Noname manuscript No.
(will be inserted by the editor)

Multigrid Interpretations of the Parareal Algorithm Leading to an Overlapping Variant and MGRIT

Martin J. Gander · Felix Kwok · Hui Zhang

Received: date / Accepted: date

Abstract The parareal algorithm is by construction a two level method, and there are several ways to interpret the parareal algorithm to obtain multilevel versions. We first review the three main interpretations of the parareal algorithm as a two-level method, a direct one, one based on geometric multigrid and one based on algebraic multigrid. The algebraic multigrid interpretation leads to the MGRIT algorithm, when using instead of only an F-smoother, a so called FCF-smoother. We show that this can be interpreted at the level of the parareal algorithm as generous overlap in time. This allows us to generalize the method to even larger overlap, corresponding in MGRIT to $F(CF)^\nu$-smoothing, $\nu \geq 1$, and we prove two new convergence results for such algorithms in the fully non-linear setting: convergence in a finite number of steps, becoming smaller when ν increases, and a general superlinear convergence estimate for this generalized version of MGRIT. We illustrate our results with numerical experiments, both for linear and non-linear systems of ordinary and partial differential equations. Our results show that overlap only sometimes leads to faster algorithms.

HZ thanks ZJNSF (LY17A010014). FK thanks RGC (ECS22300115) and NSFC (YS11501483).

M. J. Gander
Section de mathématiques, Université de Genève
E-mail: martin.gander@unige.ch

F. Kwok
Department of Mathematics, Hong Kong Baptist University
E-mail: felix_kwok@hkbu.edu.hk

H. Zhang (corresponding author)
Key Laboratory of Oceanographic Big Data Mining & Application of Zhejiang Province, Zhejiang Ocean University
E-mail: huiz@zjou.edu.cn

Keywords Parareal Algorithm with Overlap · MGRIT · Geometric and Algebraic Multigrid in Time

1 Introduction

Time parallel time integration methods have a long history, see [7] and references therein. The parareal algorithm, introduced by Lions, Maday and Turinici in [17], sparked renewed interest in such methods, and with the advent of so massively parallel computers that not all processors can be used effectively by space parallelization only, substantial research efforts have gone into deriving and studying such methods. The parareal algorithm is a two level method, very much in the spirit of domain decomposition methods, but instead of decomposing in space, it decomposes the time direction. Its convergence behavior is well understood, see for example [16] for linear problems and [8, 9] for non-linear problems.

The parareal algorithm for the simple evolution problem

$$\frac{du}{dt} = f(u) \quad \text{in } (0,T), \quad u(0) = u_0, \tag{1}$$

needs two approximate solution operators: a fine solver $F(T_2, T_1, U_1)$ which solves the differential equation (1) in the time interval (T_1, T_2) starting with U_1 at time T_1 and returning a very good approximation of the solution at time T_2, and a cheap, coarse approximation $G(T_2, T_1, U_1)$ which analogously produces a corresponding coarse approximation at time T_2. From an initial approximation U_n^0 at the coarse time points $0 = T_0 < T_1 < \ldots < T_N = T$, which can for example be obtained by sequentially using the coarse solver,

$$U_{n+1}^0 = G(T_{n+1}, T_n, U_n^0), \quad U_0^0 = u_0, \tag{2}$$

the parareal algorithm computes for iteration index $k = 0, 1, \ldots$ with $U_0^{k+1} = u_0$ a better and better approximation using the iteration

$$U_{n+1}^{k+1} = F(T_{n+1}, T_n, U_n^k) + G(T_{n+1}, T_n, U_n^{k+1}) - G(T_{n+1}, T_n, U_n^k). \quad (3)$$

We see in (3) that given an approximation U_n^k for $n = 0, 1, \ldots, N$, one can perform all expensive computations with the fine solver F in parallel, before sequentially updating the solution approximations U_n^{k+1} using the coarse solver G, which often permits to solve the evolution problem faster with a parallel computer than just time stepping sequentially with the fine solver F.

The most natural way to understand the parareal algorithm is to see it as a multiple shooting method for solving initial value problems. Denoting the exact solution trajectory on the time interval (T_n, T_{n+1}) with initial value v by $u_n(t, v)$, and imposing continuity of the solution trajectory at the time points T_n, we form a nonlinear system $\mathcal{F}(\boldsymbol{U}) = 0$ for $\boldsymbol{U} := (U_n)_{n=1}^N$ as

$$U_{n+1} - u_n(T_{n+1}, U_n) = 0, \quad n = 0, .., N-1.$$

To solve this system using Newton's method, we first write down the Jacobian matrix

$$J(\boldsymbol{U}) := \begin{pmatrix} I & & & \\ -\frac{\partial u_1}{\partial U_1}(T_2, U_1) & I & & \\ & \ddots & \ddots & \\ & & -\frac{\partial u_{N-1}}{\partial U_{N-1}}(T_N, U_{N-1}) & I \end{pmatrix},$$

then insert J into the Newton iteration scheme

$$\boldsymbol{U}^{k+1} = \boldsymbol{U}^k - \left[J(\boldsymbol{U}^k)\right]^{-1} \mathcal{F}(\boldsymbol{U}^k),$$

and multipliy by J to find the componentwise form

$$U_{n+1}^{k+1} = u_n(T_{n+1}, U_n^k) + \frac{\partial u_n}{\partial U_n}(T_{n+1}, U_n^k)(U_n^{k+1} - U_n^k). \quad (4)$$

We see that the parareal updating formula (3) is an approximation to the Newton step (4), using the approximate solution provided by the fine solver F instead of the exact one, and approximating the Jacobian term by a difference using the coarse solver G,

$$\frac{\partial u_n}{\partial U_n}(T_{n+1}, U_n^k)(U_n^{k+1} - U_n^k) \approx G(T_{n+1}, T_n, U_n^{k+1}) - G(T_{n+1}, T_n, U_n^k),$$

see [8,16] for more details.

If one is interested in generalizing two level methods to multilevel methods, it is useful to obtain a multigrid interpretation of the two level algorithm in the sense of smoothing, restriction, prolongation and coarsening, and for the parareal algorithm, there are several ways to do this. We show in Section 2 three different ways to interpret the parareal algorithm as a two level method to extend it to many levels. We then show in Section 3 that the third such interpretation, which led to MGRIT, can be interpreted in turn as an overlapping variant of the parareal algorithm, and we generalize these methods to arbitrary overlap sizes. We then prove two new convergence results for such algorithms in the fully non-linear setting: convergence in a finite number of steps, where the number of steps becomes smaller when ν increases, and a general superlinear convergence estimate for this generalized version of MGRIT. We illustrate our results using numerical experiments in Section 4.

2 Multilevel variants of parareal

We explain now how to interpret the parareal algorithm as a two-level method and generalize it to a multilevel variant in three different ways. The first one is just using parareal terminology. The second one is based on geometric multigrid, and the third one on algebraic multigrid. To simplify the exposition, we use the scalar linear model problem

$$\frac{du}{dt} = au \quad \text{in } (0, T), \quad u(0) = u_0, \quad (5)$$

but all our results also hold for the more general case of systems.

2.1 A multilevel parareal algorithm

Given N the number of coarse time intervals and M the number of fine time steps in every coarse time interval, we discretize (5) in time using the time grid $0 = t_0 < t_1 < t_2 < \ldots < t_{MN} = T$, $t_m - t_{m-1} = \Delta t$, and while our results hold for arbitrary one step methods, we use here first for simplicity just the forward Euler method to integrate, which leads to the lower bidiagonal linear system

$$A\boldsymbol{u} := \begin{pmatrix} 1 & & & \\ -\phi & 1 & & \\ & \ddots & \ddots & \\ & & -\phi & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{MN} \end{pmatrix} = \begin{pmatrix} u_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} =: \boldsymbol{f}, \quad (6)$$

where $\phi := 1 + a\Delta t$. Forward substitution to solve (6) corresponds to solving the underlying evolution problem (5) by time stepping. If we eliminate every second

unknown from the lower bidiagonal system (6), we obtain the smaller system

$$\begin{pmatrix} 1 & & & \\ -\phi^2 & 1 & & \\ & \ddots & \ddots & \\ & & -\phi^2 & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_2 \\ \vdots \\ u_{MN} \end{pmatrix} = \begin{pmatrix} u_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Similarly, we can eliminate more interior unknowns, and keep only every M-th one, which leads to

$$\tilde{A}\tilde{\boldsymbol{U}} := \begin{pmatrix} 1 & & & \\ -\tilde{F} & 1 & & \\ & \ddots & \ddots & \\ & & -\tilde{F} & 1 \end{pmatrix} \begin{pmatrix} \tilde{U}_0 \\ \tilde{U}_1 \\ \vdots \\ \tilde{U}_N \end{pmatrix} = \begin{pmatrix} u_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} =: \tilde{\boldsymbol{f}}, \quad (7)$$

where we denoted by $\tilde{F} := (1 + a\Delta t)^M$. Solving the reduced system (7), we obtain by construction at the coarse nodes $T_n := nM\Delta t$ exactly the same solution as when solving the underlying fine system, i.e. $\tilde{U}_n = u_{nM}$. The matrix $\tilde{A}$ can thus be regarded as being the Schur complement matrix when eliminating all interior fine unknowns in the coarse time intervals.

An approximation of the system matrix in (7) can be obtained by approximating the M fine integration steps in $\tilde{F}$ by one coarse integration step, $\tilde{G} := (1 + a\Delta T)$, with $\Delta T := M\Delta t$,

$$\tilde{A} := \begin{pmatrix} 1 & & & \\ -\tilde{F} & 1 & & \\ & \ddots & \ddots & \\ & & -\tilde{F} & 1 \end{pmatrix} \approx \begin{pmatrix} 1 & & & \\ -\tilde{G} & 1 & & \\ & \ddots & \ddots & \\ & & -\tilde{G} & 1 \end{pmatrix} =: \tilde{M} \quad (8)$$

One can then solve the reduced system (7) by the preconditioned stationary iteration

$$\tilde{\boldsymbol{U}}^{k+1} = \tilde{\boldsymbol{U}}^k + \tilde{M}^{-1}(\boldsymbol{f} - \tilde{A}\tilde{\boldsymbol{U}}^k). \quad (9)$$

We now show that this iteration is precisely the parareal algorithm (3), see also [20].

Theorem 1 *The stationary iteration (9) coincides with the parareal algorithm (3) applied to the linear problem (5) using Forward Euler as the time integrator, i.e.* $\tilde{U}_n^k = U_n^k$ *for* $k = 1, 2, \ldots$ *and* $n = 0, 1, \ldots, N$*, provided initially we have* $\tilde{U}_n^0 = U_n^0$ *for* $n = 0, 1, \ldots, N$.

