbb{R}^-} |e^z - R(z)|. \tag{5.5}$$

This leads to (5.3) by using the Parseval–Plancherel identity.

TABLE 5.1
Superlinear and linear convergence constants.

	Backward Euler	Trapezoidal rule	SDIRK	Radau IIA
γ_s	0.2036321888	1	0.1717941220	0.0634592650
γ_l	0.2984256075	∞	0.2338191487	0.0677592165

The convergence factor in the linear bound (4.15) applied to (5.2) can be bounded as a function of $z := -\omega^2 \Delta T$,

$$\gamma_l := \sup_{z \in \mathbb{R}^-} \frac{|e^z - R(z)|}{1 - |R(z)|}. \tag{5.6}$$

This bound is finite because of the assumptions on the one-step method. This leads to (5.4) using the Parseval–Plancherel identity. ∎

The constants γ_s and γ_l can easily be computed numerically. The values of these constants for the backward Euler method, the trapezoidal rule, the two-stage singly diagonally implicit Runge–Kutta (SDIRK) method, and the three-stage Radau IIA method are presented in Table 5.1. The assumptions on the one-step method specified in the theorem are necessary but not sufficient to guarantee convergence of the parareal algorithm as a solver on unbounded time intervals. That is, the constant γ_l may well be greater than 1. A necessary and sufficient condition for convergence follows from Remark 4.11.

COROLLARY 5.2. *A necessary and sufficient condition for convergence of the parareal algorithm with an exact solve as the F-propagator on unbounded time intervals is given by*

$$\frac{e^z - 1}{2} < R(z) < \frac{e^z + 1}{2} \quad \textit{for} \quad z \in \mathbb{R}^-. \tag{5.7}$$

Remark 5.3. A necessary and sufficient condition for convergence in the fully discrete case is found by replacing e^z by $R_f(z)$ in (5.7) and by adding the condition $|R_f(z)| < 1$.

Remark 5.4. In practice, the heat equation is usually posed on a finite spatial domain, and a spatial discretization is applied. If the spatial discretization is such that the resulting ODE system has only negative real eigenvalues, the bound derived in Theorem 5.1 continues to hold, as it is a uniform bound for all $-\omega^2 \Delta T$.

Remark 5.5. From the results in Table 5.1, we see that the SDIRK and Radau IIA methods have better convergence factors than the lower order methods. This shows that using these higher order methods on the coarse grid is speeding up the convergence of the parareal algorithm. However, this comes at the cost of a more expensive evaluation of the G-propagator.

5.2. Convergence of parareal for an advection equation. Next, we consider a pure advection problem,

$$u_t = u_x \quad \text{in } \Omega = \mathbb{R}, \quad u(0, x) \in L^2(\Omega). \tag{5.8}$$

Using a Fourier transform in space, this equation becomes a decoupled system of differential equations for each Fourier mode ω,

$$\hat{u}_t = -i\omega \hat{u}. \tag{5.9}$$

TABLE 5.2
Superlinear and linear convergence constants for the advection equation.

	Backward Euler	Trapezoidal rule	SDIRK	Radau IIA
α_s	1.224353426	2	1.185652097	1.362526017
α_l	1.632645559	∞	∞	2.231320732

We will again apply the convergence results obtained earlier for scalar ODEs. We assume that the advection equation is discretized in time using a one-step method that is stable on the whole of the imaginary axis. In the present case, this is a necessary requirement for the method to produce bounded results.

THEOREM 5.6. *Let $F(t_{n+1}, t_n, U_n^k)$ be the exact solution at t_{n+1} of (5.9) with $u(t_n) = U_n^k$, and let $G(t_{n+1}, t_n, U_n^k)$ be a one-step method with $|R(z)| \leq 1$ for z on the imaginary axis. Then, the parareal algorithm has a superlinear bound on the convergence rate on bounded time intervals,*

$$\max_{1\leq n\leq N} ||u(t_n) - U_n^k||_2 \leq \frac{\alpha_s^k}{k!} \prod_{j=1}^{k} (N-j) \max_{1\leq n\leq N} ||u(t_n) - U_n^0||_2, \tag{5.10}$$

where the constant α_s depends only on the one-step method.

