SIAM J. Sci. Comput.
Vol. 45, No. 5, pp. A2275–A2303

A UNIFIED ANALYSIS FRAMEWORK FOR ITERATIVE PARALLEL-IN-TIME ALGORITHMS[*]

MARTIN J. GANDER[†], THIBAUT LUNET[‡], DANIEL RUPRECHT[‡], AND ROBERT SPECK[§]

Abstract. Parallel-in-time integration has been the focus of intensive research efforts over the past two decades due to the advent of massively parallel computer architectures and the scaling limits of purely spatial parallelization. Various iterative parallel-in-time algorithms have been proposed, like Parareal, PFASST, MGRIT, and Space-Time Multi-Grid (STMG). These methods have been described using different notation, and the convergence estimates that are available are difficult to compare. We describe Parareal, PFASST, MGRIT, and STMG for the Dahlquist model problem using a common notation and give precise convergence estimates using generating functions. This allows us, for the first time, to directly compare their convergence. We prove that all four methods eventually converge superlinearly, and we also compare them numerically. The generating function framework provides further opportunities to explore and analyze existing and new methods.

Key words. parallel-in-time methods, PinT methods, Parareal, PFASST, MGRIT, space-time multigrid, STMG, generating functions, convergence estimates

MSC codes. 65R20, 45L05, 65L20

DOI. 10.1137/22M1487163

1. Introduction.

The efficient numerical solution of time-dependent ordinary and partial differential equations (ODEs/PDEs) has always been an important research subject in computational science and engineering. Nowadays, with high-performance computing platforms providing more and more processors whose individual processing speeds are no longer increasing, the capacity of algorithms to run concurrently becomes important. As classical parallelization algorithms start to reach their intrinsic efficiency limits, substantial research efforts have been invested to find new parallelization approaches that can translate the computing power of modern many-core high-performance computing architectures into faster simulations.

For time-dependent problems, the idea to parallelize across the time direction has gained renewed attention in the last two decades.[1] Various algorithms have been developed; for overviews see the papers by Gander [18] and Ong and Schroder [42]. Four iterative algorithms have received significant attention, namely Parareal [36] (474 citations since 2001),[2] the *Parallel Full Approximation Scheme in Space*

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section March 28, 2022; accepted for publication (in revised form) April 14, 2023; published electronically September 21, 2023.
https://doi.org/10.1137/22M1487163
Funding: This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement 955701. The JU receives support from the European Union's Horizon 2020 research & innovation program and from Belgium, France, Germany, and Switzerland. This project also received funding from the German Federal Ministry of Education and Research (BMBF), grant 16HPC048.

[†]University of Geneva, 1211 Geneva 4, Switzerland (Martin.Gander@unige.ch).
[‡]Hamburg University of Technology, 21073 Hamburg, Germany (thibaut.lunet@tuhh.de, ruprecht@tuhh.de).
[§]Forschungszentrum Jülich GmbH, 52428 Jülich, Germany (r.speck@fz-juelich.de).
[1]See also https://www.parallel-in-time.org.
[2]Number of citations since publication, according to Google Scholar in April 2023.

and Time (PFASST) [11] (254 citations since 2012), *Multi-Grid Reduction in Time* (MGRIT) [16, 14] (287 citations since 2014), and a specific form of *Space-Time Multi-Grid* (STMG) [25] (140 citations since 2016). Other algorithms have been proposed, e.g., the *Parallel (or* PARAREAL*) Implicit Time integration Algorithm* (PITA) [15] (275 citations since 2003), which is very similar to PARAREAL, the diagonalization technique [39] (63 citations since 2008), *Revisionist Integral Deferred Corrections* (RIDC) [6] (114 citations since 2010), PARAEXP [20] (103 citations since 2013), and *parallel Rational approximation of eEXponential Integrators* (REXI) [47] (28 citations since 2018).

PARAREAL, PFASST, MGRIT, and STMG have all been benchmarked for large-scale problems using large numbers of cores of high-performance computing systems [33, 35, 38, 49]. They cast the solution process in time as a large linear or nonlinear system which is solved by iterating on all time steps simultaneously. Since parallel performance is strongly linked to the rate of convergence, understanding convergence mechanisms and obtaining reliable error bounds for these iterative parallel-in-time (PinT) methods is crucial. Individual analyses exist for PARAREAL [2, 21, 26, 44, 50], MGRIT [8, 32, 48], PFASST [3, 4], and STMG [25]. There are also a few combined analyses showing equivalences between PARAREAL and MGRIT [14, 22] or connections between MGRIT and PFASST [40]. However, no systematic comparison of convergence behavior, let alone efficiencies, between these methods exists.

There are at least three obstacles to comparing these four methods: first, there is no common formalism or notation to describe them; second, the existing analyses use very different techniques to obtain convergence bounds; third, the algorithms can be applied to many different problems in different ways with many tunable parameters, all of which affect performance [28]. Our main contribution is to address, at least for the Dahlquist test problem, the first two problems by proposing a common formalism to rigorously describe PARAREAL, PFASST, MGRIT,[3] and the Time Multi-Grid (TMG) component[4] of STMG using the same notation. Then, we obtain comparable error bounds for all four methods by using the generating function method (GFM) [34]. GFM has been used to analyze PARAREAL [21] and was used to relate PARAREAL and MGRIT [22]. However, our use of GFM to derive common convergence bounds across multiple algorithms is novel, as is the presented unified framework. When coupled with a predictive model for computational cost, this GFM framework could eventually be extended to a model to compare parallel performance of different algorithms, but this is left for future work.

Our manuscript is organized as follows. In section 2, we introduce the GFM framework; in particular, in section 2.1, we give three definitions (time block, block variable, and block operator) used to build the GFM framework and provide some examples using classical time integration methods. Section 2.2 contains the central definition of a *block iteration* and again examples. In section 2.3, we state the main theoretical results and error bounds, and the next sections contain how existing algorithms from the PinT literature can be expressed in the GFM framework: PARAREAL in section 3, TMG in section 4, and PFASST in section 5. Finally, we compare in section 6 all methods using the GFM framework. Conclusions and an outlook are given in section 7.

[3]We do not analyze in detail MGRIT with FCF relaxation, only with F relaxation, in which case the two-level variant is equivalent to PARAREAL. Our framework could, however, be extended to include FCF relaxation; see Remark 3.1.

[4]Since we focus only on the time dimension, the spatial component of STMG is left out.

2. The generating function method. We consider the Dahlquist equation

$$\frac{du}{dt} = \lambda u, \quad \lambda \in \mathbb{C}, \quad t \in (0,T], \quad u(0) = u_0 \in \mathbb{C}. \tag{2.1}$$

The complex parameter λ allows us to emulate problems of parabolic ($\lambda < 0$), hyperbolic (λ imaginary), and mixed type.

2.1. Blocks, block variables, and block operators. We decompose the time interval $[0,T]$ into N time subintervals $[t_n, t_{n+1}]$ of uniform size Δt with $n \in \{0,\dots, N-1\}$.

Definition 2.1 (time block). *A* time block *(or simply block) denotes the discretization of a time subinterval $[t_n, t_{n+1}]$ using $M > 0$ grid points,*

$$\tau_{n,m} = t_n + \Delta t \tau_m, \quad m \in \{1,\dots,M\}, \tag{2.2}$$

where the $\tau_m \in [0,1]$ denote normalized grid points in time used for all blocks.

We choose the name "block" in order to have a generic name for the internal steps inside each time subinterval. A block could be several time steps of a classical time-stepping scheme (e.g., Runge–Kutta; cf. section 2.1.1), the quadrature nodes of a collocation method (cf. section 2.1.2), or a combination of both. But in every case, a block represents the time domain that is associated to one computational process of the time parallelization. A block can also collapse by setting $M := 1$ and $\tau_1 := 1$, so that we retrieve a standard uniform time discretization with time step Δt. The additional structure provided by blocks will be useful when describing and analyzing two-level methods which use different numbers of grid points per block for each level; cf. section 4.2.

Definition 2.2 (block variable). *A* block variable *is a vector*

$$\boldsymbol{u}_n = [u_{n,1}, u_{n,2}, \dots, u_{n,M}]^T, \tag{2.3}$$

where $u_{n,m}$ is an approximation of $u(\tau_{n,m})$ on the time block for the time subinterval $[t_n, t_{n+1}]$. For $M = 1$, $\boldsymbol{u}_n$ reduces to a scalar approximation of $u(\tau_{n,M}) \equiv u(t_{n+1})$.

Some iterative PinT methods like Parareal (see section 3) use values defined at the interfaces between subintervals $[t_n, t_{n+1}]$. Other algorithms, like PFASST (see section 5), update solution values in the interior of blocks. In the first case, the block variable is the right interface value with $M = 1$ and thus $\tau_1 = 1$. In the second case, it consists of *volume* values in the time block $[t_n, t_{n+1}]$ with $M > 1$. In both cases, PinT algorithms can be defined as *iterative processes updating the block variables.*

Remark 2.3. While the adjective "time" is natural for evolution problems, PinT algorithms can also be applied to recurrence relations in different contexts like deep learning [29] or when computing Gauss quadrature formulas [24]. Therefore, we will not systematically mention "time" when talking about blocks and block variables.

Definition 2.4 (block operators). *We denote as block operators the two linear functions $\boldsymbol{\phi} : \mathbb{C}^M \to \mathbb{C}^M$ and $\boldsymbol{\chi} : \mathbb{C}^M \to \mathbb{C}^M$ for which the block variables of a numerical solution of* (2.1) *satisfy*

$$\boldsymbol{\phi}(\boldsymbol{u}_1) = \boldsymbol{\chi}(u_0 \mathbb{1}), \quad \boldsymbol{\phi}(\boldsymbol{u}_{n+1}) = \boldsymbol{\chi}(\boldsymbol{u}_n), \quad n = 1, 2, \dots, N-1, \tag{2.4}$$

with $\mathbb{1} := [1,\dots,1]^T$. The time integration operator $\boldsymbol{\phi}$ is bijective and $\boldsymbol{\chi}$ is a transmission operator. The time propagator updating $\boldsymbol{u}_n$ to $\boldsymbol{u}_{n+1}$ is given by

$$\boldsymbol{\psi} := \boldsymbol{\phi}^{-1} \boldsymbol{\chi}. \tag{2.5}$$

2.1.1. Example with Runge–Kutta methods. Consider numerical integration of (2.1) with a Runge–Kutta method with stability function

$$R(z) \approx e^z. \tag{2.6}$$

Using ℓ equidistant time steps per block, there are two natural ways to write the method using block operators:

1. *The volume formulation*: set $M := \ell$ with $\tau_m := m/M$, $m = 1, \ldots, M$. Setting $r := R(\lambda\Delta t/\ell)^{-1}$, the block operators are the $M \times M$ sparse matrices

$$\boldsymbol{\phi} := \begin{pmatrix} r & & \\ -1 & r & \\ & \ddots & \ddots \end{pmatrix}, \quad \boldsymbol{\chi} := \begin{pmatrix} 0 & \ldots & 0 & 1 \\ \vdots & & \vdots & 0 \\ \vdots & & \vdots & \vdots \end{pmatrix}. \tag{2.7}$$

2. *The interface formulation*: set $M := 1$ so that

$$\boldsymbol{\phi} := R(\lambda\Delta t/\ell)^{-\ell}, \quad \boldsymbol{\chi} := 1. \tag{2.8}$$