Proof The preconditioned stationary iteration (9) computes

$$\begin{pmatrix} 1 & & & \\ -\tilde{G} & 1 & & \\ & \ddots & \ddots & \\ & & -\tilde{G} & 1 \end{pmatrix} \left(\begin{pmatrix} \tilde{U}_0^{k+1} \\ \tilde{U}_1^{k+1} \\ \vdots \\ \tilde{U}_N^{k+1} \end{pmatrix} - \begin{pmatrix} \tilde{U}_0^k \\ \tilde{U}_1^k \\ \vdots \\ \tilde{U}_N^k \end{pmatrix} \right) = \begin{pmatrix} u_0 - \tilde{U}_0^k \\ \tilde{F}\tilde{U}_0^k - \tilde{U}_1^k \\ \vdots \\ \tilde{F}\tilde{U}_{N-1}^k - \tilde{U}_N^k \end{pmatrix}.$$

The n-th line in this iteration reads

$$-\tilde{G}\tilde{U}_{n-1}^{k+1} + \tilde{G}\tilde{U}_{n-1}^k + \tilde{U}_n^{k+1} - \tilde{U}_n^k = \tilde{F}\tilde{U}_{n-1}^k - \tilde{U}_n^k,$$

and we obtain after simplification

$$\tilde{U}_n^{k+1} = \tilde{F}\tilde{U}_{n-1}^k + \tilde{G}\tilde{U}_{n-1}^{k+1} - \tilde{G}\tilde{U}_{n-1}^k. \quad (10)$$

Now applying the parareal algorithm to the linear problem (5) using forward Euler, we get for the parareal fine integrator $F(T_{n+1}, T_n, v) := (1+a\Delta t)^M v \equiv \tilde{F}v$, and for the coarse integrator $G(T_{n+1}, T_n, v) := (1 + a\Delta T)v \equiv \tilde{G}v$, and thus the updating formula (10) coincides with the parareal updating formula in (3), and the result follows by induction.

Note that this result also holds for any other integrator we could have used, since we never used the precise form of the Forward Euler integrator, but only the dependence of the solutions between two consective time points. For example, if we use the Backward Euler integrator instead, we simply need to let $\phi = (1 - a\Delta t)^{-1}$, $\tilde{F} = (1 - a\Delta t)^{-M}$, $\tilde{G} = (1 - a\Delta T)^{-1}$; see also [20] and [5, 6]. It is now straightforward to generalize the parareal algorithm to a multilevel method, since the coarse operator $\tilde{M}$ which uses $\tilde{G}$ in (8) is exactly of the same form as the underlying system matrix A, and thus can be approximately inverted using the parareal algorithm again with a stationary iteration of the form (9).

2.2 Parareal as a geometric multigrid method

For solving approximately the linear system

$$A\boldsymbol{u} = \boldsymbol{f}, \quad (11)$$

a geometric two grid method would, starting with the initial guess $\boldsymbol{u}^0$, compute for $k = 0, 1, 2, \ldots$

$$\tilde{\boldsymbol{u}}^k := \text{Smooth}(A, \boldsymbol{f}, \boldsymbol{u}^k); \quad (12)$$

$$\boldsymbol{e} := A_c^{-1} R(\boldsymbol{f} - A\tilde{\boldsymbol{u}}^k); \quad (13)$$

$$\boldsymbol{u}^{k+1} := \tilde{\boldsymbol{u}}^k + P\boldsymbol{e}; \quad (14)$$

where P is a prolongation operator often obtained by interpolation, R is a restriction operator, often $R := P^T$, and A_c is a coarse approximation of the problem, which is either a discretization on a coarse grid, or a Galerkin approximation $A_c := RAP$. A geometric multigrid method would be obtained by recursively coarsening the coarse time grid and applying this algorithm when solving the coarse problem involving the matrix A_c. Note that we did not introduce the post-smoothing step here often used in geometric multigrid, since we will not need it. To identify the parareal algorithm with this standard form of the geometric multigrid method, we also introduce a block Jacobi splitting $A = \tilde{M}_J - \tilde{N}_J$, where $\tilde{M}_J$ is a block diagonal matrix with diagonal blocks of size $M \times M$, except for the first block which is one bigger because of the initial condition of the problem. The following proposition reveals precisely which smoother, restriction R and prolongation P, and coarse operator A_c needs to be used in (12) to obtain the parareal algorithm.

Theorem 2 *If one uses only one presmoothing step with the modified block Jacobi smoother*

$$\boldsymbol{u}^k = \boldsymbol{u}^{k-1} + E\tilde{M}_J^{-1}(\boldsymbol{f} - A\boldsymbol{u}^{k-1}), \tag{15}$$

where E is the identity, except with zeros at the coarse nodes, and uses injection (zero extension) for P and for the restriction $R := P^T$, and for the coarse matrix $A_c = \tilde{M}$ from (8), then the classical two grid algorithm (12-14) produces the parareal iterates at the coarse nodes, i.e. $U_n^k = u_{nM}^k$, provided one starts with an initial approximation $\boldsymbol{u}^0$ that satisfies $U_n^0 = u_{nM}^0$.

Proof The proof is by induction on k, and the statement holds trivially for $k = 0$ by assumption. We thus assume that the statement holds for k, i.e $u_{nM}^k = U_n^k$ for $n = 0, 1, \ldots, N$, and prove that it then also holds for $k+1$. Starting with one smoothing step from (12), using the modified block Jacobi smoother (15), $\tilde{\boldsymbol{u}}^k$ contains now the fine solutions starting at each initial condition U_n^k, $\tilde{u}_{nM+j}^k = (1 + a\Delta t)^j U_n^k$, $j = 0, 1, \ldots, M-1$, except for the coarse variables which have not changed because of E, and we still have $\tilde{\boldsymbol{u}}_{nM}^k = U_n^k$ for $n = 0, 1, \ldots, N$. Now from (13) of the geometric two grid method, we obtain with $A_c = \tilde{M}$

$$\tilde{M}\boldsymbol{e} = R(\boldsymbol{f} - A\tilde{\boldsymbol{u}}^k).$$

Looking at any line $n > 0$ of this equality, we find using that the restriction R is the transpose of injection, the definition of the matrix A in (6) and $\tilde{M}$ in (8), and also that the right hand side $\boldsymbol{f}$ equals zero except in the first component

$$\boldsymbol{e}_n - \tilde{G}(\boldsymbol{e}_{n-1}) = -\tilde{u}_{nM}^k + (1 + a\Delta t)(\tilde{u}_{nM-1}^k). \tag{16}$$

Now from the Jacobi smoother, we have

$$\tilde{u}_{nM-1}^k = (1 + a\Delta t)^{M-1} u_{(n-1)M}^k = (1 + a\Delta t)^{M-1} U_{n-1}^k,$$

where we used the induction hypothesis, and replacing also the first occurrence of $\tilde{u}_{nM}^k$ by U_n^k on the right in (16) because the smoother leaves the coarse values unchanged leads to

$$\boldsymbol{e}_n = -U_n^k + \tilde{F}(U_{n-1}^k) + \tilde{G}(\boldsymbol{e}_{n-1}), \tag{17}$$

where we used the definition of the fine solver $\tilde{F}$. Now performing the last step (14), since the prolongation is just injection, and using again that $\tilde{u}_{nM}^k = U_n^k$, we get on the lines nM

$$u_{nM}^{k+1} = \tilde{F}(U_{n-1}^k) + \tilde{G}(\boldsymbol{e}_{n-1}). \tag{18}$$

We next note that (17) at step $n-1$ can be written as

$$\boldsymbol{e}_{n-1} = -U_{n-1}^k + \tilde{F}(U_{n-2}^k) + \tilde{G}(\boldsymbol{e}_{n-2}) = -U_{n-1}^k + u_{(n-1)M}^{k+1}, \tag{19}$$

where we used (18) for the last step, and inserting this into (18) and using linearity gives

$$u_{nM}^{k+1} = \tilde{F}(U_{n-1}^k) + \tilde{G}(u_{(n-1)M}^{k+1}) - \tilde{G}(U_{n-1}^k),$$

which concludes the proof by induction, since this is the recurrence formula for the parareal algorithm in (3), and $u_0^{k+1} = U_0^{k+1} = u_0$.

Theorem 2 also holds in the non-linear context when using the full approximation scheme, see [16]. However, the somewhat special block Jacobi smoother (15) is only modifying the fine nodes, and leaves the coarse nodes invariant. It is therefore not a convergent smoother, and one can only use one smoothing step, since the second step would not produce any further modification. In the context of algebraic multigrid, where the fine nodes are called F-nodes, such smoothers are called F-relaxation [22, page 423], see also [2], in the context of an approximate direct solver. The idea of the corresponding direct solver is to eliminate all fine F-nodes forming a Schur complement, then solving the Schur complement system for the coarse nodes, and finally filling in the fine nodes by solving for them based on the coarse values just obtained [3, 21]. Approximating this process leads then to Schur complement iterative methods [21], and AMG, see the early variants MGR [18] and AMG01 [22].

2.3 Parareal as an algebraic multigrid method

It is thus natural to try to interpret the parareal algorithm in the context of algebraic multigrid (AMG), see [6], where the authors say:

> "Our algorithm is based on interpreting the parareal time integration method as a two level reduction scheme, and developing a multilevel algorithm from this viewpoint",

see also [5]. In AMG, one does not have access to the underlying grid information of the problem, and therefore needs to algebraically define which variables belong to the fine grid and which variables belong to the coarse grid. Following [19, page 86], one thus partitions the unknowns into fine, so called F-nodes, and coarse, so called C-nodes[1]:

> "Thus, the set of variables on level h can be split into two disjoint subsets: the first one contains the variables also represented in the coarser level (*C-variables*), and the second one is just the complementary set (*F-variables*)."

One then reorders the system matrix from the generic linear system (11) to obtain[2]

$$A\boldsymbol{u} = \begin{bmatrix} A_{\mathrm{ff}} & A_{\mathrm{fc}} \\ A_{\mathrm{cf}} & A_{\mathrm{cc}} \end{bmatrix} \begin{pmatrix} \boldsymbol{u}_f \\ \boldsymbol{u}_c \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_f \\ \boldsymbol{f}_c \end{pmatrix} = \boldsymbol{f}. \tag{20}$$

A block-LU factorization is given by

$$\begin{bmatrix} A_{\mathrm{ff}} & A_{\mathrm{fc}} \\ A_{\mathrm{cf}} & A_{\mathrm{cc}} \end{bmatrix} = \begin{bmatrix} I \\ A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{\mathrm{ff}} & \\ & S_{\mathrm{cc}} \end{bmatrix} \begin{bmatrix} I & A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}} \\ & I \end{bmatrix}, \tag{21}$$

with the Schur complement

$$S_{\mathrm{cc}} := A_{\mathrm{cc}} - A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}}. \tag{22}$$

From this factorization, one can express symbolically the inverse of A explicitly,

$$A^{-1} = \begin{bmatrix} I & -A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}} \\ & I \end{bmatrix} \begin{bmatrix} A_{\mathrm{ff}}^{-1} & \\ & S_{\mathrm{cc}}^{-1} \end{bmatrix} \begin{bmatrix} I & \\ -A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1} & I \end{bmatrix}. \tag{23}$$

Defining the coarse restriction and extension matrices by

$$R_c := [-A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1}\ I], \quad P_c := \begin{bmatrix} -A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}} \\ I \end{bmatrix}, \tag{24}$$

and the more simple fine restriction and extension matrices by

$$R_f := [I \quad 0], \quad P_f := R_f^T, \tag{25}$$

one can obtain the following surprising result, which goes back to a special case where F-points are not mutually connected [2, 21, 22][3], see also [5]:

Lemma 1 *The inverse A^{-1} in (23) of the reordered system matrix A in (20) can be expressed as a sum of an inverse just acting on the fine variables, and a complementary inverse,*

$$A^{-1} = P_c(R_cAP_c)^{-1}R_c + P_f(R_fAP_f)^{-1}R_f. \tag{26}$$

Proof The proof is by a direct calculation: we first compute the term to be inverted in the first parentheses on the right,

$$\begin{aligned} R_cAP_c &= [-A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1}\ I] \begin{bmatrix} A_{\mathrm{ff}} & A_{\mathrm{fc}} \\ A_{\mathrm{cf}} & A_{\mathrm{cc}} \end{bmatrix} \begin{bmatrix} -A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}} \\ I \end{bmatrix} \\ &= [-A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1}\ I] \begin{bmatrix} 0 \\ A_{\mathrm{cc}} - A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}} \end{bmatrix} \\ &= S_{\mathrm{cc}}. \end{aligned} \tag{27}$$

For the fine nodes, we immediately obtain, because of the simplicity of P_f and R_f, that

$$R_fAP_f = A_{\mathrm{ff}}.$$

We thus get

$$\begin{aligned} &P_c(R_cAP_c)^{-1}R_c + P_f(R_fAP_f)^{-1}R_f \\ &\quad = \begin{bmatrix} -A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}} \\ I \end{bmatrix} S_{\mathrm{cc}}^{-1}[-A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1}\ I] + \begin{bmatrix} I \\ 0 \end{bmatrix} A_{\mathrm{ff}}^{-1}[I \quad 0] \\ &\quad = \left[\begin{array}{c|c} I & -A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}} \\ & I \end{array}\right] \begin{bmatrix} A_{\mathrm{ff}}^{-1} & \\ & S_{\mathrm{cc}}^{-1} \end{bmatrix} \begin{bmatrix} I & \\ -A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1} & I \end{bmatrix}, \end{aligned}$$

which is precisely the factored form of the inverse in (23), and concludes the proof.

This result is interesting when we look at a classical stationary iterative method with preconditioner $M \approx A$,

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + M^{-1}(\boldsymbol{f} - A\boldsymbol{u}^k). \tag{28}$$

The error $\boldsymbol{e}^k := \boldsymbol{u} - \boldsymbol{u}^k$ in this iteration satisfies

$$\boldsymbol{e}^{k+1} = (I - M^{-1}A)\boldsymbol{e}^k.$$

Using for $M^{-1} = A^{-1}$, the error propagator $(I - M^{-1}A)$ vanishes identically, but writing it down explicitly reveals the structure

$$\begin{aligned} 0 &= (I - A^{-1}A) \\ &= I - P_c(R_cAP_c)^{-1}R_cA - P_f(R_fAP_f)^{-1}R_fA, \end{aligned} \tag{29}$$

[1] See already [21]:"Die Punkte werden so in zwei Sorten – die 'Sorten A und B' – geteilt, dass keine zwei benachbarten Punkte zur gleichen Sorte gehören

[2] we use the same letters A, $\boldsymbol{u}$ and $\boldsymbol{f}$ also for the reordered system for simplicity

[3] See footnote 1

which looks like an optimal additive correction scheme between the fine and the coarse nodes in the method, i.e. a correction scheme which converges in one iteration step, and thus makes the method nilpotent, see also [4] for very recent conditions under which classical domain decomposition methods can become nilpotent, and [10–12, 14, 15] for how optimal coarse spaces can make such iterations nilpotent. To obtain a multiplicative correction scheme, we compute

$$\begin{aligned}&\left(I - P_c(R_cAP_c)^{-1}R_cA\right)\left(I - P_f(R_fAP_f)^{-1}R_fA\right)\\ &\quad = I - P_c(R_cAP_c)^{-1}R_cA - P_f(R_fAP_f)^{-1}R_fA\\ &\quad +P_c(R_cAP_c)^{-1}R_cAP_f(R_fAP_f)^{-1}R_fA,\end{aligned}$$

and the last product term cancels, because the middle term

$$\begin{aligned}R_cAP_f &= [-A_{\rm cf}A_{\rm ff}^{-1}\ I]\begin{bmatrix}A_{\rm ff} & A_{\rm fc}\\ A_{\rm cf} & A_{\rm cc}\end{bmatrix}\begin{bmatrix}I\\0\end{bmatrix}\\ &= [-A_{\rm cf}A_{\rm ff}^{-1}\ I]\begin{bmatrix}A_{\rm ff}\\ A_{\rm cf}\end{bmatrix} = 0.\end{aligned}$$

Therefore, the multiplicative correction scheme in this exact setting is simply (see e.g. [5])

$$\begin{aligned}0=&(I-A^{-1}A)\\ =&\left(I-P_c(R_cAP_c)^{-1}R_cA\right)\left(I-P_f(R_fAP_f)^{-1}R_fA\right).\end{aligned} \tag{30}$$

Such optimal algebraic correction schemes can be found in e.g. [2,19,21,22], but they are not very practical [19]:

> "Convergence does not mean anything if *numerical work* is not taken into account."

The key idea of AMG is therefore to approximate in these exact multiplicative and additive correction schemes the operators P_c, R_c and the inverses $A_{\rm ff}^{-1}$ and $S_{\rm cc}^{-1}$. We see that this is quite different from the geometric multigrid approach which is purely based on the fundamental idea that classical relaxation schemes for Laplace type operators are natural smoothers, and that coarse grids can represent coarse components of the error well. We see for example that in the coarse restriction R_c and prolongation P_c a fine inverse $A_{\rm ff}^{-1}$ is present, which could be approximated by relaxation, while the restriction and prolongation in geometric multigrid would not involve relaxation iterations.

Returning to the parareal algorithm, it was shown in [5,6] that the parareal algorithm can be written precisely as an approximation of the multiplicative preconditioner in (30), with only one approximation, namely the coarse correction operator R_cAP_c in (30) is approximated by a coarse time stepping algorithm:

Theorem 3 *In the parareal algorithm, the error propagation operator is*

$$(I - P_c\tilde{M}^{-1}R_cA)(I - P_f(R_fAP_f)^{-1}R_fA), \tag{31}$$

where $\tilde{M}$ is the coarse time stepping matrix defined in (8), A is the global system matrix in (6) and the restriction and prolongation operators are defined in (24) and (25).

Proof From the coarse stationary iteration formulation (9) of the parareal algorithm, we see that its coarse error propagation operator is given by

$$(I - \tilde{M}^{-1}\tilde{A}). \tag{32}$$

To obtain the error propagation operator for all variables, we introduce the additional restriction operator to the coarse variables $R := [0\ I]$ which just selects the coarse nodes to be used as initial conditions. Noting that P_c defined in (24) leaves the coarse nodes invariant, and just extends by fine solves in the interior, the error propagation operator of the parareal algorithm including the fine variables is

$$P_c(I - \tilde{M}^{-1}\tilde{A})R. \tag{33}$$

Now $\tilde{A} = R_cAP_c$ is precisely the Schur complement matrix we have seen in (27), since it has been obtained by elimination of the fine unknowns in (7) when forming the reduced system. We can thus rewrite the error propagation operator as

$$\begin{aligned}P_c(I - \tilde{M}^{-1}\tilde{A})R &= P_c(I - \tilde{M}^{-1}R_cAP_c)R\\ &= (I - P_c\tilde{M}^{-1}R_cA)P_cR,\end{aligned} \tag{34}$$

and since

$$P_cR = \begin{bmatrix}-A_{\rm ff}^{-1}A_{\rm fc}\\ I\end{bmatrix}[0\ I] = \begin{bmatrix}0 & -A_{\rm ff}^{-1}A_{\rm fc}\\ 0 & I\end{bmatrix} \tag{35}$$

is identical to

$$\begin{aligned}I - P_f(R_fAP_f)^{-1}R_fA &= I - \begin{bmatrix}A_{\rm ff}^{-1}\\ 0\end{bmatrix}\begin{bmatrix}A_{\rm ff} & A_{\rm fc}\end{bmatrix}\\ &= \begin{bmatrix}0 & -A_{\rm ff}^{-1}A_{\rm fc}\\ 0 & I\end{bmatrix},\end{aligned}$$

the error propagation operator of parareal is indeed (31), which concludes the proof.

We can now explain the main idea of the MGRIT algorithm: it is to replace the F-relaxation, the second term in the error propagation operator (31), by

$$\begin{aligned}&(I - P_f(R_fAP_f)^{-1}R_fA)(I - R^T(RAR^T)^{-1}RA)\\ &\qquad\times(I - P_f(R_fAP_f)^{-1}R_fA).\end{aligned} \tag{36}$$

This new relaxation (36) is called by the authors of [5] FCF-relaxation, and permits the use of more F-relaxation steps than just one in the parareal algorithm. The C-relaxation term $(I - R^T(RAR^T)^{-1}RA)$, which can be written explicitly as $[I\ 0; -A_{\rm cc}^{-1}A_{\rm cf}\ 0]$, precisely

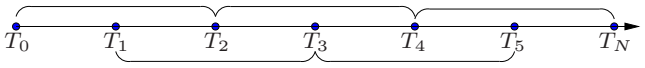

Fig. 1 FCF-relaxation interpreted as overlap in the parareal context.

closes the gap left by the matrix $E = [I_f \ 0; 0 \ 0]$ for the block Jacobi smoother with $\tilde{M}_J = [A_{\mathrm{ff}} \ A_{\mathrm{fc}}; 0 \ A_{\mathrm{cc}}]$ in the geometric multigrid interpretation (15). Indeed, one can verify that the CF-relaxation given by $(I - E\tilde{M}_J^{-1}A)(I - R^T(RAR^T)^{-1}RA) = I - \tilde{M}_J^{-1}A$ is exactly the block Jacobi smoother! One could even consider now $F(CF)^{\nu-1}$-relaxation doing ν relaxation steps.

The terminology FCF-relaxation seems to be particular to the AMG interpretation of the parareal algorithm. Historically, CF-relaxation can be found as early as [2]:

> "This suggests the use of $C-F$ relaxation which resembles the concept of red-black ordering. Gauss-Seidel relaxation is first performed (via some ordering) on the C points with a subsequent sweep (via some ordering) over the F points. This constitutes one $C - F$ sweep, which is the basis for relaxation in the current AMG"

see also [22], where the authors say:

> "In analogy to red-black relaxation, we first relax those equations which are associated with points in C in some order. Afterwards we relax the remaining equations in some order (C/F-relaxation)".

In the context of the parareal algorithm, the F-relaxation is already part of the algorithm, and only by adding a C-relaxation, a further F-relaxation step can produce further change and thus can make sense.

3 An overlapping parareal algorithm

We now show that the MGRIT algorithm can be interpreted as an overlapping variant of the parareal algorithm.

Theorem 4 *The two level MGRIT algorithm with FCF-smoother computes the same iterations as the parareal algorithm using generous overlap of one coarse time interval, i.e.*

$$\begin{aligned} &U_0^{k+1} = u_0, \quad U_1^{k+1} = \tilde{F}u_0, \\ &U_n^{k+1} = \tilde{F}\tilde{F}U_{n-2}^k + \tilde{G}U_{n-1}^{k+1} - \tilde{G}\tilde{F}U_{n-2}^k, \end{aligned} \tag{37}$$

see Figure 1 for a geometric representation.