Proof. The numerator in the superlinear bound (4.12) when applied to (5.2) is a continuous function of $z := -\omega\Delta T$. It is zero at $z = 0$ and, because of the boundedness of $|R(z)|$ along the imaginary axis, it is bounded at infinity. Hence, the numerator is uniformly bounded, and we can define its supremum as

$$\alpha_s := \sup_{z\in\mathbb{R}} |e^{iz} - R(iz)|. \tag{5.11}$$

This leads to (5.10) by using the Parseval–Plancherel identity. ∎

Remark 5.7. Typically, there is no convergence result for (5.8) on unbounded intervals. Formally, we can compute the constant α_l defined as

$$\alpha_l := \sup_{z\in\mathbb{R}} \frac{|e^{iz} - R(iz)|}{1 - |R(iz)|}. \tag{5.12}$$

However, this value will mostly be greater than 1, if not infinite.

As before, the constants α_s and α_l can be computed numerically. The values of these constants for a number of one-step methods are given in Table 5.2.

Remark 5.8. In practice, the advection problem is usually posed on a finite spatial interval, and a spatial discretization is applied. If the spatial discretization is such that the resulting ODE system has only purely imaginary eigenvalues, the bound derived in Theorem 5.6 continues to hold, as it is a uniform bound for all $i\omega\Delta T$.

Remark 5.9. The bound in Theorem 5.6 needs to become small for k less than N for the algorithm to be interesting. To see if this is possible, we study the function

$$g(\alpha, N, k) := \frac{\alpha^k}{k!} \prod_{j=1}^{k} (N-j) = \frac{(-\alpha)^k\, \Gamma(k+1-N)}{k!\, \Gamma(1-N)},$$

where Γ denotes the Gamma-function. In Figure 5.1, we show, for two values of α and various values of N, how the function $g(\alpha, N, k)$ becomes small as k grows from 1 up to N. This shows that even though there is superlinear convergence in the advection

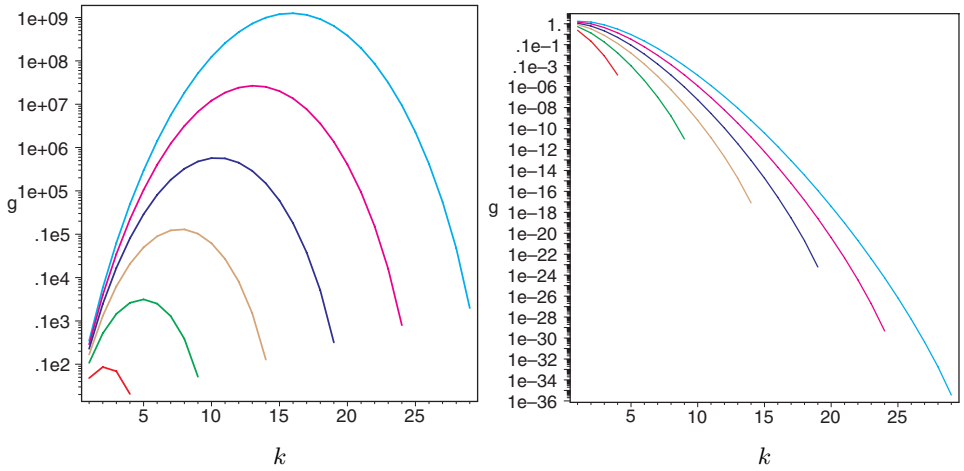

FIG. 5.1. *Behavior of the superlinear convergence factor for* $\alpha = 1.2$ *on the left and* $\alpha = 0.06$ *on the right, as a function of the iteration number* k*, for the values of* $N = 5, 10, 15, 20, 25, 30$.

case, the constants α_s of the various methods do not permit a speedup of the parareal algorithm. In the diffusion case, however, there is substantial speedup, as the constant for the Radau IIA method shows.