2.1.2. Example with collocation methods. Collocation methods are special implicit Runge–Kutta methods [52, Chap. 4, sect. 4] and instrumental when defining PFASST in section 5. We show their representation with block operators. Starting from the Picard formulation for (2.1) in one time subinterval $[t_n, t_{n+1}]$,

$$u(t) = u(t_n) + \int_{t_n}^{t} \lambda u(\tau) d\tau, \tag{2.9}$$

we choose a quadrature rule to approximate the integral. We consider only Lobatto or Radau-II type quadrature nodes where the last quadrature node coincides with the right subinterval boundary. This gives us quadrature nodes for each subinterval that form the block discretization points $\tau_{n,m}$ of Definition 2.1, with $\tau_M = 1$. We approximate the solution $u(\tau_{n,m})$ at each node by

$$u_{n,m} = u_{n,0} + \lambda\Delta t \sum_{j=1}^{M} q_{m,j} u_{n,j} \quad \text{with} \quad q_{m,j} := \int_0^{\tau_m} l_j(s) ds, \tag{2.10}$$

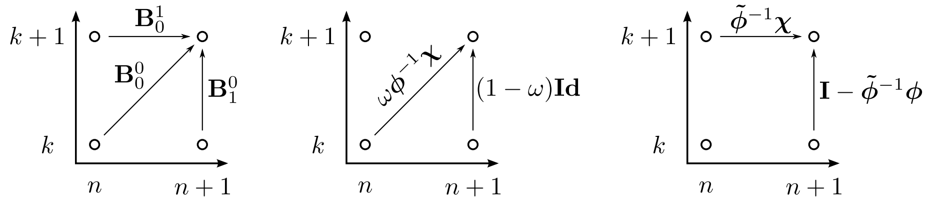
where l_j are the Lagrange polynomials associated with the nodes τ_m. Combining all the nodal values, we form the block variable $\boldsymbol{u}_n$, which satisfies the linear system

$$(\mathbf{I} - \mathbf{Q})\boldsymbol{u}_n = \begin{pmatrix} u_{n,0} \\ \vdots \\ u_{n,0} \end{pmatrix} = \begin{bmatrix} 0 & \ldots & 0 & 1 \\ \vdots & & \vdots & \vdots \\ 0 & \ldots & 0 & 1 \end{bmatrix} \boldsymbol{u}_{n-1} =: \mathbf{H}\boldsymbol{u}_{n-1}, \tag{2.11}$$

with the quadrature matrix $\mathbf{Q} := \lambda\Delta t(q_{m,j})$, $\mathbf{I}$ the identity matrix, and $\mathbf{H}$ sometimes called the transfer matrix that copies the last value of the previous time block to obtain the initial value $u_{n,0}$ of the current block.[5] The integration and transfer block operators from Definition 2.4 then become[6] $\boldsymbol{\phi} := (\mathbf{I} - \mathbf{Q})$, $\boldsymbol{\chi} := \mathbf{H}$.

[5]This specific form of the matrix $\mathbf{H}$ comes from the use of Lobatto or Radau-II rules, which treat the right interface of the time subinterval as a node. A similar description can also be obtained for Radau-I or Gauss-type quadrature rules that do not use the right boundary as node, but we omit it for the sake of simplicity.

[6]The notation $\mathbf{H}$ is specific to SDC and collocation methods (see, e.g., [3]), while the $\boldsymbol{\chi}$ notation from the GFM framework is generic for arbitrary time integration methods.



$k+1$
k
n
$n+1$
$\mathbf{B}_0^1$
$\mathbf{B}_0^0$
$\mathbf{B}_1^0$
$\omega\tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi}$
$(1-\omega)\mathbf{Id}$
$\tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi}$
$\mathbf{I}-\tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\phi}$

FIG. 1. *kn-graphs for a generic primary block iteration (left), damped Block Jacobi (middle), and Approximate Block Gauss–Seidel (right).*

2.2. Block iteration. Having defined the block operators for our problem, we write the numerical approximation (2.4) of (2.1) as the *all-at-once global problem*

$$\mathbf{A}\boldsymbol{u} := \begin{pmatrix} \boldsymbol{\phi} & & & \\ -\boldsymbol{\chi} & \boldsymbol{\phi} & & \\ & \ddots & \ddots & \\ & & -\boldsymbol{\chi} & \boldsymbol{\phi} \end{pmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{\chi}(u_0 \mathbb{1}) \\ 0 \\ \vdots \\ 0 \end{bmatrix} =: \boldsymbol{f}. \tag{2.12}$$

Iterative PinT algorithms solve (2.12) by updating a vector $\boldsymbol{u}^k = [\boldsymbol{u}_1^k, \ldots, \boldsymbol{u}_N^k]^T$ to $\boldsymbol{u}^{k+1}$ until some stopping criterion is satisfied. If the global iteration can be written as a local update for each block variable separately, we call the local update formula a block iteration.

DEFINITION 2.5 (primary block iteration). *A primary block iteration is an updating formula for* $n \geq 0$ *of the form*

$$\boldsymbol{u}_{n+1}^{k+1} = \mathbf{B}_1^0(\boldsymbol{u}_{n+1}^k) + \mathbf{B}_0^1\left(\boldsymbol{u}_n^{k+1}\right) + \mathbf{B}_0^0\left(\boldsymbol{u}_n^k\right), \quad \boldsymbol{u}_0^k = u_0\mathbb{1} \quad \forall k \in \mathbb{N}, \tag{2.13}$$

where $\mathbf{B}_1^0$, $\mathbf{B}_0^1$, *and* $\mathbf{B}_0^0$ *are linear operators from* $\mathbb{C}^M$ *to* $\mathbb{C}^M$ *that satisfy the consistency condition*[7]

$$(\mathbf{B}_1^0 - \mathbf{I})\boldsymbol{\psi} + \mathbf{B}_0^1 + \mathbf{B}_0^0 = 0 \tag{2.14}$$

with $\boldsymbol{\psi}$ *defined in* (2.5).

Note that a block iteration is always associated with an all-at-once global problem, and the primary block iteration (2.13) should converge to the solution of (2.12).

Figure 1 (left) shows a graphical representation of a primary block iteration using a *kn-graph* to represent the dependencies of $\boldsymbol{u}_{n+1}^{k+1}$ on the other block variables. The x-axis represents the block index n (time), and the y-axis represents the iteration index k. Arrows show dependencies from previous n or k indices and can only go from left to right and/or from bottom to top. For the primary block iteration, we consider only dependencies from the previous block n and iterate k for $\boldsymbol{u}_{n+1}^{k+1}$.

More general block iterations can also be considered for specific iterative PinT methods, e.g., MGRIT with FCF-relaxation (see Remark 3.1). Other algorithms also consist of combinations of two or more block iterations, for example, STMG (cf. section 4) or PFASST (cf. section 5). But we show in those sections that we can reduce those combinations into a single primary block iteration, hence we focus here mostly on primary block iterations to introduce our analysis framework.

We next describe the Block Jacobi relaxation (section 2.2.1) and the approximate Block Gauss–Seidel iteration (section 2.2.2), which are key components used to describe iterative PinT methods.

[7]Condition (2.14) is necessary for the block iteration to have the correct fixed point.

2.2.1. Block Jacobi relaxation. A damped Block Jacobi iteration for the global problem (2.12) can be written as

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \omega \mathbf{D}^{-1}(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k), \tag{2.15}$$

where $\mathbf{D}$ is a block diagonal matrix constructed with the integration operator $\boldsymbol{\phi}$, and $\omega > 0$ is a relaxation parameter. For $n > 0$, the corresponding block formulation is

$$\boldsymbol{u}_{n+1}^{k+1} = (1-\omega)\boldsymbol{u}_{n+1}^k + \omega \boldsymbol{\phi}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^k, \tag{2.16}$$

which is a primary block iteration with $\mathbf{B}_0^1 = 0$. Its kn-graph is shown in Figure 1 (middle). The consistency condition (2.14) is satisfied, since

$$((1-\omega)\mathbf{I} - \mathbf{I})\boldsymbol{\phi}^{-1}\boldsymbol{\chi} + 0 + \omega\boldsymbol{\phi}^{-1}\boldsymbol{\chi} = 0. \tag{2.17}$$

Note that selecting $\omega = 1$ simplifies the block iteration to

$$\boldsymbol{u}_{n+1}^{k+1} = \boldsymbol{\phi}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^k. \tag{2.18}$$

2.2.2. Approximate Block Gauss–Seidel iteration. Let us consider a Block Gauss–Seidel type preconditioned iteration for the global problem (2.12),

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \mathbf{P}_{GS}^{-1}(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k), \quad \mathbf{P}_{GS} = \begin{bmatrix} \tilde{\boldsymbol{\phi}} & & \\ -\boldsymbol{\chi} & \tilde{\boldsymbol{\phi}} & \\ & \ddots & \ddots \end{bmatrix}, \tag{2.19}$$

where the block operator $\tilde{\boldsymbol{\phi}}$ corresponds to an approximation of $\boldsymbol{\phi}$. This approximation can be based on time-step coarsening, but could also use other approaches, e.g., a lower order time integration method. In general, $\tilde{\boldsymbol{\phi}}$ must be cheaper than $\boldsymbol{\phi}$, but it is also less accurate. Subtracting $\boldsymbol{u}^k$ in (2.19) and multiplying by $\mathbf{P}_{GS}$ yields the block iteration of this *Approximate Block Gauss–Seidel* (ABGS),

$$\boldsymbol{u}_{n+1}^{k+1} = \left[\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\phi}\right]\boldsymbol{u}_{n+1}^k + \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^{k+1}. \tag{2.20}$$

Its kn-graph is shown in Figure 1 (right). Note that a standard Block Gauss–Seidel iteration for (2.12) (i.e., with $\tilde{\boldsymbol{\phi}} = \boldsymbol{\phi}$) is actually a direct solver, the iteration converges in one step by integrating all blocks with $\boldsymbol{\phi}$ sequentially, and its block iteration is simply

$$\boldsymbol{u}_{n+1}^{k+1} = \boldsymbol{\phi}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^{k+1}. \tag{2.21}$$

2.3. Generating function and error bound for a block iteration. Before giving a generic expression for the error bound of the primary block iteration (2.13) using the GFM framework, we first need a definition and a preliminary result. The primary block iteration (2.13) is defined for each block index $n \geq 0$, thus we can define the following.

DEFINITION 2.6 (generating function). *The generating function associated with the primary block iteration* (2.13) *is the power series*

$$\rho_k(\zeta) := \sum_{n=0}^{\infty} e_{n+1}^k \zeta^{n+1}, \tag{2.22}$$

where $e_{n+1}^k := \left\|\boldsymbol{u}_{n+1}^k - \boldsymbol{u}_{n+1}\right\|$ *is the difference between the* k^{th} *iterate* $\boldsymbol{u}_{n+1}^k$ *and the exact solution* $\boldsymbol{u}_{n+1}$ *for one block of* (2.4) *in some norm on* $\mathbb{C}^M$.

Since the analysis works in any norm, we do not specify a particular one here. In the numerical examples we use the L^∞ norm on $\mathbb{C}^M$.