Proof We first need to find an interpretation of the additional CF-relaxation in (36) used by MGRIT in terms of the parareal algorithm. For the C-part, we obtain

$$\begin{aligned} I - R^T(RAR^T)^{-1}RA &= I - \begin{bmatrix} 0 & 0 \\ 0 & A_{\mathrm{cc}}^{-1} \end{bmatrix} \begin{bmatrix} A_{\mathrm{ff}} & A_{\mathrm{fc}} \\ A_{\mathrm{cf}} & A_{\mathrm{cc}} \end{bmatrix} \\ &= \begin{bmatrix} I & 0 \\ -A_{\mathrm{cc}}^{-1}A_{\mathrm{cf}} & 0 \end{bmatrix}, \end{aligned}$$

and multiplying with the F-part we have seen in (35) leads to

$$\begin{bmatrix} I & 0 \\ -A_{\mathrm{cc}}^{-1}A_{\mathrm{cf}} & 0 \end{bmatrix} \begin{bmatrix} 0 & -A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}} \\ 0 & I \end{bmatrix} = \begin{bmatrix} 0 & -A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}} \\ 0 & A_{\mathrm{cc}}^{-1}A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}} \end{bmatrix}.$$

Now this term is multiplied from the left by the parareal error propagation operator (31) which equals $P_c(I - \tilde{M}^{-1}\tilde{A})R$ from (33). Multiplying with the result above, we get

$$\begin{aligned} &P_c(I - \tilde{M}^{-1}\tilde{A})R \begin{bmatrix} 0 & -A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}} \\ 0 & A_{\mathrm{cc}}^{-1}A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}} \end{bmatrix} \\ &\qquad = P_c(I - \tilde{M}^{-1}\tilde{A})[0 \quad A_{\mathrm{cc}}^{-1}A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}}]. \end{aligned}$$

Now parareal is only operating on the coarse variables, so we obtain for them the error propagation operator of MGRIT to be

$$(I - \tilde{M}^{-1}\tilde{A})A_{\mathrm{cc}}^{-1}A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}}. \tag{38}$$

Now recall the Schur complement

$$S_{\mathrm{cc}} = A_{\mathrm{cc}} - A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}},$$

which equals $\tilde{A}$ as we have seen in (27), and since in parareal $A_{\mathrm{cc}} = I$, because the original matrix only contains ones on the diagonal, we get

$$I - \tilde{A} = I - (A_{\mathrm{cc}} - A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}}) = A_{\mathrm{cc}}^{-1}A_{\mathrm{cf}}A_{\mathrm{ff}}^{-1}A_{\mathrm{fc}},$$

and inserting this into (38) we see that the error propagation operator of MGRIT on the coarse variables is simply

$$(I - \tilde{M}^{-1}\tilde{A})(I - \tilde{A}).$$

This corresponds to the two step iterative procedure

$$\begin{aligned} \boldsymbol{Y}^k &= \boldsymbol{U}^k + \boldsymbol{f} - \tilde{A}\boldsymbol{U}^k, \\ \tilde{M}\boldsymbol{U}^{k+1} &= \tilde{M}\boldsymbol{Y}^k + \boldsymbol{f} - \tilde{A}\boldsymbol{Y}^k. \end{aligned} \tag{39}$$

Writing this componentwise, we obtain

$$Y_0^k = u_0, \quad Y_n^k = \tilde{F}U_{n-1}^k \tag{40}$$

$$U_0^k = u_0, \quad U_n^{k+1} = \tilde{F}Y_{n-1}^k + \tilde{G}U_{n-1}^{k+1} - \tilde{G}Y_{n-1}^k. \tag{41}$$

Substituting the values of Y_n^k into the equation for U_n^{k+1} then yields (37), which concludes the proof.

Corollary 1 *The two level MGRIT algorithm with the $F(CF)^\nu$-smoother computes the same iterations as the parareal algorithm using $\nu\Delta T$ overlap.*

Proof Analogously to the proof of Theorem 4, we obtain for the error propagation operator

$$(I - \tilde{M}^{-1}\tilde{A})(I - \tilde{A})^\nu,$$

which corresponds to the $(\nu+1)$-step iterative procedure

$$\begin{aligned} \boldsymbol{U}^{k+\frac{\mu}{\nu+1}} &= \boldsymbol{U}^{k+\frac{\mu-1}{\nu+1}} + \boldsymbol{f} - \tilde{A}\boldsymbol{U}^{k+\frac{\mu-1}{\nu+1}},\ \mu = 1,..,\nu, \\ \tilde{M}\boldsymbol{U}^{k+1} &= \tilde{M}\boldsymbol{U}^{k+\frac{\nu}{\nu+1}} + \boldsymbol{f} - \tilde{A}\boldsymbol{U}^{k+\frac{\nu}{\nu+1}}. \end{aligned} \tag{42}$$

Writing this componentwise, we obtain

$$U_0^{k+\frac{\mu}{\nu+1}} = u_0, \quad U_n^{k+\frac{\mu}{\nu+1}} = \tilde{F}U_{n-1}^{k+\frac{\mu-1}{\nu+1}},\ \mu = 1,..,\nu, \tag{43}$$

$$U_0^{k+1} = u_0, \quad U_n^{k+1} = \tilde{F}U_{n-1}^{k+\frac{\nu}{\nu+1}} + \tilde{G}U_{n-1}^{k+1} - \tilde{G}U_{n-1}^{k+\frac{\nu}{\nu+1}}, \tag{44}$$

and the result follows.

We can now prove our first convergence result on this new, overlapping parareal algorithm, namely convergence in a finite number of steps, smaller than for the classical parareal algorithm which converges the latest after N iterations:

Theorem 5 *The parareal algorithm with $\nu\Delta T$ overlap, which is equivalent to MGRIT with $F(CF)^\nu$-relaxation, converges the latest after $k = \lceil N/(\nu+1)\rceil$ iterations, independently of the initial guess U_n^0 for $n = 1, 2, \ldots$.*

Proof The proof is by induction on k. For the base case $k = 0$, we obtain from (43)

$$U_n^{\frac{\nu}{\nu+1}} = \tilde{F}^n u_0, \quad 0 \le n \le \nu,$$

and therefore by (44)

$$U_{n+1}^1 = \tilde{F}\tilde{F}^n u_0 + \tilde{G}U_n^1 - \tilde{G}\tilde{F}^n u_0, \quad 0 \le n \le \nu.$$

Thus, by induction on n and the fact that $U_0^1 = u_0$, we obtain $U_n^1 = \tilde{F}^n u_0$ for $0 \le n \le \nu+1$. Next, assume that for $N \le k(\nu+1)$ coarse time intervals, we have after k iterations the correct fine solutions,

$$U_n^k = \tilde{F}^n u_0, \quad \text{for } n = 0, 1, \ldots, N. \tag{45}$$

Then from (43) we obtain for $0 \le n \le N$

$$U_{n+\mu}^{k+\frac{\mu}{\nu+1}} = \tilde{F}^\mu\tilde{F}^n u_0 = \tilde{F}^{n+\mu}u_0, \quad \text{for } 0 \le \mu \le \nu.$$

So for n up to $(k+1)(\nu+1)$, we deduce from (44) that

$$U_{n+1}^{k+1} = \tilde{F}\tilde{F}^n u_0 + \tilde{G}U_n^{k+1} - \tilde{G}\tilde{F}^n u_0 \quad \text{for } 0 \le n \le k(\nu+1)+\nu.$$

An induction on n now shows that $U_n^{k+1} = \tilde{F}^n u_0$ for all $n \le (k+1)(\nu+1)$; in other words, the method converges to the exact solution in the first $(k+1)(\nu+1)$ intervals. This concludes the proof by induction.

We next prove a new, general convergence estimate for MGRIT which is equivalent to overlapping parareal, in the same spirit as [8]:

Theorem 6 *Let $F(T_n, T_{n-1}, U_{n-1}^k) = \Phi_{\Delta T_{n-1}}(U_{n-1}^k)$ be the exact solution on the time subdomain $\Omega_{n-1} = (T_{n-1}, T_n)$, and let $G(T_n, T_{n-1}, U_{n-1}^k)$ be a coarse integrator such that the local truncation error $\tau(t, \Delta T, z) := F(t+\Delta T, t, z) - G(t+\Delta T, t, z)$ satisfies for all z*

$$\|\tau(t, \Delta T, y) - \tau(t, \Delta T, z)\| \le C_1 \Delta T H^p \|y - z\|,$$
$$\|\tau(t, \Delta T, z)\| \le C_3 \Delta T H^p,$$

where H is the time step parameter for the coarse integrator, and p is the order of the truncation error of the coarse integrator G. Moreover, assume that F and G are Lipschitz continuous, and that the constant C_2 is such that

$$\|F(t+\Delta T, t, y) - F(t+\Delta T, t, z)\| \le (1 + C_2\Delta T)^q \|y - z\|,$$
$$\|G(t+\Delta T, t, y) - G(t+\Delta T, t, z)\| \le (1 + C_2\Delta T)^q \|y - z\|$$

with $C_2 > 0$ and $q \in \{-1, 1\}$, depending on whether perturbations are amplified or attenuated by the solution operator. Then the error in Ω_{n-1} after k iterations of overlapping parareal is bounded above by $\tilde{E}_n^k$, where

$$\tilde{E}_n^k = \begin{cases} 0, & n \le (\nu+1)k, \\ \gamma\alpha^k\beta^{k\nu}c_{n-(\nu+1)k-1}^{(k)}, & n \ge (\nu+1)k+1, \end{cases}$$

where $\alpha = C_1 H^p \Delta T$, $\beta = (1 + C_2\Delta T)^q$ and $\gamma = C_3 H^p \Delta T$, and

$$c_n^{(k)} = \sum_{j=0}^{n} \binom{k+j}{j} \beta^j.$$

More explicitly, in the $q = 1$ (amplifying) case, we have

$$\tilde{E}_n^k \le \frac{C_3}{C_1} \frac{(C_1 T_{n-k\nu})^{k+1}}{(k+1)!} e^{C_2(T_n - T_{k+1})} H^{(k+1)p}$$

for $n \ge (\nu+1)k+1$, and in the $q = -1$ (decaying) case, we have

$$\tilde{E}_n^k \le \frac{C_3}{C_1(1 + C_2\Delta T)^{k(\nu-1)-1}} \left(\frac{C_1}{C_2}\right)^{k+1} H^{p(k+1)}$$

for $n \ge (\nu+1)k+1$.