Remark 5.10. If one considers the pure advection case on a sufficiently bounded domain with a Dirichlet boundary condition, it is the boundary condition which largely determines the solution. In such a case, our numerical experiments show that the parareal algorithm converges linearly.

6. Numerical experiments. In order to illustrate the theoretical results, we first show some numerical experiments for the scalar model problem (4.1) with $f = 0$, $a = -1$, $u_0 = 1$. These results are graphically displayed in Figure 6.1. The backward Euler method is chosen for both the coarse approximation and the fine approximation, with time step ΔT and $\Delta T/M$, respectively. In the top left, top right, and bottom left pictures, we show the convergence results obtained for $T = 1$, $T = 10$, and $T = 50$, respectively, using $N = 10$ and $M = 20$ in each case. In each picture the L^∞-norm of the true error is plotted as a function of the iteration index k, together with the bounds as presented in Proposition 4.1 (earlier bound), in Corollary 4.6 (superlinear bound) and in Theorem 4.9 (linear bound).

One can clearly see that parareal has two different convergence regimes: for $T = 1$, the algorithm converges superlinearly, and the superlinear bound is quite sharp; for $T = 10$, the convergence rate is initially linear, and then a transition occurs to the superlinear convergence regime. Finally, for $T = 50$, the algorithm is in the linear convergence regime and the linear bound from Theorem 4.9 is quite sharp. Note also that the earlier bound from Proposition 4.1 indicates stagnation for $T = 10$, since $\Delta T = 1$, and divergence for $T = 50$, since then $\Delta T > 1$. The parareal algorithm does, however, also converge for $\Delta T \geq 1$.

From our analysis, one can also see that the convergence factor is small for small ΔT, then becomes bigger and reaches a maximum, and finally becomes smaller again for large ΔT; see Figure 4.2, for example. This effect is demonstrated in the bottom right picture of Figure 6.1 for the same model problem as before. Again, the backward Euler discretization is used for both the coarse and fine solver. For each case the same F-propagator is used, defined by $\Delta t = 1/20$.