LEMMA 2.7. *The generating function for the primary block iteration* (2.13) *satisfies*

$$\rho_{k+1}(\zeta) \le \frac{\gamma + \alpha\zeta}{1-\beta\zeta}\rho_k(\zeta), \tag{2.23}$$

where $\alpha := \left\|\mathbf{B}_0^0\right\|$, $\beta := \left\|\mathbf{B}_0^1\right\|$, $\gamma := \left\|\mathbf{B}_1^0\right\|$, *and the operator norm is induced by the chosen vector norm.*

Proof. We start from (2.13) and subtract the exact solution of (2.4),

$$\boldsymbol{u}_{n+1}^{k+1} - \boldsymbol{u}_{n+1} = \mathbf{B}_1^0(\boldsymbol{u}_{n+1}^k) + \mathbf{B}_0^1\left(\boldsymbol{u}_n^{k+1}\right) + \mathbf{B}_0^0\left(\boldsymbol{u}_n^k\right) - \boldsymbol{\psi}(\boldsymbol{u}_n). \tag{2.24}$$

Using the linearity of the block operators and the consistency condition (2.14) with $\boldsymbol{u}_n$, this simplifies to

$$\boldsymbol{u}_{n+1}^{k+1} - \boldsymbol{u}_{n+1} = \mathbf{B}_1^0(\boldsymbol{u}_{n+1}^k - \boldsymbol{u}_{n+1}) + \mathbf{B}_0^1\left(\boldsymbol{u}_n^{k+1} - \boldsymbol{u}_n\right) + \mathbf{B}_0^0\left(\boldsymbol{u}_n^k - \boldsymbol{u}_n\right). \tag{2.25}$$

We apply the norm and use the triangle inequality and the operator norms defined above to get the recurrence relation

$$e_{n+1}^{k+1} \le \gamma e_{n+1}^k + \beta e_n^{k+1} + \alpha e_n^k \tag{2.26}$$

for the error. We multiply this inequality by ζ^{n+1} and sum for $n \in \mathbb{N}$ to get

$$\sum_{n=0}^{\infty} e_{n+1}^{k+1}\zeta^{n+1} \le \gamma\sum_{n=0}^{\infty} e_{n+1}^{k}\zeta^{n+1} + \beta\sum_{n=0}^{\infty} e_{n}^{k+1}\zeta^{n+1} + \alpha\sum_{n=0}^{\infty} e_{n}^{k}\zeta^{n+1}. \tag{2.27}$$

Note that this is a formal power series expansion for ζ small in the sense of generating functions [34, section 1.2.9]. Using Definition 2.6 and that $e_0^k = 0$ for all k we find

$$\rho_{k+1}(\zeta) \le \gamma\rho_k(\zeta) + \beta\zeta\sum_{n=1}^{\infty} e_n^{k+1}\zeta^n + \alpha\zeta\sum_{n=1}^{\infty} e_n^k\zeta^n. \tag{2.28}$$

Shifting indices leads to

$$(1-\beta\zeta)\rho_{k+1}(\zeta) \le (\gamma + \alpha\zeta)\rho_k(\zeta) \tag{2.29}$$

and concludes the proof. ◻

THEOREM 2.8. *Consider the primary block iteration* (2.13) *and let*

$$\delta := \max_{n=1,\dots,N}\left\|\boldsymbol{u}_n^0 - \boldsymbol{u}_n\right\| \tag{2.30}$$

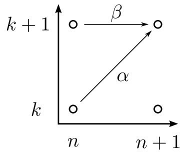
be the maximum error of the initial guess over all blocks. Then, using the notation of Lemma 2.7, *we have*

$$e_{n+1}^k \le \theta_{n+1}^k(\alpha,\beta,\gamma)\delta \tag{2.31}$$

for $k > 0$, *where* θ_{n+1}^k *is a bounding function defined as follows:*

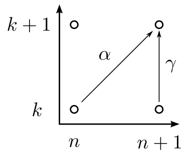
- *if only* $\gamma = 0$, *then*

$$\theta_{n+1}^k = \frac{\alpha^k}{(k-1)!}\sum_{i=0}^{n-k}\prod_{l=1}^{k-1}(i+l)\beta^i; \tag{2.32}$$



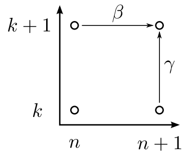
- *if* only $\beta = 0$*, then*

$$\theta_{n+1}^k = \begin{cases} (\gamma+\alpha)^k \text{ if } k \leq n, \\ \gamma^k \sum_{i=0}^{n} \binom{k}{i} \left(\frac{\alpha}{\gamma}\right)^i \text{ otherwise}; \end{cases} \tag{2.33}$$



- *if* only $\alpha = 0$*, then*

$$\theta_{n+1}^k = \frac{\gamma^k}{(k-1)!} \sum_{i=0}^{n} \prod_{l=1}^{k-1} (i+l)\beta^i; \tag{2.34}$$



- *if* neither α*, nor* β*, nor* γ *is zero, then*

$$\theta_{n+1}^k = \gamma^k \sum_{i=0}^{\min(n,k)} \sum_{l=0}^{n-i} \binom{k}{i} \binom{l+k-1}{l} \left(\frac{\alpha}{\gamma}\right)^i \beta^l. \tag{2.35}$$

We call any error bound obtained from one of these formulas a GFM-bound.

The proof uses Lemma 2.7 to bound the generating function at $k = 0$ by

$$\rho_0(\zeta) \leq \delta \sum_{n=0}^{\infty} \zeta^{n+1}, \tag{2.36}$$

which covers arbitrary initial guesses for defining starting values $\boldsymbol{u}_n^0$ for each block. For specific initial guesses, $\rho_0(\zeta)$ can be bounded differently [21, Proof of Thm. 1]. The error bound is then computed by coefficient identification after a power series expansion. The full rather technical proof can be found in Appendix A.

In the numerical examples shown below, we find that the estimate from Theorem 2.8 is not always sharp; cf. section 5.5.1. If the last time point of the blocks coincides with the right bound of the subinterval,[8] it is helpful to define the *interface error* at the right boundary point of the n^{th} block as

$$\bar{e}_{n+1}^k := |\bar{u}_{n+1}^k - \bar{u}_{n+1}|, \tag{2.37}$$

where $\bar{u}$ is the last element of the block variable $\boldsymbol{u}$. We then multiply (2.25) by $e_M^T = [0, \ldots, 0, 1]$ to get

$$e_M^T(\boldsymbol{u}_{n+1}^{k+1} - \boldsymbol{u}_{n+1}) = \boldsymbol{b}_1^0(\boldsymbol{u}_{n+1}^k - \boldsymbol{u}_{n+1}) + \boldsymbol{b}_0^1(\boldsymbol{u}_n^{k+1} - \boldsymbol{u}_n) + \boldsymbol{b}_0^0(\boldsymbol{u}_n^k - \boldsymbol{u}_n), \tag{2.38}$$

where $\boldsymbol{b}_i^j$ is the last row of the block operator $\mathbf{B}_i^j$. Taking the absolute value on both sides, we recognize the interface error $\bar{e}_{n+1}^{k+1}$ on the left-hand side. By neglecting the error from interior points and using the triangle inequality, we get the approximation[9]

$$\bar{e}_{n+1}^{k+1} \lessapprox \bar{\gamma}\bar{e}_{n+1}^k + \bar{\beta}\bar{e}_n^{k+1} + \bar{\alpha}\bar{e}_n^k, \tag{2.39}$$

where $\bar{\alpha} := |\bar{b}_0^0|$, $\bar{\beta} := |\bar{b}_1^0|$, $\bar{\gamma} := |\bar{b}_0^1|$.

[8]This is the case for all time-integration methods considered in this paper, even if this is not a necessary condition to use the GFM framework.

[9]For an interface block iteration ($M = 1, \tau_1 = 1$), (2.39) becomes a rigorous inequality and Corollary 2.9 thus becomes an upper bound.

COROLLARY 2.9 (interface error approximation). *Defining for the initial interface error the bound* $\bar{\delta} := \max_{n\in\{1,\dots,N\}} \left\|\bar{u}_n^0 - \bar{u}_n\right\|$, *we obtain for the interface error the approximation*

$$\bar{e}_{n+1}^k \lessapprox \bar{\theta}_{n+1}^k \bar{\delta}, \quad \bar{\theta}_{n+1}^k := \theta_{n+1}^k(\bar{\alpha}, \bar{\beta}, \bar{\gamma}), \tag{2.40}$$

with θ_{n+1}^k *defined in Theorem* 2.8.

Proof. The result follows as in the proof of Lemma 2.7 using approximate relations. ∎

Remark 2.10. For the general case, the error at the interface $\bar{e}_{n+1}^{k+1}$ *is not the same as* the error for the whole block e_{n+1}^{k+1}. Only a block discretization using a single point ($M = 1$) makes the two values identical. Furthermore, Corollary 2.9 is generally not an upper bound, but an approximation thereof.

3. Writing Parareal and MGRIT as block iterations.

3.1. Description of the algorithm. The PARAREAL algorithm introduced by Lions, Maday, and Turinici [36] corresponds to a block iteration update with scalar blocks ($M = 1$), and its convergence was analyzed in [26, 44]. We propose here a new description of PARAREAL in the scope of the GFM framework, which states that PARAREAL is simply a combination of two preconditioned iterations applied to the global problem (2.12), namely one block Jacobi relaxation without damping (section 2.2.1), followed by an ABGS iteration (section 2.2.2).

We denote by $\boldsymbol{u}^{k+1/2}$ the intermediate solution after the Block Jacobi step. Using (2.18) and (2.20), the two successive primary block iteration steps are

$$\boldsymbol{u}_{n+1}^{k+1/2} = \boldsymbol{\phi}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^k, \tag{3.1}$$

$$\boldsymbol{u}_{n+1}^{k+1} = \left[\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\phi}\right]\boldsymbol{u}_{n+1}^{k+1/2} + \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^{k+1}. \tag{3.2}$$

Combining both yields the primary block iteration

$$\boldsymbol{u}_{n+1}^{k+1} = \left[\boldsymbol{\phi}^{-1}\boldsymbol{\chi} - \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi}\right]\boldsymbol{u}_n^k + \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^{k+1}. \tag{3.3}$$

Now as stated in section 2.2.2, $\tilde{\boldsymbol{\phi}}$ is an approximation of the integration operator $\boldsymbol{\phi}$, which is cheaper to invert but less accurate.[10] In other words, if we define

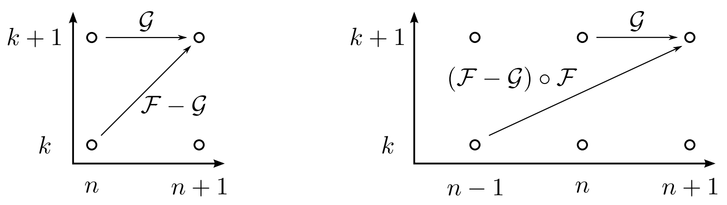
$$\mathcal{F} := \boldsymbol{\phi}^{-1}\boldsymbol{\chi}, \quad \mathcal{G} := \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi} \tag{3.4}$$

to be a fine and a coarse propagator on one block, then (3.3) becomes

$$\boldsymbol{u}_{n+1}^{k+1} = \mathcal{F}\boldsymbol{u}_n^k + \mathcal{G}\boldsymbol{u}_n^{k+1} - \mathcal{G}\boldsymbol{u}_n^k, \tag{3.5}$$

which is the PARAREAL update formula derived from the approximate Newton update in the multiple shooting approximation in [26]. Iteration (3.5) is a primary block iteration in the sense of Definition 2.5 with $\mathbf{B}_1^0 := 0$, $\mathbf{B}_0^1 := \mathcal{G}$, and $\mathbf{B}_0^0 := \mathcal{F} - \mathcal{G}$. Its kn-graph is shown in Figure 2 (left). The consistency condition (2.14) is satisfied, since $(0 - \mathbf{I})\mathcal{F} + \mathcal{G} + (\mathcal{F} - \mathcal{G}) = 0$. If we subtract $\boldsymbol{u}_{n+1}^k$ in (3.3), multiply both sides

[10] In the original paper [36], this approximation is done using larger time-steps, but many other types of approximations have been used since then in the literature.