Proof From the definition of the overlapping parareal algorithm (40)–(41), we obtain, using the fact that F is the exact solution, that

$$\begin{aligned} U_n^{k+1} - u(T_n) &= F(T_n, T_{n-1}, U_{n-1}^{k+\frac{\nu}{\nu+1}}) + G(T_n, T_{n-1}, U_{n-1}^{k+1}) \\ &\quad - G(T_n, T_{n-1}, U_{n-1}^{k+\frac{\nu}{\nu+1}}) - F(T_n, T_{n-1}, u(T_{n-1})) \\ &= (F(T_n, T_{n-1}, U_{n-1}^{k+\frac{\nu}{\nu+1}}) - G(T_n, T_{n-1}, U_{n-1}^{k+\frac{\nu}{\nu+1}})) \\ &\quad - (F(T_n, T_{n-1}, u(T_{n-1})) - G(T_n, T_{n-1}, u(T_{n-1}))) \\ &\quad + G(T_n, T_{n-1}, U_{n-1}^{k+1}) - G(T_n, T_{n-1}, u(T_{n-1})). \end{aligned}$$

Using the assumption on τ for the first two terms and the Lipschitz condition on G for the last term, we get

$$\begin{aligned}\|U_n^{k+1} - u(T_n)\| \le C_1 H^p \Delta T \|U_{n-1}^{k+\frac{\nu}{\nu+1}} - u(T_{n-1})\| \\ + (1 + C_2\Delta T)^q \|U_{n-1}^{k+1} - u(T_{n-1})\|,\end{aligned}$$

On the other hand, using the fact that

$$U_{n-1}^{k+\frac{\mu}{\nu+1}} = F(T_n, T_{n-1}, U_{n-1}^{k+\frac{\mu-1}{\nu+1}})$$

for $\mu = 1, \dots, \nu$, we obtain

$$\begin{aligned}&\|U_{n-1}^{k+\frac{\mu}{\nu+1}} - u(T_n)\| \\ &\quad \le \|F(T_n, T_{n-1}, U_{n-1}^{k+\frac{\mu-1}{\nu+1}}) - F(T_n, T_{n-1}, u(T_{n-1}))\| \\ &\quad \le (1 + C_2\Delta T)^{\mu q} \|U_{n-\mu}^k - u(T_{n-1})\|,\end{aligned}$$

with the convention that the right hand side becomes zero whenever $\mu > n$. Substituting gives the estimate

$$\begin{aligned}&\|U_n^{k+1} - u(T_n)\| \\ &\quad \le C_1 H^p \Delta T (1 + C_2\Delta T)^{\nu q} \|U_{n-\nu-1}^k - u(T_{n-\nu-1})\| \\ &\qquad + (1 + C_2\Delta T)^q \|U_{n-1}^{k+1} - u(T_{n-1})\|,\end{aligned}$$

which is valid for $n \ge \nu + 1$, and for $n = 1, \dots, \nu$, we simply have

$$\|U_1^{k+1} - u(T_1)\| \le (1 + C_2\Delta T)^{nq} \|U_0^{k+1} - u(T_0)\| = 0. \quad (46)$$

Moreover, since the initial guess U_n^0 is obtained by coarse integration, i.e., $U_n^0 = G(T_n, T_{n-1}, U_{n-1}^0)$ for $n \ge 1$, subtracting the exact solution yields

$$\begin{aligned}&\|U_n^0 - u(T_n)\| \\ &\le \|\tau(T_{n-1}, \Delta T, u(T_{n-1}))\| + \|Gu(T_{n-1}) - GU_{n-1}^0\|.\end{aligned}$$

Thus, we want to study the recurrence relation

$$\tilde{E}_n^{k+1} = \alpha\beta^\nu \tilde{E}_{n-\nu-1}^k + \beta\tilde{E}_{n-1}^{k+1}, \quad \tilde{E}_n^0 = \gamma + \beta\tilde{E}_{n-1}^0, \quad (47)$$

where $\alpha = C_1 H^p \Delta T$, $\beta = (1 + C_2\Delta T)^q$ and $\gamma = C_3 H^p \Delta T$, since $\tilde{E}_n^{k+1}$ is then an upper bound for $\|U_n^{k+1} - u(T_n)\|$. Multiplying (47) by ζ^n and summing over $n \ge \nu + 1$, we find that the generating function $\rho_k(\zeta) = \sum_{n\ge1} \tilde{E}_n^k \zeta^n$ satisfies the recurrence relation

$$\begin{aligned}\rho_{k+1}(\zeta) &= \alpha\beta^\nu \zeta^{\nu+1} \rho_k(\zeta) + \beta\zeta\rho_{k+1}(\zeta), \\ \rho_0(\zeta) &= \frac{\gamma\zeta}{1-\zeta} + \beta\zeta\rho_0(\zeta)\end{aligned}$$

where we used the fact that $\tilde{E}_n^{k+1} = 0$ for $n \le \nu$, which is true from (46). Solving for ρ_k by induction, we get

$$\rho_k(\zeta) = \left(\frac{\alpha\beta^\nu\zeta^{\nu+1}}{1-\beta\zeta}\right)^k, \quad \rho_0(\zeta) = \frac{\gamma}{1-\zeta} \frac{\alpha^k\beta^{k\nu}\zeta^{(\nu+1)k+1}}{(1-\beta\zeta)^{k+1}}. \quad (48)$$

A direct calculation of the power series of ρ_k yields

$$\begin{aligned}\rho_k(\zeta) &= \gamma\alpha^k\beta^{k\nu}\zeta^{(\nu+1)k+1} \left(\sum_{i\ge0} \zeta^i\right) \left(\sum_{j\ge0} \binom{k+j}{j} \beta^j \zeta^j\right) \\ &= \gamma\alpha^k\beta^{k\nu} \sum_{n\ge0} \left(\sum_{j=0}^n \binom{k+j}{j} \beta^j\right) \zeta^{n+(\nu+1)k+1} \\ &= \sum_{n\ge0} \gamma\alpha^k\beta^{k\nu} c_n^{(k)} \zeta^{n+(\nu+1)k+1}.\end{aligned}$$

Thus, we have

$$\tilde{E}_n^k = \gamma\alpha^k\beta^{k\nu} c_{n-(\nu+1)k-1}^{(k)}, \qquad n \ge (\nu+1)k+1,$$

and $\tilde{E}_n^k = 0$ otherwise. Note that if $0 < \beta < 1$, i.e., if $q = -1$, then we have the estimate

$$c_n^{(k)} = \sum_{j=0}^n \binom{k+j}{j} \beta^j \le \frac{1}{(1-\beta)^{k+1}} = \frac{1}{\beta^{k+1}(C_2\Delta T)^{k+1}},$$

where the inequality is due to the fact that $c_n^{(k)}$ is a truncated Taylor series of the function $f(\beta) = (1-\beta)^{-(k+1)}$, with a remainder term that is necessarily positive. Then we see that the error of the overlapping parareal algorithm is bounded above by

$$\begin{aligned}\tilde{E}_n^k &= \gamma\alpha^k\beta^{k\nu} c_{n-(\nu+1)k-1}^{(k)} \\ &\le \frac{C_3}{C_1} \left(\frac{C_1}{C_2}\right)^{k+1} \beta^{k(\nu-1)-1} H^{p(k+1)},\end{aligned}$$

i.e., independent of n, cf. [16]. On the other hand, if $q = 1$, i.e., if $\beta \ge 1$, then replacing the factor $1 - \zeta$ by $1 - \beta\zeta$ in the denominator of (48) only increases the coefficients in the power series of $\rho_k(\zeta)$. Using now the binomial expansion

$$\frac{1}{(1-\beta\zeta)^{k+2}} = \sum_{j\ge0} \binom{k+1+j}{j} \beta^j \zeta^j,$$

we obtain the power series expansion

$$\begin{aligned}\tilde{\rho}_k(\zeta) &:= \frac{\gamma\alpha^k\beta^{k\nu}\zeta^{(\nu+1)k+1}}{(1-\beta\zeta)^{k+2}} \\ &= \gamma \sum_{j\ge0} \binom{k+1+j}{j} \alpha^k\beta^{j+k\nu}\zeta^{j+(\nu+1)k+1}.\end{aligned}$$

So we have the upper bound

$$\tilde{E}_n^k \le \binom{n-k\nu}{k+1} \gamma\alpha^k\beta^{n-k-1}$$

for $n \ge (\nu+1)k+1$, which upon substitution yields

$$\begin{aligned}\tilde{E}_n^k &\le \binom{n-k\nu}{k+1} C_3 C_1^k H^{(k+1)p} (\Delta T)^{k+1} (1 + C_2\Delta T)^{n-k-1} \\ &\le \frac{C_3}{C_1} \frac{(C_1 T_{n-k\nu})^{k+1}}{(k+1)!} e^{C_2(T_n - T_{k+1})} H^{(k+1)p},\end{aligned}$$

as required, which concludes the proof.

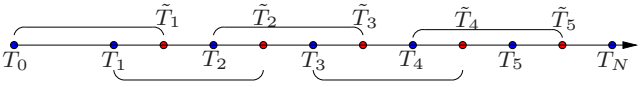

Fig. 2 General overlap.

For comparison, the convergence of non-overlapping parareal is given by

$$E_n^k = \begin{cases} 0, & n \le k, \\ \binom{n}{k+1}\gamma\alpha^k\beta^{n-k-1}, & n \ge k+1 \text{ and } q=1, \\ \gamma\alpha^k c_{n-k-1}^{(k)}, & n \ge k+1 \text{ and } q=-1, \end{cases}$$

with the same definitions of α, β, γ and $c_n^{(k)}$. Thus, we see that the overlapping method always converges in fewer iterations than the non-overlapping method.

Under the natural assumption that overlapping parareal is $(\nu+1)$ times as expensive as non-overlapping parareal, we should consider the ratio

$$\frac{\tilde{E}_n^k}{E_n^{(\nu+1)k}} = \left(\frac{\beta}{\alpha}\right)^{k\nu} \frac{c_{n-2k-1}^{(k)}}{c_{n-2k-1}^{((\nu+1)k)}} = \left(\frac{\beta}{\alpha}\right)^{k\nu} \frac{\sum_{j=0}^{n-(\nu+1)k-1}\binom{k+j}{j}\beta^j}{\sum_{j=0}^{n-(\nu+1)k-1}\binom{(\nu+1)k+j}{j}\beta^j}.$$

Clearly, the second fraction is less than 1, so the overlapping method will be an improvement over the non-overlapping one if $\beta < \alpha$. Since α measures the error of the coarse integrator, this means overlapping yields an improvement if the problem is so strongly diffusive that the coarse integrator becomes relatively inaccurate over an interval of length ΔT.

We considered also the case with general overlap, see Figure 2, but quickly found that then the coarse correction has to go through all the nodes $T_i, \tilde{T}_i, i = 1, .., N-1$, and the resulting method does therefore not look advantageous.

4 Numerical experiments

We now present several numerical experiments on model problems with increasing complexity. We start with the test equation of Dahlquist, using Backward Euler, and also a fourth order Runge-Kutta scheme. We then test the algorithms on the Lorentz equations, before turning our attention to partial differential equations, namely the heat equation, the advection diffusion equation, and the Burger's equation. We will see that overlap can be worthwhile in certain situations, whereas in others it is not useful to use overlap, it can even lead to a slower algorithm considering overall run time. The free software Julia [1] is used for some of the computing.