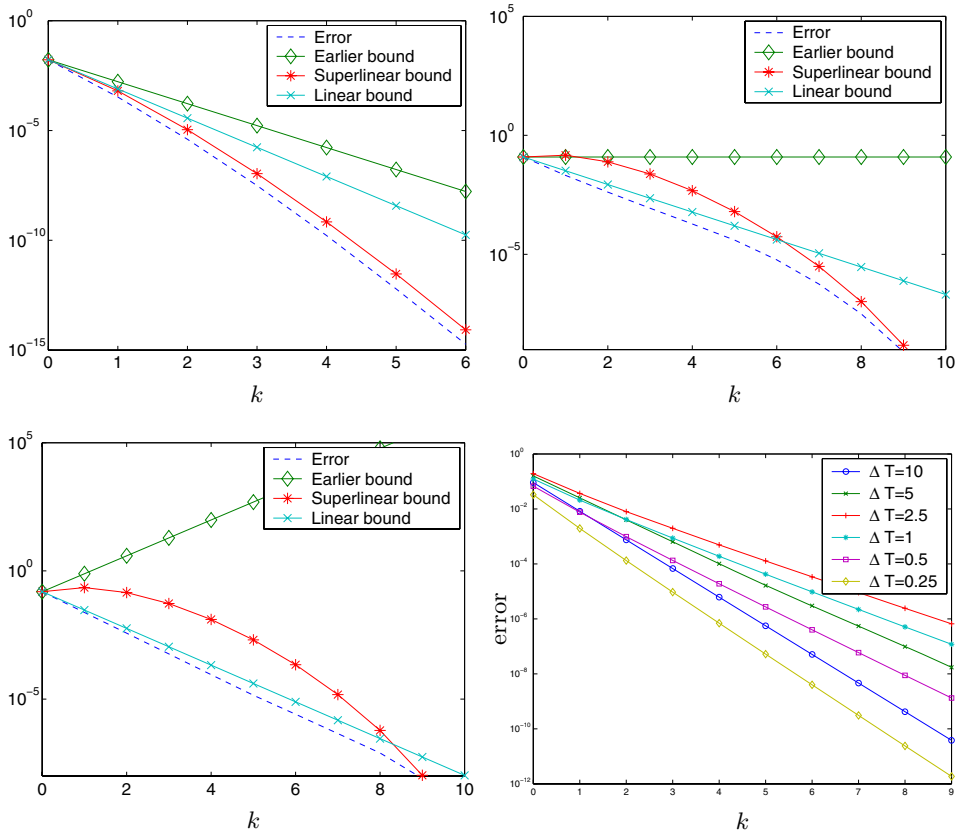

FIG. 6.1. *Parareal convergence for* (4.1): L^∞*-norm of the error as a function of the iteration index* k *on a short (top left), medium (top right), and long (bottom left) time interval; dependence of the convergence speed on* ΔT *(bottom right).*

We now turn our attention to the PDE case and show some experiments for the heat equation $u_t = u_{xx} + f$ in $(0, L) \times (0, T]$ with homogeneous initial and boundary conditions and with $f = x^4(1-x) + t^2$. We use backward Euler for the integration in time and choose the domain size L such that the problem contains a frequency where the linear convergence bound for backward Euler is attained; see Figure 4.2, top left. With $\Delta T = 1/2$ and $M = 10$ we obtain the results shown in Figure 6.2. On the left, results are shown for $T = 4$, where the algorithm with $\Delta T = 1/2$ will converge in 8 steps. One can see that this is clearly the case. Before that, the algorithm is in the superlinear convergence regime, as predicted by the superlinear bound. Note that the latter bound indicates zero as the error at the eighth step and thus cannot be plotted on the logarithmic scale. On the right, the error is shown for $T = 8$, and the algorithm is now in the linear convergence regime.

We finally show four experiments for the advection equation $u_t = u_x + f$ in $(0, L) \times (0, T]$. To emulate the situation of an unbounded domain, we impose in the first two experiments periodic boundary conditions. We use as initial condition $u(x, 0) = e^{-20(x-\frac{L}{2})^2}$ and the forcing function $f \equiv 0$. With $\Delta T = 1/2$, $T = 4$, and $M = 30$, we obtain the results in Figure 6.3 on the left.

In the top left panel of Figure 6.3, we have chosen the domain size L for the worst-case behavior of our superlinear bound, and in the bottom left panel of Figure 6.3 we

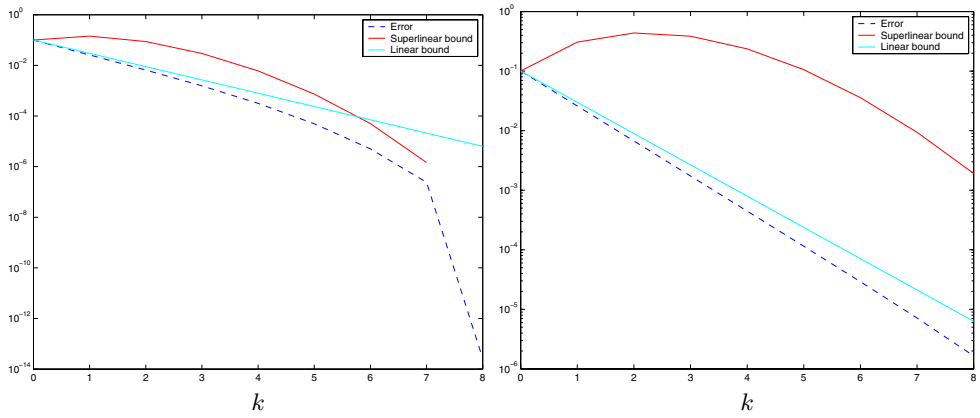

FIG. 6.2. *Error in the L^∞-norm in time and L^2-norm in space for the parareal algorithm applied to the heat equation, on a short (left) and long (right) interval.*