$k+1$
k
n
$n+1$
$\mathcal{G}$
$\mathcal{F}-\mathcal{G}$
$n-1$
$(\mathcal{F}-\mathcal{G})\circ\mathcal{F}$

FIG. 2. *kn-graphs for* PARAREAL*/MGRIT with F-relaxation (left) and* MGRIT *with FCF-relaxation/*PARAREAL *with overlap (right).*

by $\boldsymbol{\phi}$, and rearrange terms, we can write PARAREAL as the preconditioned fixed-point iteration

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \mathbf{M}^{-1}(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k), \quad \mathbf{M} := \begin{pmatrix} \boldsymbol{\phi} & & \\ -\boldsymbol{\phi}\tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi} & \boldsymbol{\phi} & \\ & \ddots & \ddots \end{pmatrix}, \tag{3.6}$$

with iteration matrix $\mathbf{R}_{\text{PARAREAL}} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A}$.

Remark 3.1. It is known in the literature that PARAREAL is equivalent to a two-level MGRIT algorithm with F-relaxation [14, 22, 48]. In MGRIT, however, one also often uses FCF-relaxation, which is a combination of *two* nondamped ($\omega = 1$) Block Jacobi relaxation steps, followed by an ABGS step: denoting by $\boldsymbol{u}^{k+1/3}$ and $\boldsymbol{u}^{k+2/3}$ the intermediary Block Jacobi iterations, we obtain

$$\boldsymbol{u}_{n+1}^{k+1/3} = \boldsymbol{\phi}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^k, \tag{3.7}$$

$$\boldsymbol{u}_{n+1}^{k+2/3} = \boldsymbol{\phi}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^{k+1/3}, \tag{3.8}$$

$$\boldsymbol{u}_{n+1}^{k+1} = \left[\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\phi}\right]\boldsymbol{u}_{n+1}^{k+2/3} + \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^{k+1}. \tag{3.9}$$

Shifting the n index in the first Block Jacobi iteration, combining all of them and reusing the $\mathcal{F}$ and $\mathcal{G}$ notation then gives

$$\boldsymbol{u}_{n+1}^{k+1} = \mathbf{B}_{-1}^{0}(\boldsymbol{u}_{n-1}^{k}) + \mathbf{B}_0^1\left(\boldsymbol{u}_n^{k+1}\right), \quad \mathbf{B}_{-1}^0 = (\mathcal{F}-\mathcal{G})\mathcal{F},\ \mathbf{B}_0^1 = \mathcal{G}, \tag{3.10}$$

which is the update formula of PARAREAL with overlap, shown to be equivalent to MGRIT with FCF-relaxation [22, Thm. 4].[11]

This block iteration, whose kn-graph is represented in Figure 2 (right), not only links two successive block variables with time index $n+1$ and n, but also uses a block with time index $n-1$. It is not a primary block iteration in the sense of Definition 2.5 anymore. Although it can be analyzed using generating functions [22, Thm. 6], we focus on primary block iterations here and leave more complex block iterations like this one for future work.

3.2. Convergence analysis with GFM-bounds. In their convergence analysis of PARAREAL for nonlinear problems [21], the authors obtain a double recurrence of the form $e_{n+1}^{k+1} \leq \alpha e_n^k + \beta e_n^{k+1}$, where α and β come from Lipschitz constants and local

[11]It was shown in [22] that MGRIT with (FC)$^\nu$F-relaxation, where $\nu > 0$ is the number of additional FC-relaxations, is equivalent to an overlapping version of PARAREAL with ν overlaps. Generalizing our computations shows that those algorithms are equivalent to $(\nu - 1)$ nondamped Block Jacobi iterations followed by an ABGS step.

truncation error bounds. Using the same notation as in section 3.1, with $\alpha = \|\mathcal{F} - \mathcal{G}\|$ and $\beta = \|\mathcal{G}\|$, we find [21, Thm. 1] that

$$e_{n+1}^k \leq \delta \frac{\alpha^k}{k!} \bar{\beta}^{n-k} \prod_{l=1}^{k} (n+1-l), \quad \bar{\beta} = \max(1, \beta). \tag{3.11}$$

This is different from the GFM-bound

$$e_{n+1}^k \leq \delta \frac{\alpha^k}{(k-1)!} \sum_{i=0}^{n-k} \prod_{l=1}^{k-1} (i+l) \beta^i \tag{3.12}$$

we get when applying Theorem 2.8 with $\gamma = 0$ to the block iteration of Parareal. The difference stems from an approximation in the proof of [21, Thm. 1] which leads to the simpler and more explicit bound in (3.11). The two bounds are equal when $\beta = 1$, but for $\beta \neq 1$, the GFM-bound in (3.12) is sharper. To illustrate this, we use the interface formulation of section 2.1.1: we set $M := 1$, $\tau_1 := 1$ and use the block operators

$$\boldsymbol{\phi} := R(\lambda \Delta t/\ell)^{-\ell}, \quad \boldsymbol{\chi} := 1, \quad \tilde{\boldsymbol{\phi}} := R_\Delta(\lambda \Delta t/\ell_\Delta)^{-\ell_\Delta}. \tag{3.13}$$

We solve (2.1) for $\lambda \in \{i, -1\}$ with $t \in [0, 2\pi]$ and $u_0 = 1$, using $N := 10$ blocks, $\ell := 10$ fine time steps per block, the standard fourth order Runge–Kutta method for $\boldsymbol{\phi}$, and $\ell_\Delta = 2$ coarse time steps per block with Backward Euler for $\tilde{\boldsymbol{\phi}}$. Figure 3 shows the resulting error (dashed line) at the last time point, the original error bound (3.11), and the new bound (3.12). We also plot the linear bound obtained from the L^∞ norm of the iteration matrix $\mathbf{R}_{\text{Parareal}}$ defined just after (3.6). For both values of λ, the GFM-bounds coincide with the linear bound from $\mathbf{R}_{\text{Parareal}}$ for the first iteration, and the GFM-bound captures the superlinear contraction in later iterations. For $\lambda = i$, the old and new bounds are similar since β is close to 1. However, for $\lambda = -1$ where β is smaller than one, the new bound gives a sharper estimate of the error, and we can also see that the new bound captures well the transition from the linear to the superlinear convergence regime. On the left in Figure 3, Parareal seems to converge well for imaginary $\lambda = i$. This, however, should not be seen as a working example of Parareal for a hyperbolic type problem, but is rather the effect of the relatively good accuracy of the coarse solver using 20 points per wave length for one wavelength present in the solution time interval we consider. Denoting by ϵ_Δ the L_∞ error with respect to the exact solution, the accuracy of the coarse solver ($\epsilon_\Delta =$ 6.22e-01) allows the Parareal error to reach the fine solver error ($\epsilon_\Delta =$ 8.16e-07) in $K = 8$ iterations. Since the ideal parallel speedup of Parareal, neglecting the coarse solver cost, is bounded by $N/K = 1.25$ [1, sect. 4], this indicates, however, almost no speedup in practical applications (see also [28]). If we increase the coarse solver error, for instance by multiplying λ by a factor 4 to have now four times more wavelength in the domain, and only 12.5 points per wavelength resolution in the coarse solver, the convergence of Parareal deteriorates massively, as we can see in Figure 4 (left), while this is not the case for the purely negative real fourfold $\lambda = -4$.

This illustrates how Parareal has great convergence difficulties for hyperbolic problems, already well-documented in the literature; see, e.g., [17, 23]. This is analogous to the difficulties due to the pollution error and damping in multigrid methods when solving medium to high frequency associated time harmonic problems; see [10, 12, 13, 19, 7] and references therein.

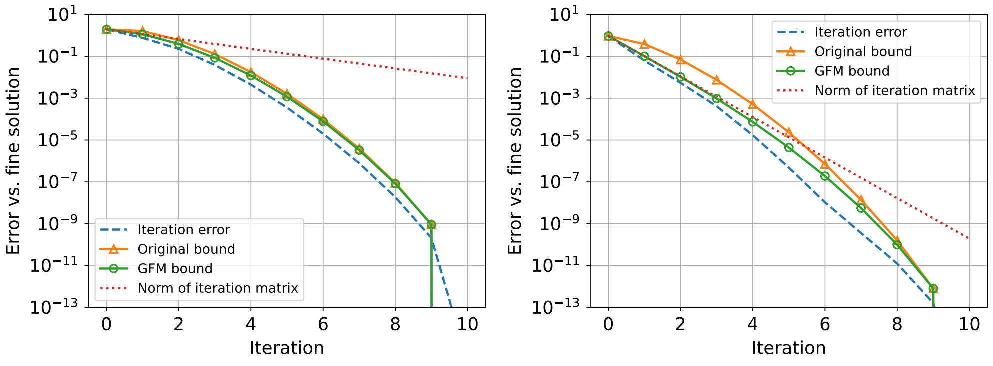


FIG. 3. *Error bounds for* PARAREAL *for* (2.1). *Left:* $\lambda = i$*; right:* $\lambda = -1$. *Note that for* $\lambda = i$*, the GFM-bound and the original one are almost identical.*

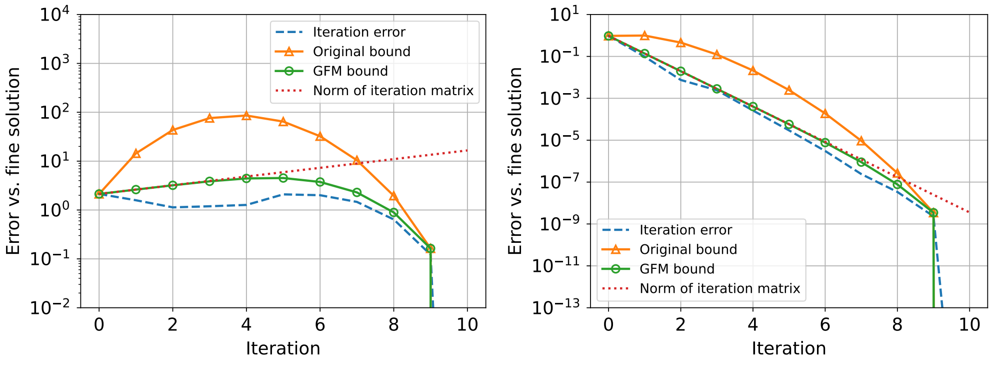


FIG. 4. *Error bounds for* PARAREAL *for* (2.1). *Left:* $\lambda = 4i$*; right:* $\lambda = -4$.

4. Writing two-level Time Multi-Grid as a block iteration. The idea of TMG goes back to the 1980s and 1990s [5, 30, 41]. Furthermore, not long after PARAREAL was introduced, it was shown to be equivalent to a TMG method, independently of the type of approximation used for the coarse solver [26]. This inspired the development of other time multilevel methods, in particular MGRIT [14]. However, PARAREAL and MGRIT are usually viewed as iterations acting on values located at the block interface, while TMG-based algorithms, in particular STMG [25], use an iteration updating volume values (i.e., all fine time points in the time domain). In this section, we focus on a generic description of TMG and show how to write its two-level form applied to the Dahlquist problem as block iteration. In particular, we will show in section 5 that PFASST can be expressed as a specific variant of TMG. The extension of this analysis to more levels and comparison with multilevel MGRIT is left for future work.