4.1 Dahlquist test equation

We first use the Dahlquist test equation,

$$u_t = \lambda u, \quad \text{in } (0,T) \text{ with } u(0) = 1,$$

discretized by Backward Euler. With $\lambda = -1$, $T = 5$, $\Delta T = T/10$, which corresponds to using 10 processors, and $\Delta t = \Delta T/20$, we obtain the convergence curves shown in the top row of Figure 3. We see that the algorithms are here in their superlinear convergence regime, and while in the left panel it seems that the overlap is beneficial, because convergence is faster, the right panel which corresponds more to equal work per iteration shows that since the work with overlap is double, parareal without overlap converges faster. Performing the same experiment on a longer time interval, $T = 50$, with $\Delta T = T/100$, which would correspond to using 100 processors on this longer time interval, and the same $\Delta t = \Delta T/20$ as before, we get the convergence curves shown in the second row in Figure 3. The algorithm is now in the linear convergence regime, but again it seems that the overlap does not pay off. We now repeat the experiment on the long time interval, but use less processors, only 50, $\Delta T = T/50$, $\Delta t = \Delta T/40$, see the third row in Figure 3. We see that it still does not pay off to use overlap, but the difference has become smaller. So we continue in this direction on the long time interval, using only 25 processors, $\Delta T = T/25$, $\Delta t = \Delta T/80$, see the bottom row in Figure 3. Now it seems that even counting the overlapping iteration as double the work, the overlapping variant converges faster, as one can see in the bottom right panel of Figure 3.

We next use a classical RK4 scheme to solve the Dahlquist test equation. We show in Figure 4 in the top row the results for $\lambda = -1$, $T = 50$, $\Delta T = T/50$, which would correspond to 50 processors, and $\Delta t = \Delta T/40$. Again the overlap is not paying off, but when using only 25 processors in the same experiment, i.e. $\Delta T = T/25$, $\Delta t = \Delta T/80$, even counting work, the bottom row in Figure 4 shows now that overlap leads to a faster method.

4.2 Lorenz equation

We now simulate the Lorenz equations,

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

with parameters $\sigma = 10$, $r = 28$ and $b = \frac{8}{3}$ so that the equations are in the chaotic regime for solutions. We

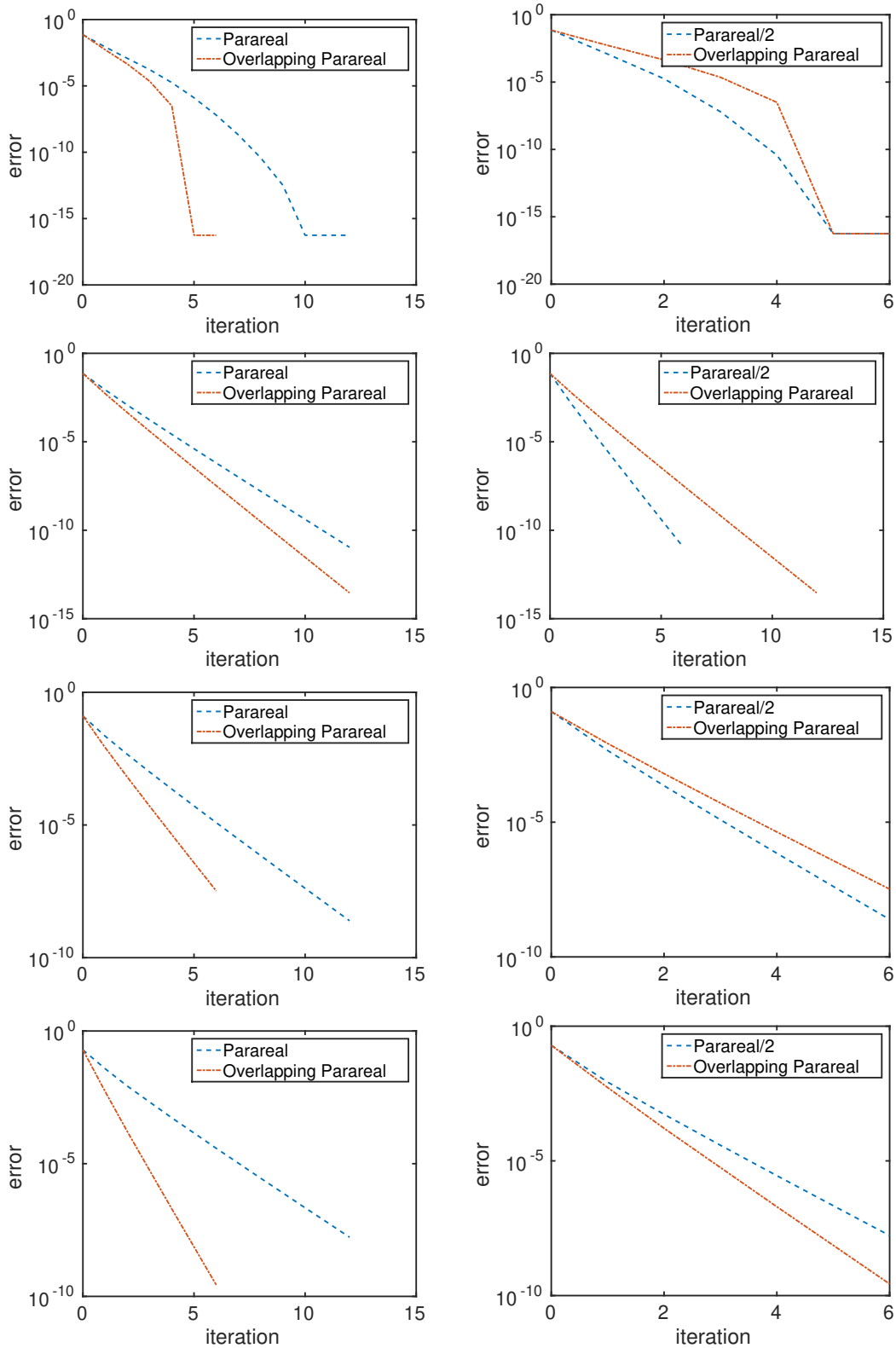

Fig. 3 Dahlquist test equation with Backward Euler. Left: error as a function of iteration number for the classical parareal algorithm and the overlapping variant corresponding to MGRIT. Right: same, but counting parareal iterations only half since MGRIT uses two fine solves per iteration. Top row: $T = 5$ using 10 processors. Second row: $T = 50$ using 100 processors, i.e. same coarse time interval size as in the top row. Third row: $T = 50$ using 50 processors, i.e. double the coarse time interval size compared to the second row. Last row: $T = 50$ using 25 processors, i.e. four times the coarse time interval size compared to the second row.

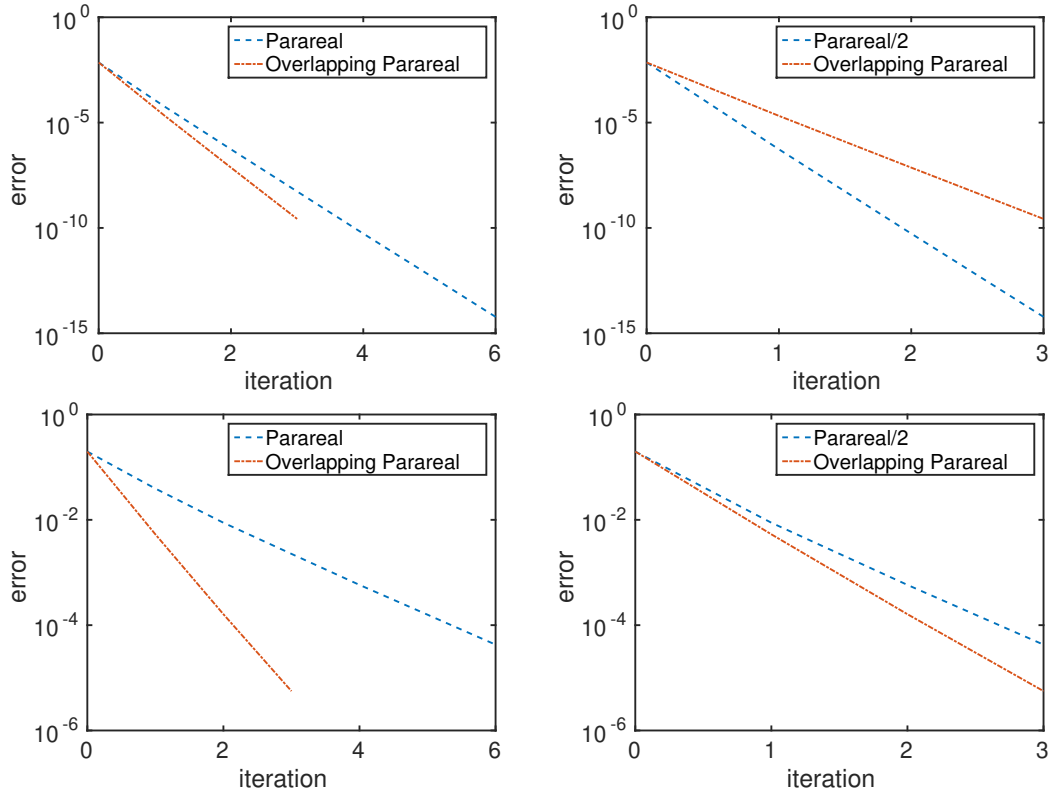

Fig. 4 Dahlquist test equation as in Figure 3 but with RK4 instead of Backward Euler. Top row: $T = 50$ using 50 processors. Bottom row: $T = 50$ using 25 processors.

use the initial conditions $(x, y, z)(0) = (20, 5, -5)$, and again a standard fourth order Runge Kutta method for the numerical integration. For the configuration $T = 10$, $\Delta T = T/180$, which corresponds to using 180 processors, and $\Delta t = \Delta T/10$, the convergence curves are shown in the top row of Figure 5. We see that the algorithm is in its superlinear convergence regime, and overlap does virtually not help at all, the non-overlapping method converges in about the same number of iterations, and is much faster if one counts actual work. Performing the same experiment with only half the number of processors, $\Delta T = T/90$, $\Delta t = \Delta T/10$, we get the results in the bottom row of Figure 5, which show that overlap is still not helpful.

4.3 Heat equation

We experiment with the heat equation,

$$u_t = \Delta u + x^4(1-x) + t^2 \quad \text{in } (0,1) \times (0,T)$$

with $u(x, 0) = 0$. We discretize in space using $\Delta x = 1/10$, and integrate in time using the backward Euler method. For $T = 8$, $\Delta T = T/64$ corresponding to 64 processors, and $\Delta t = \Delta T/5$, the convergence results are shown in the top row of Figure 6. We see that the overlap is not competitive in this case. If we perform the same experiment, but using 32 processors, i.e. with $\Delta T = T/32$, $\Delta t = \Delta T/10$, we get the results shown in the middle row of Figure 6, where the overlapping variant is slightly faster. Then, in the bottom row, with $\Delta T = T/16$ corresponding to 16 processors and $\Delta t = \Delta T/20$, the gain is even more substantial.