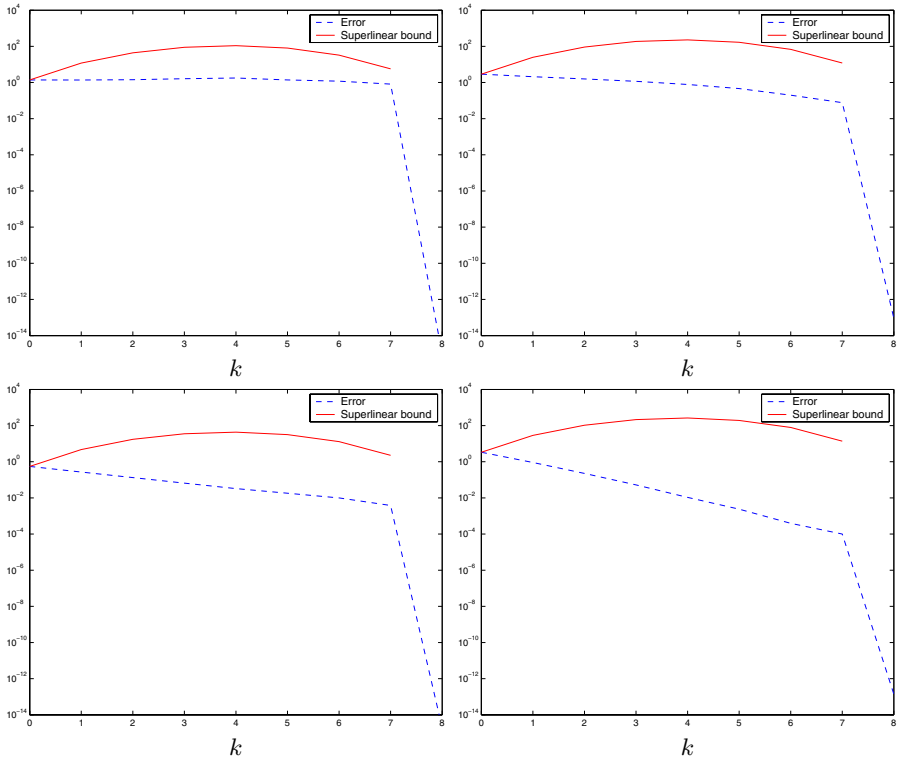

FIG. 6.3. *Error in the L^∞-norm in time and L^2-norm in space for the parareal algorithm applied to the advection equation, with periodic boundary conditions in the graphs on the left, and a Dirichlet condition in the graphs on the right. In the top row the domain size L is chosen for worst-case behavior, and in the bottom row $L = \frac{1}{2}$.*

set $L = \frac{1}{2}$. In the former case, the algorithm stagnates and converges only in step 8, when the fine solution has been computed through sequentially. In the latter case we observe slow, linear convergence. We now repeat the same two experiments but with a boundary condition $u(L,t) = \sin(5t)$. As one can see in Figure 6.3 on the right,

the boundary condition improves the behavior of the parareal algorithm applied to the advection equation, as discussed in Remark 5.10. This is expected based on the information propagation along characteristics.

7. Concluding remarks. We have pointed out how the parareal algorithm relates to some earlier algorithms that have appeared in the literature on time-parallel time-integration methods. We showed that parareal can be seen as a variant of the multiple shooting method, with a Jacobian matrix that is computed approximately by finite differences on a coarse mesh. As a Newton-type method for solving a nonlinear system of equations, its superlinear convergence on bounded time intervals is not unexpected. This convergence behavior was studied in detail in section 4.3.

We also showed that the algorithm can be rewritten as a two-level multigrid-in-time method of full approximation storage type. It is characterized by a possibly aggressive coarsening in time: the coarsening ratio M is not restricted to a small integer for the method to be effective. The algorithm uses a rather unusual smoother which corresponds to a single phase in a two-color relaxation scheme, combined with most simple restriction and prolongation operators. As a multigrid method for general time-dependent problems, parareal can be interpreted as a type of parabolic multigrid method, a class of solvers for parabolic PDEs that includes the space-time multigrid, time-parallel multigrid, and multigrid waveform relaxation methods. All of these can be written in the framework of formula (3.2) and differ only in the details of the multigrid components. With those methods, parareal has in common the linear convergence rate on long time intervals, as studied in section 4.4. In contrast to those methods, which are only effective for parabolic PDEs, parareal is applicable to a much wider range of problems.