4.1. Definition of a coarse block problem for Time Multi-Grid. To build a coarse problem, we consider a coarsened version of the global problem (2.12), with an $\mathbf{A}_C$ matrix having $N \cdot M_C$ rows instead of $N \cdot M$ for $\mathbf{A}$. For each of the N blocks, let $(\tau_m^C)_{1\leq m\leq M_C}$ be the normalized M_C grid points[12] of a *coarser* block discretization, with $M_C < M$.

[12]Those do not need to be a subset of the fine block grid points, although they usually are in applications.

We can define a coarse block operator $\boldsymbol{\phi}_C$ by using the same time integration method as for $\boldsymbol{\phi}$ on every block, but with fewer time points. This is equivalent to geometric coarsening used for h-multigrid (or geometric multigrid [51]), e.g., when using one time step of a Runge–Kutta method between each time grid point. It can also be equivalent to spectral coarsening used for p-multigrid (or spectral element multigrid [43]), e.g., when one step of a collocation method on M points is used within each block (as for PFASST, see section 5.3).

We also consider the associated transmission operator $\boldsymbol{\chi}_C$ and denote by $\boldsymbol{u}_n^C$ the block variable on this coarse time block, which satisfies

$$\boldsymbol{\phi}_C(\boldsymbol{u}_1^C) = \boldsymbol{\chi}_C \mathbf{T}_F^C(u_0 \mathbb{1}), \quad \boldsymbol{\phi}_C \boldsymbol{u}_{n+1}^C = \boldsymbol{\chi}_C \boldsymbol{u}_n^C, \quad n = 1, 2, \dots, N-1. \tag{4.1}$$

Let $\boldsymbol{u}^C$ be the global coarse variable that solves

$$\mathbf{A}_C \boldsymbol{u}^C := \begin{pmatrix} \boldsymbol{\phi}_C & & & \\ -\boldsymbol{\chi}_C & \boldsymbol{\phi}_C & & \\ & \ddots & \ddots & \\ & & -\boldsymbol{\chi}_C & \boldsymbol{\phi}_C \end{pmatrix} \begin{bmatrix} \boldsymbol{u}_1^C \\ \boldsymbol{u}_2^C \\ \vdots \\ \boldsymbol{u}_N^C \end{bmatrix} = \begin{bmatrix} \boldsymbol{\chi}_C \mathbf{T}_F^C(u_0 \mathbb{1}) \\ 0 \\ \vdots \\ 0 \end{bmatrix} =: \boldsymbol{f}^{\boldsymbol{C}}. \tag{4.2}$$

$\mathbf{T}_F^C$ is a block restriction operator, i.e., a transfer matrix from a fine (F) to a coarse (C) block discretization. Similarly, we have a block prolongation operator $\mathbf{T}_C^F$, i.e., a transfer matrix from a coarse (C) to a fine (F) block discretization.

Remark 4.1. While both $\boldsymbol{\phi}_C$ and $\tilde{\boldsymbol{\phi}}$ are approximations of the fine operator $\boldsymbol{\phi}$, the main difference between $\boldsymbol{\phi}_C$ and $\tilde{\boldsymbol{\phi}}$ is the size of the vectors they can be applied to (M_C and M). Furthermore, $\boldsymbol{\phi}_C$ itself does need the transfer operators $\mathbf{T}_C^F$ and $\mathbf{T}_F^C$ to compute approximate values on the fine time points, while $\tilde{\boldsymbol{\phi}}$ alone is sufficient (even if it can hide some restriction and interpolation process within). However, the definition of a Coarse Grid Correction in the classical multigrid formalism needs this separation between transfer and coarse operators (see [51, sect. 2.2.2]), which limits the use of $\tilde{\boldsymbol{\phi}}$ and requires the introduction of $\boldsymbol{\phi}_C$.

4.2. Block iteration of a Coarse Grid Correction. Let us consider a standalone Coarse Grid Correction (CGC), without pre- or postsmoothing,[13] of a two-level multigrid iteration [31] applied to (2.12). One CGC step applied to (2.12) can be written as

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \bar{\mathbf{T}}_C^F \mathbf{A}_C^{-1} \bar{\mathbf{T}}_F^C (\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k), \tag{4.3}$$

where $\bar{\mathbf{T}}_C^F$ denotes the block diagonal matrix formed with $\mathbf{T}_C^F$ on the diagonal, and similarly for $\bar{\mathbf{T}}_F^C$. When splitting (4.3) into two steps,

$$\mathbf{A}_C \boldsymbol{d} = \bar{\mathbf{T}}_F^C (\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k), \tag{4.4}$$

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \bar{\mathbf{T}}_C^F \boldsymbol{d}, \tag{4.5}$$

the CGC term (or defect) $\boldsymbol{d}$ appears explicitly. Expanding the two steps for $n > 0$ into a block formulation and inverting $\boldsymbol{\phi}_C$ leads to

$$\boldsymbol{d}_{n+1} = \boldsymbol{\phi}_C^{-1} \mathbf{T}_F^C \boldsymbol{\chi} \boldsymbol{u}_n^k - \boldsymbol{\phi}_C^{-1} \mathbf{T}_F^C \boldsymbol{\phi} \boldsymbol{u}_{n+1}^k + \boldsymbol{\phi}_C^{-1} \boldsymbol{\chi}_C \boldsymbol{d}_n, \tag{4.6}$$

$$\boldsymbol{u}_{n+1}^{k+1} = \boldsymbol{u}_{n+1}^k + \mathbf{T}_C^F \boldsymbol{d}_{n+1}. \tag{4.7}$$

[13]The CGC is not convergent by itself without a smoother.

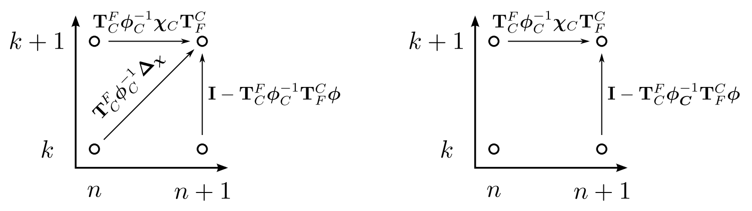

$k+1$
k
n
$n+1$
$\mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\boldsymbol{\chi}_C\mathbf{T}_F^C$
$\mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\boldsymbol{\Delta}_\chi$
$\mathbf{I}-\mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi}$
$k+1$
k
n
$n+1$
$\mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\boldsymbol{\chi}_C\mathbf{T}_F^C$
$\mathbf{I}-\mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi}$

FIG. 5. *kn-graphs for the CGC block iteration, with Assumption* 4.2 *only (left) and with both Assumptions* 4.2 *and* 4.3 *(right).*

Now we need the following simplifying assumption.

Assumption 4.2. Prolongation $\mathbf{T}_C^F$ followed by restriction $\mathbf{T}_F^C$ leaves the coarse block variables unchanged, i.e.,

$$\mathbf{T}_F^C\mathbf{T}_C^F = \mathbf{I}. \tag{4.8}$$

This condition is satisfied in many situations (e.g., restriction with standard injection on a coarse subset of the fine points, or polynomial interpolation with any possible coarse block discretization).[14] Using it in (4.7) for block index n yields

$$\boldsymbol{d}_n = \mathbf{T}_F^C\left(\boldsymbol{u}_n^{k+1} - \boldsymbol{u}_n^k\right). \tag{4.9}$$

Inserting $\boldsymbol{d}_n$ into (4.6) on the right and the resulting $\boldsymbol{d}_{n+1}$ into (4.7) leads to

$$\boldsymbol{u}_{n+1}^{k+1} = (\mathbf{I} - \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi})\boldsymbol{u}_{n+1}^k + \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\boldsymbol{\chi}_C\mathbf{T}_F^C\boldsymbol{u}_n^{k+1} + \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\Delta_\chi\boldsymbol{u}_n^k \tag{4.10}$$

with $\Delta_\chi := \mathbf{T}_F^C\boldsymbol{\chi} - \boldsymbol{\chi}_C\mathbf{T}_F^C$.

This is a primary block iteration in the sense of Definition 2.5, and we give its *kn*-graph in Figure 5 (left). We can simplify it further using a second assumption.

Assumption 4.3. We consider operators $\mathbf{T}_F^C$, $\boldsymbol{\chi}$ and $\boldsymbol{\chi}_C$ such that

$$\Delta_\chi = \mathbf{T}_F^C\boldsymbol{\chi} - \boldsymbol{\chi}_C\mathbf{T}_F^C = 0. \tag{4.11}$$

This holds for classical time-stepping methods when both left and right time subinterval boundaries are included in the block variables, or for collocation methods using Radau-II or Lobatto type nodes.

This last assumption is important to define PFASST (cf. section 5.3 and see Bolten, Moser, and Speck [3, Remark 1] for more details) and simplifies the analysis of TMG, as both methods use this block iteration. Then, (4.10) reduces to

$$\boldsymbol{u}_{n+1}^{k+1} = (\mathbf{I} - \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi})\boldsymbol{u}_{n+1}^k + \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}\boldsymbol{u}_n^{k+1}. \tag{4.12}$$

Again, this is a primary block iteration for which the *kn*-graph is given in Figure 5 (right). It satisfies the consistency condition[15] (2.14) since $((\mathbf{I} - \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi}) - \mathbf{I})\boldsymbol{\phi}^{-1}\boldsymbol{\chi} + \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi} = 0$.

[14]In some situations, e.g., when the transpose of linear interpolation is used for restriction (full-weighting), we do not get the identity in Assumption 4.2 but an invertible matrix. The same simplifications can be done, except one must take into account the inverse of $(\mathbf{T}_F^C\mathbf{T}_C^F)$.

[15]Note that the consistency condition is satisfied even without Assumption 4.3.

4.3. Two-level Time Multi-Grid. Gander and Neumüller introduced STMG for discontinuous Galerkin approximations in time [25], which leads to a similar system as (2.12). We describe the two-level approach for general time discretizations, following their multilevel description [25, sect. 3]. Consider a coarse problem defined as in section 4.2 and a damped Block Jacobi smoother as in section 2.2.1 with relaxation parameter ω. Then, a two-level TMG iteration requires the following steps, each corresponding to a block iteration:

1. ν_1 prerelaxation steps (2.15) with Block Jacobi smoother,
2. one CGC (4.3) inverting the coarse grid operators,
3. ν_2 postrelaxation steps (2.15) with the Block Jacobi smoother.

If we combine all these block iterations we do not obtain a primary block iteration but a more complex expression, of which the analysis is beyond the scope of this paper. However, a primary block iteration in the sense of Definition 2.5 is obtained when

- Assumption 4.3 holds, so that $\Delta_\chi = 0$,
- only one prerelaxation step is used, $\nu_1 = 1$,
- and no postrelaxation step is considered, $\nu_2 = 0$.