To understand this, we need to consider the convergence behavior of overlapping versus classical parareal as a function of the decay factor. To make things concrete, suppose we apply our algorithms on $du/dt = -\lambda y$ with $\lambda > 0$, using Backward Euler as the coarse solver. Using the same notation as in Theorem 6, this leads to

$$\alpha \approx \frac{\lambda^2}{2} e^{-\lambda \Delta T}, \quad \beta = e^{-\lambda \Delta T},$$

so that $\beta/\alpha = 2/\lambda^2$ is small when λ is large, i.e., when the system is strongly diffusive. As mentioned in the remark immediately after Theorem 6, a small β/α means the overlapping method compares favorably against classical parareal, and in the heat equation, there are many such strongly diffusive components (e.g., all the high frequency components in space). This is why overlapping parareal performs better than classical parareal in this case, even with the additional cost taken into account.

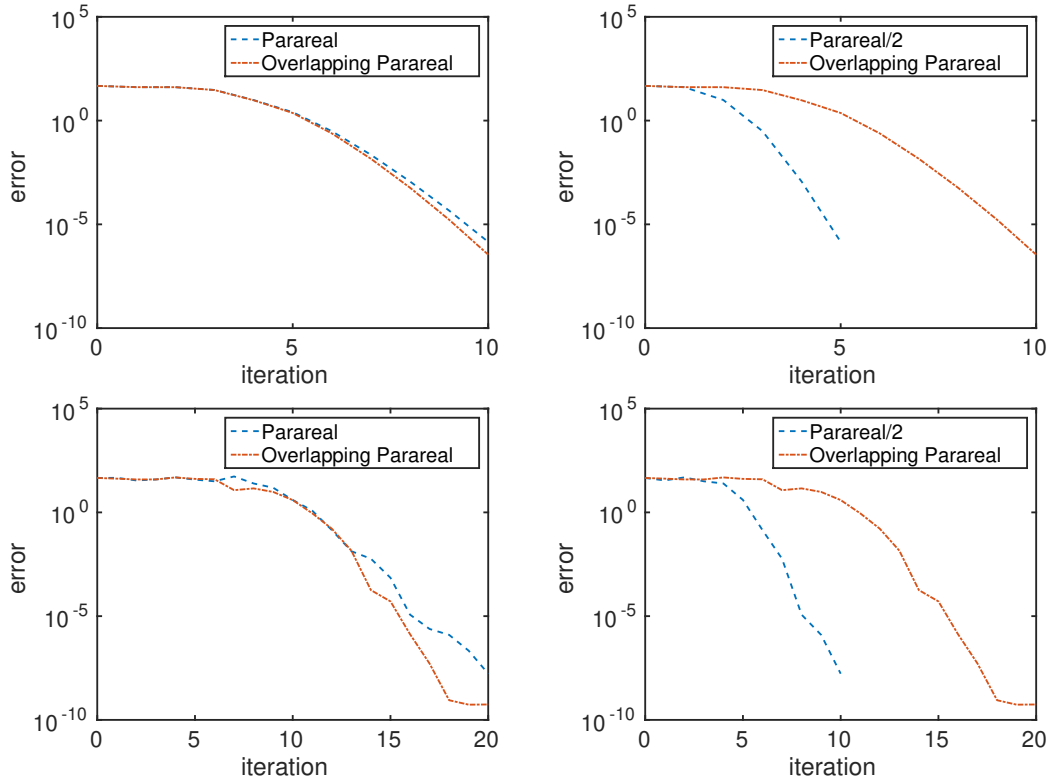

Fig. 5 Lorenz equations using RK4. Left: error as a function of iteration number for the classical parareal algorithm and the overlapping variant using 180 processors. Right: same, but counting parareal iterations only half. Top row: $T = 10$ using 180 processors. Bottom row: $T = 10$ using only 90 processors.

4.4 Advection-diffusion equation

We test with the advection-diffusion equation

$$u_t = au_{xx} - bu_x + x^4(1-x) + t^2 \quad \text{in } (0,1) \times (0,T)$$

with $u(x,0) = 0$, $a = 0.001$ and $b = 1$. We discretize in space using $\Delta x = 1/100$ with upwinding for the advection term bu_x, and integrate in time using again the backward Euler method. For $T = 8$, $\Delta T = T/64$, $\Delta t = \Delta T/5$, the convergence results are shown in the top row of Figure 7. We see that overlapping parareal outperforms its classical counterpart, even when we consider it twice as expensive per iteration. Just like the heat equation, the advection-diffusion equation has many strongly diffusive components, which is why the additional cost of the overlapping version is worth the effort. The gain is even more pronounced when we use 16 processors, see the bottom row of Figure 7.

4.5 Burger's equation

We finally solve the viscous Burger's equation

$$u_t = au_{xx} - uu_x + x^4(1-x) + t^2 \quad \text{in } (0,1) \times (0,T)$$

with $u(x,0) = 0$, $a = 0.001$. We discretize in space with $\Delta x = 1/100$ and upwinding for the advection term uu_x, and integrate in time with the backward Euler method. For $T = 8$, $\Delta T = T/128$, $\Delta t = \Delta T/5$, the convergence results are shown in the top row of Figure 8. We see that the overlap accelerates the convergence because of the diffusivity, and we definitely gain when fewer processors are used, see the middle and bottom rows of Figure 8. Note that the error of the overlapping method drops to zero at the seventh iteration, so it is not plotted in the bottom row of Figure 8.

The trend of faster convergence when H ($= \Delta T$ in the experiments) increases is not predictable from our theoretical results in this paper. One would need to do a second kind of analysis like in [13, Theorem 5.3 and Theorem 5.5], based essentially on the diffusivity (see [13, Assumption 5.1]), which we leave for future work.

5 Conclusions

We have shown three different two-level interpretations of the parareal algorithm. Each interpretation permits the extension of the pararal algorithm to a multilevel variant. The third one of these interpretations led to the

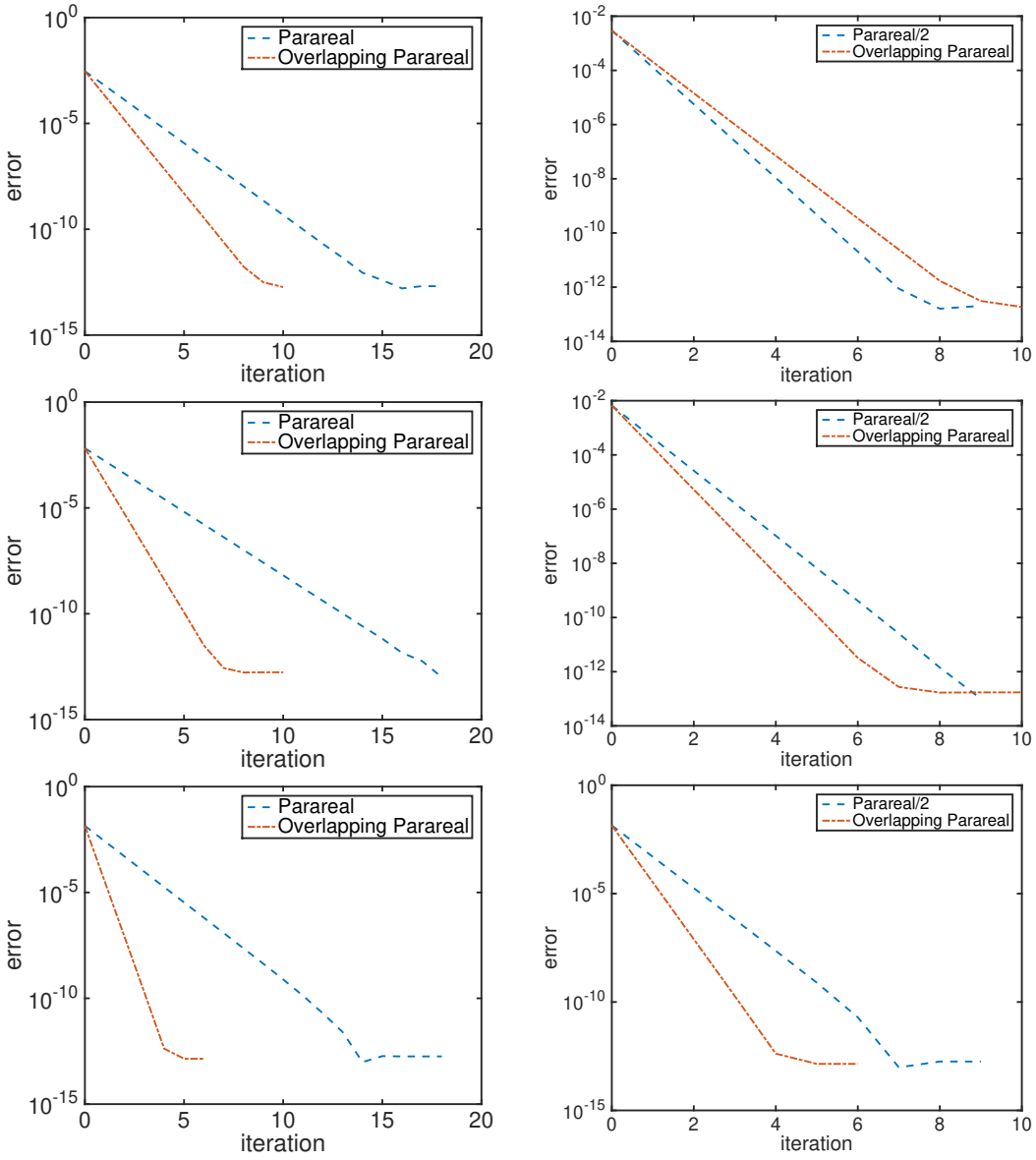

Fig. 6 Heat equation case using 64 processors in the top row, 32 in the middle row, and 16 in the bottom row.

MGRIT algorithm, which also uses a so-called FCF-smoother, instead of the simple F-smoother used by the parareal algorithm. We showed that MGRIT can be interpreted in turn as a parareal algorithm with overlap, and that using an $F(CF)^\nu$-smoother implies using ν coarse time intervals of overlap. We then gave two new convergence results for this overlapping parareal algorithm, one that shows convergence in a finite number of steps, which becomes smaller the more overlap one is using, and an accurate convergence estimate. Both results hold for the fully non-linear case of the algorithm. We illustrated our results with numerical experiments, which showed that sometimes overlap can lead to a faster algorithm, but in other cases the method, even though it is converging faster, becomes overall slower, because of the cost of the overlap. These results apply equally to the MGRIT algorithm with $F(CF)^\nu$-smoother in the fully non-linear setting.

Acknowledgment: We would like to thank Jacob Schroder, Rob Falgout and Panayot Vassilevski for the very helpful discussions about algebraic multigrid, and for the historical references.