Recently new parareal variants have appeared, e.g., genuinely *multi*grid versions, with more than two grid levels and methods that combine coarsening in time with coarsening in space. Promising results with such methods have already been reported in the literature, e.g., in [9, 11]. Possibly, insights from the present paper may prove to be useful in the study of the theoretical properties of these new algorithms; perhaps, they may direct us to still more general or effective time-parallel time-integration methods. This will be the subject of further investigation.

REFERENCES

[1] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah, *Parallel-in-time molecular-dynamics simulations*, Phys. Rev. E, 66 (2002), pp. 057706–1–4.

[2] G. Bal, *On the convergence and the stability of the parareal algorithm to solve partial differential equations*, in Proceedings of the 15th International Domain Decomposition Conference, Lect. Notes Comput. Sci. Eng. 40, R. Kornhuber, R. H. W. Hoppe, J. Péeriaux, O. Pironneau, O. B. Widlund, and J. Xu, eds., Springer, Berlin, 2003, pp. 426–432.

[3] G. Bal and Y. Maday, *A "parareal" time discretization for non-linear PDE's with application to the pricing of an American put*, in Recent Developments in Domain Decomposition Methods, Lect. Notes Comput. Sci. Eng. 23, Springer, Berlin, 2002, pp. 189–202.

[4] A. Bellen and M. Zennaro, *Parallel algorithms for initial value problems for difference and differential equations*, J. Comput. Appl. Math., 25 (1989), pp. 341–350.

[5] A. Brandt, *Multi-level adaptive solutions to boundary value problems*, Math. Comp., 31 (1977), pp. 333–390.

[6] P. Chartier and B. Philippe, *A parallel shooting technique for solving dissipative ODEs*, Computing, 51 (1993), pp. 209–236.

[7] A. Deshpande, S. Malhotra, C. C. Douglas, and M. H. Schultz, *A rigorous analysis of time domain parallelism*, Parallel Algorithms Appl., 6 (1995), pp. 53–62.

[8] C. Farhat and M. Chandesris, *Time-decomposed parallel time-integrators: Theory and feasibility studies for fluid, structure, and fluid-structure applications*, Internat. J. Numer. Methods Engrg., 58 (2003), pp. 1397–1434.

[9] P. F. Fischer, F. Hecht, and Y. Maday, *A parareal in time semi-implicit approximation of the Navier-Stokes equations*, in Proceedings of the 15th International Domain Decomposition Conference, Lect. Notes Comput. Sci. Eng. 40, R. Kornhuber, R. H. W. Hoppe, J. Péeriaux, O. Pironneau, O. B. Widlund, and J. Xu, eds., Springer, Berlin, 2003, pp. 433–440.

[10] I. Garrido, M. S. Espedal, and G. E. Fladmark, *A convergence algorithm for time parallelization applied to reservoir simulation*, in Proceedings of the 15th International Domain Decomposition Conference, Lect. Notes Comput. Sci. Eng. 40, R. Kornhuber, R. H. W. Hoppe, J. Péeriaux, O. Pironneau, O. B. Widlund, and J. Xu, eds., Springer, Berlin, 2003, pp. 469–476.

[11] I. Garrido, B. Lee, G. E. Fladmark, and M. E. Espedal, *Convergent iterative schemes for time parallelization*, Math. Comp., 26 (2006), pp. 1403–1428.

[12] W. Hackbusch, *Parabolic multigrid methods*, in Computing Methods in Applied Sciences and Engineering, VI, R. Glowinski and J.-L. Lions, eds., North–Holland, Amsterdam 1984, pp. 189–197.

[13] G. Horton, *The time-parallel multigrid method*, Comm. Appl. Numer. Methods, 8 (1992), pp. 585–595.