Then, the two-level iteration reduces to the two block updates from (2.16) and (4.12),

$$\boldsymbol{u}_{n+1}^{k+1/2} = (1-\omega)\boldsymbol{u}_{n+1}^{k} + \omega\boldsymbol{\phi}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^k, \tag{4.13}$$

$$\boldsymbol{u}_{n+1}^{k+1} = \left(\mathbf{I} - \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi}\right)\boldsymbol{u}_{n+1}^{k+1/2} + \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\boldsymbol{\chi}_C\mathbf{T}_F^C\boldsymbol{u}_n^{k+1}, \tag{4.14}$$

using $k+1/2$ as an intermediate index. Combining (4.13) and (4.14) leads to the primary block iteration

$$\boldsymbol{u}_{n+1}^{k+1} = \left(\mathbf{I} - \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi}\right)\left[(1-\omega)\boldsymbol{u}_{n+1}^{k} + \omega\boldsymbol{\phi}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^k\right] + \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\boldsymbol{\chi}_C\mathbf{T}_F^C\boldsymbol{u}_n^{k+1}. \tag{4.15}$$

If $\omega \neq 1$, all block operators in this primary block iteration are nonzero, and applying Theorem 2.8 leads to the error bound (2.35). Since the latter is similar to the one obtained for PFASST in section 5.5.2, we leave its comparison with numerical experiments to section 5. For $\omega = 1$ we get the simplified iteration

$$\boldsymbol{u}_{n+1}^{k+1} = \left(\boldsymbol{\phi}^{-1}\boldsymbol{\chi} - \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}\right)\boldsymbol{u}_n^k + \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\boldsymbol{\chi}_C\mathbf{T}_F^C\boldsymbol{u}_n^{k+1} \tag{4.16}$$

and the following result.

PROPOSITION 4.4. *Consider a CGC as in section* 4.2*, such that the prolongation and restriction operators (in time) satisfy Assumption* 4.2*. If Assumption* 4.3 *also holds and only one Block Jacobi prerelaxation step* (2.15) *with* $\omega = 1$ *is used before the CGC, then two-level* TMG *is equivalent to* PARAREAL*, where the coarse solver* $\mathcal{G}$ *uses the same time integrator as the fine solver* $\mathcal{F}$ *but with larger time steps, i.e.,*

$$\mathcal{G} := \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}. \tag{4.17}$$

This is a particular case of a result obtained before by Gander and Vandewalle [26, Theorem 3.1] but is presented here in the context of our GFM framework and the definition of PARAREAL given in section 3.1. In particular, it shows that the simplified two-grid iteration on (2.12) is equivalent to the preconditioned fixed-point iteration (3.6) of PARAREAL if some conditions are met and $\tilde{\boldsymbol{\phi}}^{-1} := \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C$ is used as the approximate integration operator.[16] However, the TMG iteration here updates also

[16]Note that, even if $\mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C$ is not invertible, this abuse of notation is possible as (3.6) requires an approximation of $\boldsymbol{\phi}^{-1}$ rather than an approximation of $\boldsymbol{\phi}$ itself.

the fine time point values, using $\mathbf{T}_C^F$ to interpolate the coarse values computed with $\boldsymbol{\phi}_C$, hence applying the Parareal update *to all volume values.* This is the only "difference" with the original definition of Parareal in [36], where the update is only applied to the interface value between blocks.

One key idea of STMG that we have not described yet is the block diagonal Jacobi smoother used for relaxation. Even if its diagonal blocks use a time integration operator identical to those of the fine problem (hence requiring the inversion of $\boldsymbol{\phi}$), their spatial part in STMG is approximated using one V-cycle multigrid iteration in space based on a pointwise smoother [25, sect. 4.3]. We do not cover this aspect in our description of TMG here, since we focus on time only, but describe in the next section a similar approach that is used for PFASST.

5. Writing PFASST as a block iteration.

PFASST is also based on a TMG approach using an approximate relaxation step, but the approximation of the Block Jacobi smoother is *done in time and not in space, in contrast to* STMG. In addition, the CGC in PFASST is also approximated, i.e., there is no direct solve on the coarse level to compute the CGC as in STMG. One PFASST iteration is therefore a combination of an *Approximate Block Jacobi* (ABJ) smoother (see section 5.2), followed by one (or more) ABGS iteration(s) of section 2.2.2 on the coarse level to approximate the CGC [11, sect. 3.2]. While we describe only the two-level variant, the algorithm can use more levels [11, 49]. The main component of PFASST is the approximation of the time integrator blocks using Spectral Deferred Correction (SDC) [9], from which its other key components (ABJ and ABGS) are built. Hence we first describe how SDC is used to define an ABGS iteration in section 5.1, then ABJ in section 5.2, and finally PFASST in section 5.3.

5.1. Approximate Block Gauss–Seidel with Spectral Deferred Correction.

SDC can be seen as a preconditioner when integrating the ODE problem (2.1) with collocation methods; see section 2.1.2. Consider the block operators

$$\boldsymbol{\phi} := (\mathbf{I} - \mathbf{Q}), \quad \boldsymbol{\chi} := \mathbf{H} \quad \Longrightarrow \quad (\mathbf{I} - \mathbf{Q})\boldsymbol{u}_{n+1} = \mathbf{H}\boldsymbol{u}_n. \tag{5.1}$$

SDC approximates the quadrature matrix $\mathbf{Q}$ by

$$\mathbf{Q}_\Delta = \lambda \Delta t \left(\tilde{q}_{m,j}\right), \quad \tilde{q}_{m,j} = \int_0^{\tau_m} \tilde{l}_j(s) ds, \tag{5.2}$$

where $\tilde{l}_j$ is an approximation of the Lagrange polynomial l_j. Usually, $\mathbf{Q}_\Delta$ is lower triangular [46, sect. 3] thus easy to invert.[17] This approximation is used to build the preconditioned iteration

$$\boldsymbol{u}_{n+1}^{k+1} = \boldsymbol{u}_{n+1}^k + [\mathbf{I} - \mathbf{Q}_\Delta]^{-1}\left(\mathbf{H}\boldsymbol{u}_n - (\mathbf{I} - \mathbf{Q})\boldsymbol{u}_{n+1}^k\right) \tag{5.3}$$

to solve (5.1), with $\boldsymbol{u}_{n+1}$ as unknown. We obtain the generic preconditioned iteration for one block,

$$\boldsymbol{u}_{n+1}^{k+1} = \left[\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\phi}\right]\boldsymbol{u}_{n+1}^k + \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n \quad \text{with} \quad \tilde{\boldsymbol{\phi}} := \mathbf{I} - \mathbf{Q}_\Delta. \tag{5.4}$$

This shows that SDC inverts the $\boldsymbol{\phi}$ operator approximately using $\tilde{\boldsymbol{\phi}}$ block by block to solve the global problem (2.12), i.e., it fixes an n in (5.4), iterates over k until

[17]The notation $\mathbf{Q}_\Delta$ was chosen instead of $\tilde{\mathbf{Q}}$ for consistency with the literature; cf. [46, 3, 4].

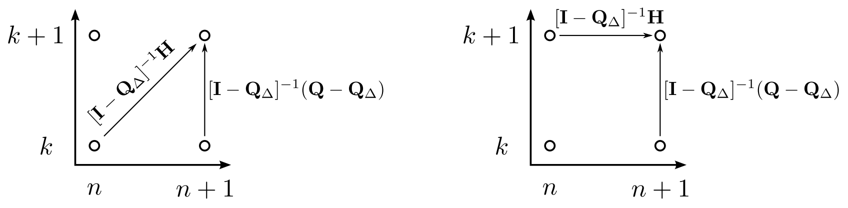

$k+1$
k
n
$n+1$
$[\mathbf{I}-\mathbf{Q}_\Delta]^{-1}\mathbf{H}$
$[\mathbf{I}-\mathbf{Q}_\Delta]^{-1}(\mathbf{Q}-\mathbf{Q}_\Delta)$
$k+1$
k
n
$n+1$
$[\mathbf{I}-\mathbf{Q}_\Delta]^{-1}\mathbf{H}$
$[\mathbf{I}-\mathbf{Q}_\Delta]^{-1}(\mathbf{Q}-\mathbf{Q}_\Delta)$

FIG. 6. *kn-graphs for Block Jacobi SDC (left) and Block Gauss–Seidel SDC (right).*

convergence, and then increments n by one. Hence SDC gives a natural way to define an approximate block integrator $\tilde{\boldsymbol{\phi}}$ to be used to build ABJ and ABGS iterations. Defining the ABGS iteration (2.19) of section 2.2.2 using the SDC block operators gives the block updating formula

$$\boldsymbol{u}_{n+1}^{k+1} = \boldsymbol{u}_{n+1}^{k} + [\mathbf{I}-\mathbf{Q}_\Delta]^{-1}\left(\mathbf{H}\boldsymbol{u}_n^{k+1} - (\mathbf{I}-\mathbf{Q})\boldsymbol{u}_{n+1}^{k}\right), \tag{5.5}$$

which we call *Block Gauss–Seidel SDC*, very similar to (5.3) except that we use the new iterate $\boldsymbol{u}_n^{k+1}$ and not the converged solution $\boldsymbol{u}_n$. This is a primary block iteration in the sense of Definition 2.5 with

$$\begin{aligned} &\mathbf{B}_1^0 := \mathbf{I} - [\mathbf{I}-\mathbf{Q}_\Delta]^{-1}(\mathbf{I}-\mathbf{Q}) = [\mathbf{I}-\mathbf{Q}_\Delta]^{-1}(\mathbf{Q}-\mathbf{Q}_\Delta),\\ &\mathbf{B}_0^0 := 0, \quad \mathbf{B}_0^1 := [\mathbf{I}-\mathbf{Q}_\Delta]^{-1}\mathbf{H}, \end{aligned} \tag{5.6}$$

and its kn-graph is shown in Figure 6 (right).

5.2. Approximate Block Jacobi with Spectral Deferred Correction. Here we solve the global problem (2.12) using a preconditioner that can be easily parallelized (Block Jacobi) and combine it with the approximation of the collocation operator $\boldsymbol{\phi}$ by $\tilde{\boldsymbol{\phi}}$ defined in (5.1) and (5.4). This leads to the *global* preconditioned iteration

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \mathbf{P}_{Jac}^{-1}(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k), \quad \mathbf{P}_{Jac} = \begin{bmatrix} \tilde{\boldsymbol{\phi}} & & \\ & \tilde{\boldsymbol{\phi}} & \\ & & \ddots \end{bmatrix}. \tag{5.7}$$

This is equivalent to the Block Jacobi relaxation in section 2.2.1 with $\omega = 1$, except that the block operator $\boldsymbol{\phi}$ is approximated by $\tilde{\boldsymbol{\phi}}$. Using the SDC block operators (5.1) gives the block updating formula

$$\boldsymbol{u}_{n+1}^{k+1} = \boldsymbol{u}_{n+1}^{k} + [\mathbf{I}-\mathbf{Q}_\Delta]^{-1}\left(\mathbf{H}\boldsymbol{u}_n^{k} - (\mathbf{I}-\mathbf{Q})\boldsymbol{u}_{n+1}^{k}\right), \tag{5.8}$$

which we call *Block Jacobi SDC*. This is a primary block iteration with

$$\begin{aligned} &\mathbf{B}_1^0 := \mathbf{I} - [\mathbf{I}-\mathbf{Q}_\Delta]^{-1}(\mathbf{I}-\mathbf{Q}) = [\mathbf{I}-\mathbf{Q}_\Delta]^{-1}(\mathbf{Q}-\mathbf{Q}_\Delta),\\ &\mathbf{B}_0^0 := [\mathbf{I}-\mathbf{Q}_\Delta]^{-1}\mathbf{H}, \quad \mathbf{B}_0^1 := 0. \end{aligned} \tag{5.9}$$

Its kn-graph is shown in Figure 6 (left). This block iteration can be written in the more generic form

$$\boldsymbol{u}_{n+1}^{k+1} = \left[\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\phi}\right]\boldsymbol{u}_{n+1}^{k} + \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^k. \tag{5.10}$$

This is similar to (5.4) except that we use the current iterate $\boldsymbol{u}_n^k$ from the previous block and not the converged solution $\boldsymbol{u}_n$. Note that $\boldsymbol{\phi}$ and $\tilde{\boldsymbol{\phi}}$ do not need to correspond to the SDC operators (5.1) and (5.4). This block iteration does not explicitly depend on the use of SDC, hence the name *Approximate Block Jacobi*.