References

1. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. SIAM Review **59**(1), 65–98 (2017)
2. Brandt, A., McCormick, S., Ruge, J.: Algebraic multigrid (amg) for automatic multigrid solutions with application

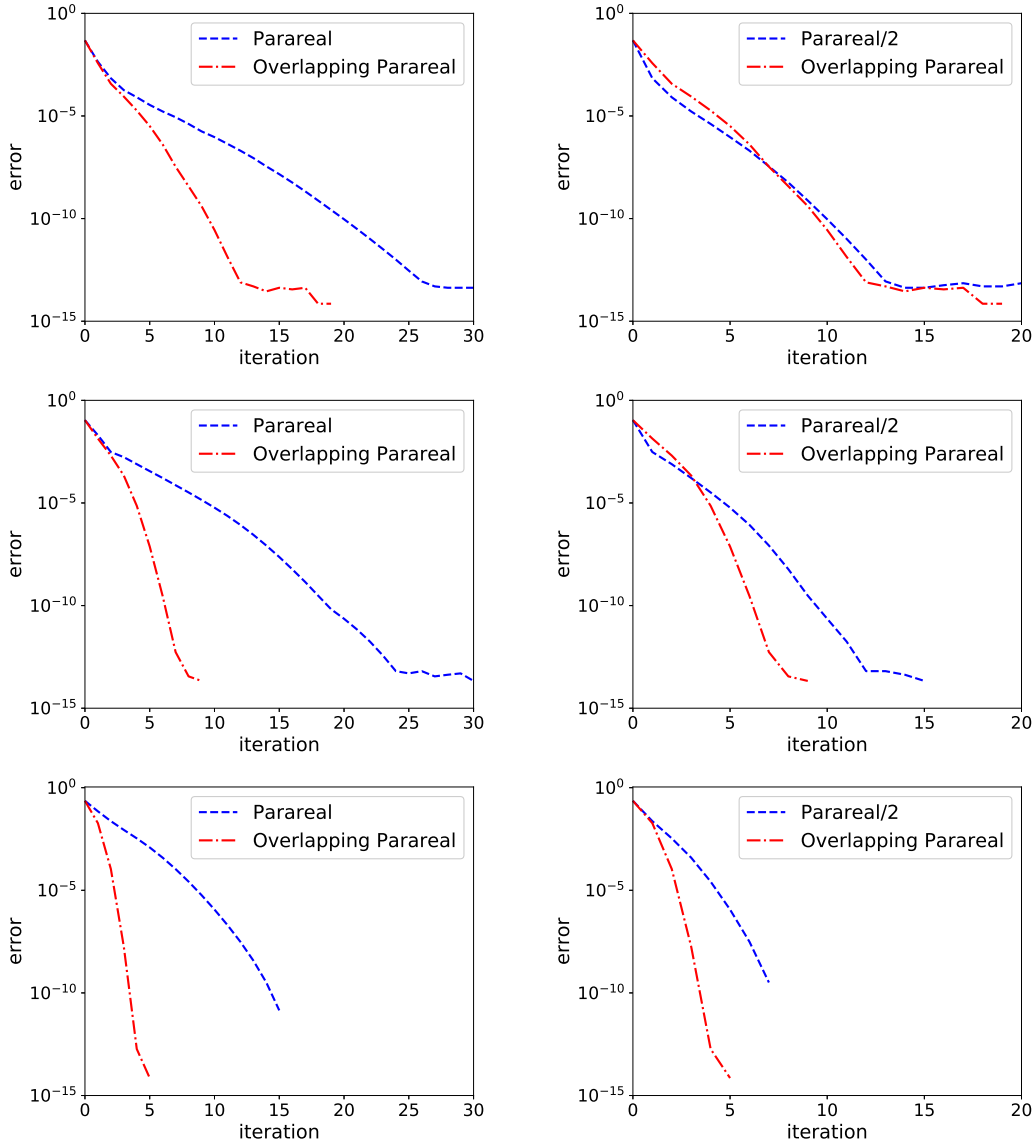

Fig. 7 Advection-diffusion equation case using 64 processors in the top row, 32 in the middle row, and 16 in the bottom row.

to geodetic computations. Report, Inst. for computational Studies, Fort collins, colo (1982)
3. Buzbee, B.L., Golub, G.H., Nielson, C.W.: On direct methods for solving poisson's equations. SIAM Journal on Numerical analysis **7**(4), 627–656 (1970)
4. Chaouqui, F., Gander, M.J., Santugini-Repiquet, K.: On nilpotent subdomain iterations. Domain Decomposition Methods in Science and Engineering XXIII, Springer (2016)
5. Falgout, R.D., Friedhoff, S., Kolev, T., MacLachlan, S.P., Schroder, J.B.: Parallel time integration with multigrid. SIAM Journal on Scientific Computing **36**(6), C635–C661 (2014)
6. Friedhoff, S., Falgout, R., Kolev, T., MacLachlan, S., Schroder, J.B.: A multigrid-in-time algorithm for solving evolution equations in parallel. In: Sixteenth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, United States (2013)
7. Gander, M.J.: 50 years of time parallel time integration. In: Multiple Shooting and Time Domain Decomposition Methods, pp. 69–113. Springer (2015)
8. Gander, M.J., Hairer, E.: Nonlinear convergence analysis for the parareal algorithm. In: O.B. Widlund, D.E. Keyes (eds.) Domain Decomposition Methods in Science and Engineering XVII, *Lecture Notes in Computational Science and Engineering*, vol. 60, pp. 45–56. Springer (2008)
9. Gander, M.J., Hairer, E.: Analysis for parareal algorithms applied to Hamiltonian differential equations. Journal of Computational and Applied Mathematics **259**, 2–13 (2014)
10. Gander, M.J., Halpern, L.: Méthodes de décomposition de domaines–notions de base. Techniques de l'ingénieur, Méthodes numériques, base documentaire: TIB105DUO. (ref. article: af1375) (2013)
11. Gander, M.J., Halpern, L., Santugini-Repiquet, K.: Discontinuous coarse spaces for DD-methods with discontinuous iterates. In: Domain Decomposition Methods in Science and Engineering XXI, pp. 607–615. Springer (2014)
12. Gander, M.J., Halpern, L., Santugini-Repiquet, K.: A new coarse grid correction for RAS/AS. In: Domain De-

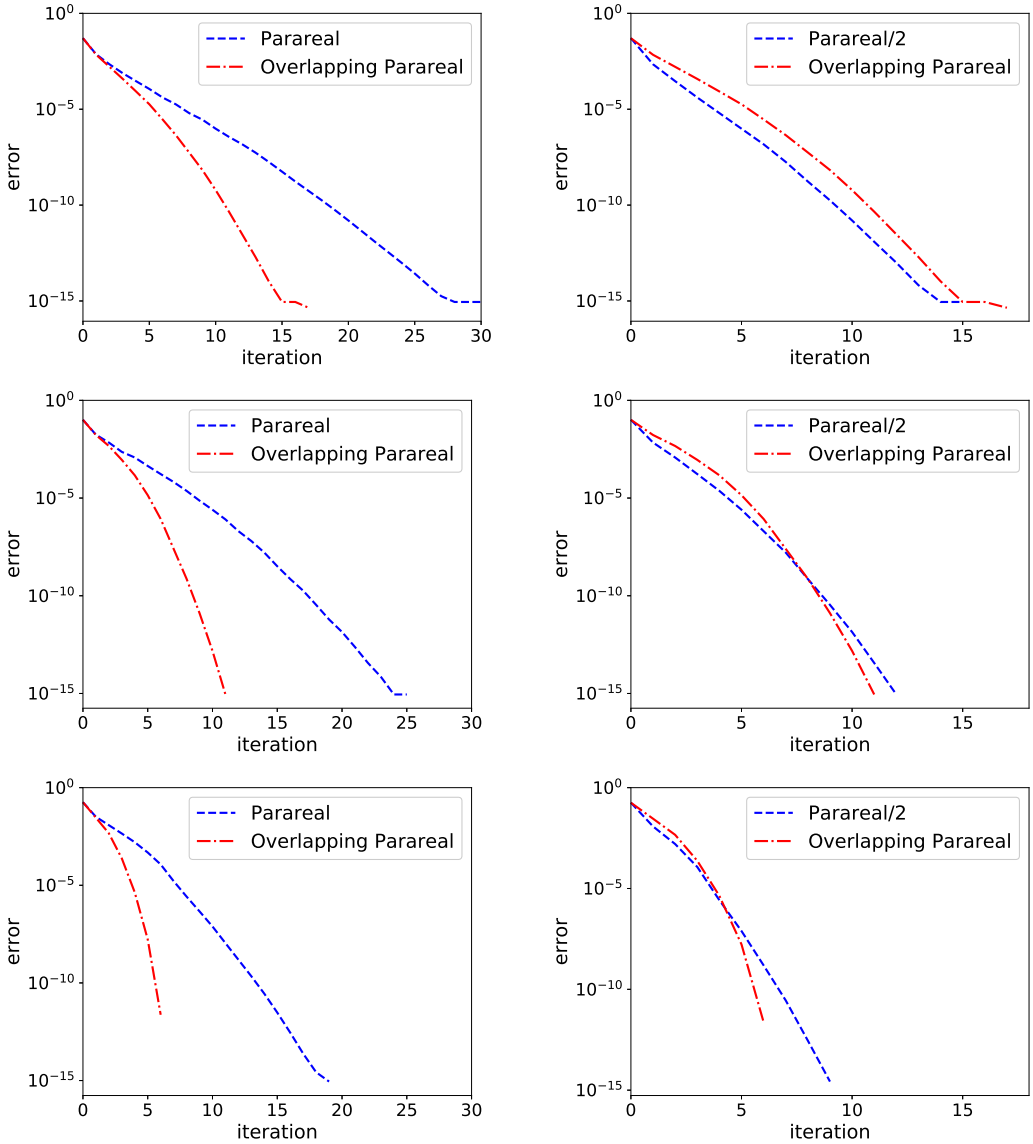

Fig. 8 Burger's equation case using 128 processors in the top row, 64 in the middle row, and 32 in the bottom row.

composition Methods in Science and Engineering XXI, pp. 275–283. Springer (2014)
13. Gander, M.J., Jiang, Y.L., Song, B., Zhang, H.: Analysis of two parareal algorithms for time-periodic problems. SIAM Journal on Scientific Computing **35**(5), A2393–2415 (2013)
14. Gander, M.J., Loneland, A.: SHEM: An optimal coarse space for RAS and its multiscale approximation. Domain Decomposition Methods in Science and Engineering XXIII, Springer (2016)
15. Gander, M.J., Loneland, A., Rahman, T.: Analysis of a new harmonically enriched multiscale coarse space for domain decomposition methods. arXiv preprint arXiv:1512.05285 (2015)
16. Gander, M.J., Vandewalle, S.: Analysis of the parareal time-parallel time-integration method. SIAM Journal on Scientific Computing **29**(2), 556–578 (2007)
17. Lions, J.L., Maday, Y., Turinici, G.: A parareal in time discretization of PDEs. C.R. Acad. Sci. Paris, Serie I **332**, 661–668 (2001)
18. Ries, M., Trottenberg, U., Winter, G.: A note on MGR methods. Linear Algebra and its Applications **49**, 1–26 (1983)
19. Ruge, J.W., Stüben, K.: Algebraic multigrid. Multigrid methods **3**(13), 73–130 (1987)
20. Ruprecht, D., Speck, R., Krause, R.: Parareal for diffusion problems with space-and time-dependent coefficients. In: Domain Decomposition Methods in Science and Engineering XXII, pp. 371–378. Springer (2016)
21. Schröder, J.: Zur Lösung von Potentialaufgaben mit Hilfe des Differenzenverfahrens. ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik **34**(7), 241–253 (1954)
22. Stüben, K.: Algebraic multigrid (AMG): experiences and comparisons. Applied mathematics and computation **13**(3), 419–451 (1983)