[14] G. Horton and S. Vandewalle, *A space-time multigrid method for parabolic partial differential equations*, SIAM J. Sci. Comput., 16 (1995), pp. 848–864.

[15] G. Horton, S. Vandewalle, and P. Worley, *An algorithm with polylog parallel complexity for solving parabolic partial differential equations*, SIAM J. Sci. Comput., 16 (1995), pp. 531–541.

[16] H. B. Keller, *Numerical Methods for Two-Point Boundary Value Problems*, Blaisdell, Waltham, MA, 1968.

[17] M. Kiehl, *Parallel multiple shooting for the solution of initial value problems*, Parallel Comput., 20 (1994), pp. 275–295.

[18] R. Kornhuber, R. H. W. Hoppe, J. Péeriaux, O. Pironneau, O. B. Widlund, and J. Xu, eds., *Proceedings of the 15th International Domain Decomposition Conference*, Lect. Notes Comput. Sci. Eng. 40, Springer, Berlin, 2003.

[19] J.-L. Lions, Y. Maday, and G. Turinici, *A "parareal" in time discretization of PDE's*, C. R. Acad. Sci. Paris Sér. I Math., 332 (2001), pp. 661–668.

[20] C. Lubich and A. Ostermann, *Multi-grid dynamic iteration for parabolic equations*, BIT, 27 (1987), pp. 216–234.

[21] Y. Maday and G. Turinici, *A parareal in time procedure for the control of partial differential equations*, C. R. Math. Acad. Sci. Paris. Sér. I Math., 335 (2002), pp. 387–391.

[22] Y. Maday and G. Turinici, *The parareal in time iterative solver: A further direction to parallel implementation*, in Proceedings of the 15th International Domain Decomposition Conference, Lect. Notes Comput. Sci. Eng. 40, R. Kornhuber, R. H. W. Hoppe, J. Péeriaux, O. Pironneau, O. B. Widlund, and J. Xu, eds., Springer, Berlin, 2003, pp. 441–448.

[23] W. L. Miranker and W. Liniger, *Parallel methods for the numerical integrating of ordinary differential equations*, Math. Comp., 91 (1967), pp. 303–320.

[24] J. Nievergelt, *Parallel methods for integrating ordinary differential equations*, Comm. ACM, 7 (1964), pp. 731–733.

[25] L. Reichel and L. N. Trefethen, *Eigenvalues and pseudo-eigenvalues of Toeplitz matrices*, Linear Algebra Appl., 162/164 (1992), pp. 153–185.

[26] J. H. Saltz and V. K. Naik, *Towards developing robust algorithms for solving partial differential equations on MIMD machines*, Parallel Comput., 6 (1988), pp. 19–44.

[27] L. F. Shampine and H. A. Watts, *Block implicit one-step methods*, Math. Comp., 23 (1969), pp. 731–740.

[28] R. D. Skeel, *A theoretical framework for proving accuracy results for deferred corrections*, SIAM J. Numer. Anal., 19 (1982), pp. 171–96.

[29] G. A. Staff and E. M. Rønquist, *Stability of the parareal algorithm*, in Proceedings of the 15th International Domain Decomposition Conference, Lect. Notes Comput. Sci. Eng. 40, R. Kornhuber, R. H. W. Hoppe, J. Péeriaux, O. Pironneau, O. B. Widlund, and J. Xu, eds., Springer, Berlin, 2003, pp. 449–456.

[30] K. Stueben, *A review of algebraic multigrid*, J. Comput. Appl. Math., 198 (2001), pp. 281–309.

[31] S. Vandewalle and E. V. de Velde, *Space-time concurrent multigrid waveform relaxation*, Ann. Numer. Math., 1 (1994), pp. 347–360.

[32] S. Vandewalle and R. Piessens, *Efficient parallel algorithms for solving initial-boundary value and time-periodic parabolic partial differential equations*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1330–1346.

[33] D. E. Womble, *A time-stepping algorithm for parallel computers*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 824–837.