5.3. PFASST. We now give a simplified description of PFASST [11] applied to the Dahlquist problem (2.1). In particular, this corresponds to doing only one SDC sweep on the coarse level. To write PFASST as a block iteration, we first build the coarse level as in section 4.2. From that we can form the $\tilde{\mathbf{Q}}$ quadrature matrix associated with the coarse nodes and the coarse matrix $\tilde{\mathbf{H}}$, as we would have done if we were using the collocation method of section 2.1.2 on the coarse nodes. This leads to the definition of the $\boldsymbol{\phi}_C$ and $\boldsymbol{\chi}_C$ operators for the coarse level, combined with the transfer operators $\mathbf{T}_F^C$ and $\mathbf{T}_C^F$, from which we can build the global matrices $\mathbf{A}_C$, $\bar{\mathbf{T}}_C^F$ and $\bar{\mathbf{T}}_F^C$; see section 4.2. Then we build the two-level PFASST iteration by defining a specific smoother and a modified CGC.

The smoother corresponds to a Block Jacobi SDC iteration (5.8) from section 5.2 to produce an intermediate solution

$$\boldsymbol{u}_{n+1}^{k+1/2} = [\mathbf{I} - \mathbf{Q}_\Delta]^{-1}(\mathbf{Q} - \mathbf{Q}_\Delta)\boldsymbol{u}_{n+1}^k + [\mathbf{I} - \mathbf{Q}_\Delta]^{-1}\mathbf{H}\boldsymbol{u}_n^k, \tag{5.11}$$

denoted with iteration index $k+1/2$. Using a CGC as in section 4.2 would provide the global update formula

$$\mathbf{A}_C\boldsymbol{d} = \bar{\mathbf{T}}_F^C(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^{k+1/2}), \tag{5.12}$$

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^{k+1/2} + \bar{\mathbf{T}}_C^F\boldsymbol{d}. \tag{5.13}$$

Instead of a direct solve with $\mathbf{A}_C$ to compute the defect $\boldsymbol{d}$, in PFASST one uses L Block Gauss–Seidel SDC iterations (or sweeps) to approximate it. Then (5.12) becomes

$$\tilde{\mathbf{P}}_{GS}\boldsymbol{d}^\ell = (\tilde{\mathbf{P}}_{GS} - \mathbf{A}_C)\boldsymbol{d}^{\ell-1} + \bar{\mathbf{T}}_F^C(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^{k+1/2}), \quad \boldsymbol{d}^0 = 0, \quad \ell \in \{1, \dots, L\}, \tag{5.14}$$

and reduces for one sweep only ($L = 1$) to

$$\tilde{\mathbf{P}}_{GS}\boldsymbol{d} = \bar{\mathbf{T}}_F^C(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^{k+1/2}), \quad \tilde{\mathbf{P}}_{GS} = \begin{bmatrix} \tilde{\boldsymbol{\phi}}_C & & \\ -\boldsymbol{\chi}_C & \tilde{\boldsymbol{\phi}}_C & \\ & \ddots & \ddots \end{bmatrix}. \tag{5.15}$$

Here $\tilde{\mathbf{P}}_{GS}$ correspond to the $\mathbf{P}_{GS}$ preconditioning matrix, but written on the coarse level using an SDC-based approximation $\tilde{\boldsymbol{\phi}}_C$ of the $\boldsymbol{\phi}_C$ coarse time integrator. Combined with the prolongation on the fine level (5.13), we get the modified CGC update

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^{k+1/2} + \bar{\mathbf{T}}_C^F\tilde{\mathbf{P}}_{GS}^{-1}\bar{\mathbf{T}}_F^C(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^{k+1/2}), \quad \tilde{\mathbf{P}}_{GS} = \begin{bmatrix} \tilde{\boldsymbol{\phi}}_C & & \\ -\boldsymbol{\chi}_C & \tilde{\boldsymbol{\phi}}_C & \\ & \ddots & \ddots \end{bmatrix}, \tag{5.16}$$

and together with (5.11) a two-level method for the global system (2.12) [4, sect. 2.2]. Note that this is the same iteration we obtained for the CGC in section 4.2, except that the coarse operator $\boldsymbol{\phi}_C$ has been replaced by $\tilde{\boldsymbol{\phi}}_C$. Assumption 4.3 holds, since using Lobatto or Radau-II nodes means $\mathbf{H}$ has the form (2.11), which implies

$$\Delta_\chi = \mathbf{T}_F^C\mathbf{H} - \tilde{\mathbf{H}}\mathbf{T}_F^C = 0. \tag{5.17}$$

Using similar computations as in section 4.2 and the block operators defined for collocation and SDC (cf. sections 2.1.2 and 5.1) we obtain the block iteration

$$\boldsymbol{u}_{n+1}^{k+1} = [\mathbf{I} - \mathbf{T}_C^F(\mathbf{I} - \tilde{\mathbf{Q}}_\Delta)^{-1}\mathbf{T}_F^C(\mathbf{I} - \mathbf{Q})]\boldsymbol{u}_{n+1}^{k+1/2} + \mathbf{T}_C^F(\mathbf{I} - \tilde{\mathbf{Q}}_\Delta)^{-1}\mathbf{T}_F^C\mathbf{H}\boldsymbol{u}_n^{k+1} \tag{5.18}$$

Table 1
Classification of two-level TMG *methods, depending on their smoother for fine-level relaxation and computation of the CGC.*

CGC \ Smoother	Block Jacobi ($\omega = 1$)	Approximate block Jacobi
Direct solver	TMG ($\omega = 1$)	TMG$_f$
ABGS (one step)	TMG$_c$	Two-level PFASST

by substitution into (4.12). Finally, the combination of the two gives

$$\begin{aligned} \boldsymbol{u}_{n+1}^{k+1} = & [\mathbf{I} - \mathbf{T}_C^F(\mathbf{I} - \tilde{\mathbf{Q}}_\Delta)^{-1}\mathbf{T}_F^C(\mathbf{I} - \mathbf{Q})][\mathbf{I} - \mathbf{Q}_\Delta]^{-1}(\mathbf{Q} - \mathbf{Q}_\Delta)\boldsymbol{u}_{n+1}^k \\ & + (\mathbf{I} - \mathbf{T}_C^F[\mathbf{I} - \tilde{\mathbf{Q}}_\Delta]^{-1}\mathbf{T}_F^C(\mathbf{I} - \mathbf{Q}))[\mathbf{I} - \mathbf{Q}_\Delta]^{-1}\mathbf{H}\boldsymbol{u}_n^k \\ & + \mathbf{T}_C^F(\mathbf{I} - \tilde{\mathbf{Q}}_\Delta)^{-1}\mathbf{T}_F^C\mathbf{H}\boldsymbol{u}_n^{k+1}. \end{aligned} \tag{5.19}$$

Using the generic formulation with the $\boldsymbol{\phi}$ operators gives[18]

$$\begin{aligned} \boldsymbol{u}_{n+1}^{k+1} = & [\mathbf{I} - \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi}](\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\phi})\boldsymbol{u}_{n+1}^k \\ & + (\mathbf{I} - \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi})\tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^k + \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}\boldsymbol{u}_n^{k+1}. \end{aligned} \tag{5.20}$$

This is again a primary block iteration in the sense of Definition 2.5, but in contrast to most previously described block iterations, all block operators are nonzero.

5.4. Similarities between PFASST, TMG, and Parareal. From the description in the previous section, it is clear that PFASST is very similar to TMG. While TMG uses a (damped) Block Jacobi smoother for prerelaxation and a direct solve for the CGC, PFASST uses instead an approximate Block Jacobi smoother and solves the CGC using one (or more) ABGS iterations on the coarse grid. This interpretation was obtained by Bolten, Moser, and Speck [3, Theorem 1] but is derived here using the GFM framework, and we summarize those differences in Table 1. Changing only the CGC or the smoother in TMG with $\omega = 1$ in contrast to both like in PFASST produces two further PinT algorithms. We call those TMG$_c$ (replacing the coarse solver by one step of ABGS) and TMG$_f$ (replacing the fine Block Jacobi solver by ABJ). Note that TMG$_c$ can be interpreted as Parareal using an approximate integration operator and larger time step for the coarse propagator if we set

$$\mathcal{G} := \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}. \tag{5.21}$$

Thus, the version of Parareal used in section 3.2 is equivalent to TMG$_c$ and differs from PFASST only by the type of smoother used on the fine level.

5.5. Analysis and numerical experiments.

5.5.1. Convergence of PFASST iteration components. Since Block Jacobi SDC (5.8) can be written as a primary block iteration, we can apply Theorem 2.8 with $\beta = 0$ to get the error bound

$$e_{n+1}^k \leq \begin{cases} \delta(\gamma+\alpha)^k \text{ if } k \leq n, \\ \delta\gamma^k \displaystyle\sum_{i=0}^{n} \binom{k}{i} \left(\frac{\alpha}{\gamma}\right)^i \text{ otherwise,} \end{cases} \tag{5.22}$$

[18]We implicitly use $[\mathbf{I} - \mathbf{Q}_\Delta]^{-1}(\mathbf{Q} - \mathbf{Q}_\Delta) = \mathbf{I} - [\mathbf{I} - \mathbf{Q}_\Delta]^{-1}(\mathbf{I} - \mathbf{Q}) = \mathbf{I} - \tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\phi}$; see (5.6).

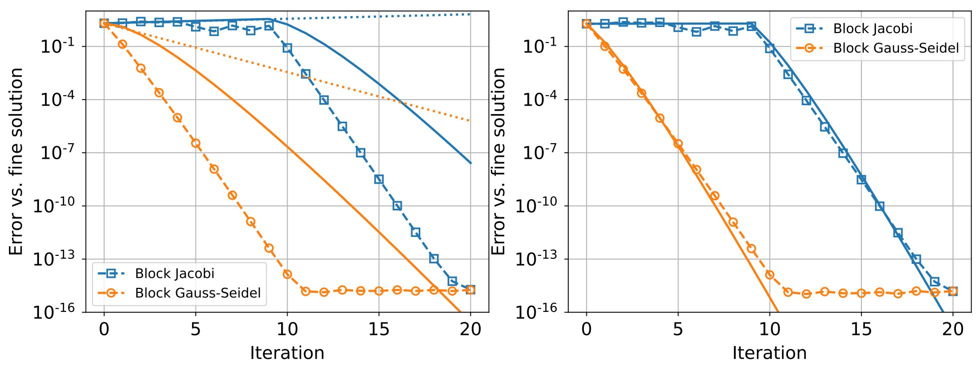


FIG. 7. *Comparison of numerical errors with GFM-bounds for Block Jacobi SDC and Block Gauss–Seidel SDC. Left: error on the block variables (dashed), GFM-bounds (solid), linear bound from the iteration matrix (dotted). Right: error estimate using the interface approximation from Corollary* 2.9. *Note that the numerical errors on block variables (left) and at the interface (right) are close but not identical (see Remark* 2.10*).*

with $\gamma := \left\|[\mathbf{I}-\mathbf{Q}_\Delta]^{-1}(\mathbf{Q}-\mathbf{Q}_\Delta)\right\|$, $\alpha := \left\|[\mathbf{I}-\mathbf{Q}_\Delta]^{-1}\mathbf{H}\right\|$. Note that γ is proportional to $\lambda\Delta t$ through the $\mathbf{Q}-\mathbf{Q}_\Delta$ term and for small Δt, α tends to $\|\mathbf{H}\|$ which is constant. We can identify two convergence regimes: for early iterations ($k \leq n$), the bound does not contract if $\gamma+\alpha \geq 1$ (which is generally the case). For later iterations ($k > n$), a small-enough time step leads to convergence of the algorithm through the γ^k factor.

Similarly, for Block Gauss–Seidel SDC (5.5), Theorem 2.8 with $\alpha = 0$ gives

$$e_{n+1}^k \leq \delta \frac{\gamma^k}{(k-1)!} \sum_{i=0}^{n} \prod_{l=1}^{k-1} (i+l)\beta^i, \tag{5.23}$$

where $\gamma := \left\|[\mathbf{I}-\mathbf{Q}_\Delta]^{-1}(\mathbf{Q}-\mathbf{Q}_\Delta)\right\|$, $\beta := \left\|[\mathbf{I}-\mathbf{Q}_\Delta]^{-1}\mathbf{H}\right\|$. This iteration contracts in early iterations if γ is small enough. Since the value for γ is the same for both Block Gauss–Seidel SDC and Block Jacobi-SDC, both algorithms have an asymptotically similar convergence rate.

We illustrate this with the following example. Let $\lambda := i$, $u_0 := 1$, and let the time interval $[0, \pi]$ be divided into $N = 10$ subintervals. Inside each subinterval, we use one step of the collocation method from section 2.1.2 with $M := 10$ Lobatto–Legendre nodes [27]. This gives us block variables of size $M = 10$ and we choose $\mathbf{Q}_\Delta$ as the matrix defined by a single Backward Euler step between nodes to build the $\tilde{\boldsymbol{\phi}}$ operator. The starting value $\boldsymbol{u}^0$ for the iteration is initialized with random numbers starting from the same seed. Figure 7 (left) shows the numerical error for the last block using the L^∞ norm, the bound obtained with the GFM method, and the linear bound using the norm of the global iteration matrix. As for PARAREAL in section 3.2, the GFM-bound is similiar to the iteration matrix bound for the first few iterations, but much tighter for later iterations. In particular, the linear bound cannot show the change in convergence regime of the Block Jacobi SDC iteration (after $k = 10$) but the GFM-bound does. Also, we observe that while the GFM-bound overestimates the error, the interface approximation of Corollary 2.9 gives a very good estimate of the error at the interface; see Figure 7 (right).

5.5.2. Analysis and convergence of PFASST. The GFM framework provides directly an error bound for PFASST: applying Theorem 2.8 to (5.19) gives

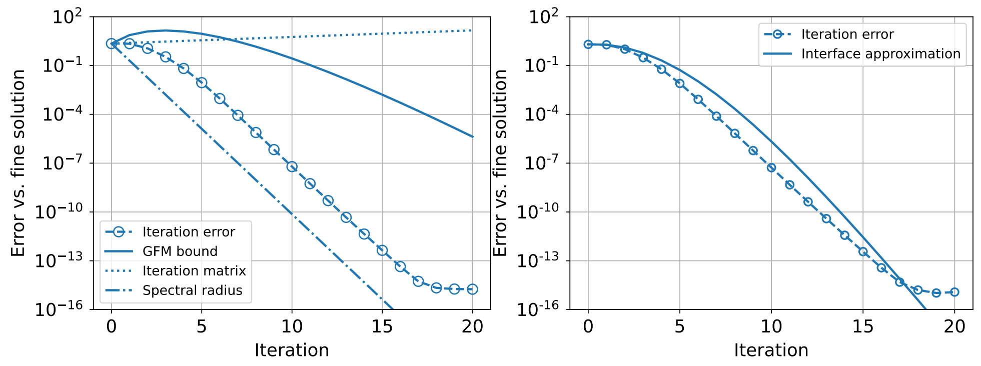


Fig. 8. *Comparison of numerical errors with GFM-bounds for* PFASST*. Left: error bound using volume values. Right: estimate using the interface approximation. Note that the numerical errors on block variables (left) and at the interface (right) are very close but not identical (see Remark* 2.10*).*

$$e_{n+1}^k \leq \delta\gamma^k \sum_{i=0}^{\min(n,k)} \sum_{l=0}^{n-i} \binom{k}{i} \binom{l+k-1}{l} \left(\frac{\alpha}{\gamma}\right)^i \beta^l \tag{5.24}$$

with $\gamma := ||[\mathbf{I}-\mathbf{T}_C^F(\mathbf{I}-\tilde{\mathbf{Q}}_\Delta)^{-1}\mathbf{T}_F^C(\mathbf{I}-\mathbf{Q})][\mathbf{I}-\mathbf{Q}_\Delta]^{-1}(\mathbf{Q}-\mathbf{Q}_\Delta)||$, $\beta := ||\mathbf{T}_C^F(\mathbf{I}-\tilde{\mathbf{Q}}_\Delta)^{-1}\tilde{\mathbf{H}}\mathbf{T}_F^C||$, and $\alpha := ||(\mathbf{I}-\mathbf{T}_C^F[\mathbf{I}-\tilde{\mathbf{Q}}_\Delta]^{-1}\mathbf{T}_F^C(\mathbf{I}-\mathbf{Q}))[\mathbf{I}-\mathbf{Q}_\Delta]^{-1}\mathbf{H}||$.

We compare this bound with numerical experiments. Let $\lambda := i$, $u_0 := 1$. The time interval $[0, 2\pi]$ for the Dahlquist problem (2.1) is divided into $N = 10$ subintervals. Inside each subinterval we use $M := 6$ Lobatto–Legendre nodes on the fine level and $M_C := 2$ Lobatto nodes on the coarse level. The $\mathbf{Q}_\Delta$ and $\tilde{\mathbf{Q}}_\Delta$ operators use Backward Euler. In Figure 8 (left) we compare the measured numerical error with the GFM-bound and the linear bound from the iteration matrix. As in section 5.5.1, both bounds overestimate the numerical error, even if the GFM-bound shows convergence for the later iterations, which the linear bound from the iteration matrix cannot. We also added an error estimate built using the spectral radius of the iteration matrix, for which an upper bound was derived in [4]. For this example, the spectral radius reflects the asymptotic convergence rate for the last iterations better than GFM. This highlights a weakness of the current GFM-bound: applying norm and triangle inequalities to the vector error recurrence (2.25) can induce a large approximation error in the scalar error recurrence (2.26) that is then solved with generating functions. Improving this is planned for future work.

However, one advantage of the GFM-bound over the spectral radius is its generic aspect allowing it to be applied to many iterative algorithms, even those having an iteration matrix with spectral radius equal to zero like Parareal [45]. Furthermore, the interface approximation from Corollary 2.9 allows us to get a significantly better estimation of the numerical error, as shown in Figure 8 (right). For the GFM-bound we have $(\alpha, \beta, \gamma) = (0.16, 1, 0.19)$, while for the interface approximation we get $(\bar{\alpha}, \bar{\beta}, \bar{\gamma}) = (0.16, 0.84, 0.02)$. In the second case, since $\bar{\gamma}$ is one order smaller than the other coefficients, we get an error estimate that is closer to the one for Parareal in section 3.2 where $\gamma = 0$. This similarity between PFASST and Parareal (cf. section 5.4) will be highlighted in the next section.

6. Comparison of iterative PinT algorithms.

Using the notation of the GFM framework, we provide the primary block iterations of all iterative PinT algorithms investigated throughout this paper in Table 2. In particular, the first

Table 2

Summary of all the methods we analyzed, and their block iteration operators. Note that TMG with $\omega = 1$ and TMG$_c$ *corresponds to* Parareal *with a specific choice of the coarse propagator.*

Algorithm	$\mathbf{B}_1^0\ (\boldsymbol{u}_{n+1}^k)$	$\mathbf{B}_0^0\ (\boldsymbol{u}_n^k)$	$\mathbf{B}_0^1\ (\boldsymbol{u}_n^{k+1})$
Damped Block Jacobi	$\mathbf{I} - \omega\mathbf{I}$	$\omega\boldsymbol{\phi}^{-1}\boldsymbol{\chi}$	–
ABJ	$\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-\mathbf{1}}\boldsymbol{\phi}$	$\tilde{\boldsymbol{\phi}}^{-\mathbf{1}}\boldsymbol{\chi}$	–
ABGS	$\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-\mathbf{1}}\boldsymbol{\phi}$	–	$\tilde{\boldsymbol{\phi}}^{-\mathbf{1}}\boldsymbol{\chi}$
Parareal	–	$(\boldsymbol{\phi}^{-1} - \tilde{\boldsymbol{\phi}}^{-\mathbf{1}})\boldsymbol{\chi}$	$\tilde{\boldsymbol{\phi}}^{-\mathbf{1}}\boldsymbol{\chi}$
TMG	$(1-\omega)(\mathbf{I} - \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi})$	$\omega(\boldsymbol{\phi}^{-1} - \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C)\boldsymbol{\chi}$	$\mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}$
TMG$_c$	–	$(\boldsymbol{\phi}^{-1} - \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C)\boldsymbol{\chi}$	$\mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}$
TMG$_f$	$(\mathbf{I} - \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi})(\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-\mathbf{1}}\boldsymbol{\phi})$	$(\tilde{\boldsymbol{\phi}}^{-\mathbf{1}} - \mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi}\tilde{\boldsymbol{\phi}}^{-\mathbf{1}})\boldsymbol{\chi}$	$\mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}$
PFASST	$(\mathbf{I} - \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi})(\mathbf{I} - \tilde{\boldsymbol{\phi}}^{-\mathbf{1}}\boldsymbol{\phi})$	$(\tilde{\boldsymbol{\phi}}^{-\mathbf{1}} - \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C\boldsymbol{\phi}\tilde{\boldsymbol{\phi}}^{-\mathbf{1}})\boldsymbol{\chi}$	$\mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}$

Table 3

Maximum error over time for each block propagator run sequentially. The first column shows the error of the fine propagator, while the next three columns show the error of the three possible approximate propagators. In the top row, $\boldsymbol{\phi}$ corresponds to a collocation method with $M = 5$ nodes while $\boldsymbol{\phi}_C$ is a collocation method with $M = 3$ nodes. $\tilde{\boldsymbol{\phi}}$ is a backward Euler method with $M = 5$ steps per block while $\tilde{\boldsymbol{\phi}}_C$ is Backward Euler with $M = 3$ steps per block. In the bottom row, $\boldsymbol{\phi}$ corresponds to $M = 5$ uniform steps per block of a fourth order Runge–Kutta method, and $\boldsymbol{\phi}_C$ is the same method with $M = 3$ steps per block. $\tilde{\boldsymbol{\phi}}$ is a second order Runge–Kutta method (Heun) with $M = 5$ uniform steps per block while $\tilde{\boldsymbol{\phi}}_C$ is the same method with $M = 3$ uniform time steps per block.

	$\boldsymbol{\phi}^{-1}\boldsymbol{\chi}$	$\tilde{\boldsymbol{\phi}}^{-\mathbf{1}}\boldsymbol{\chi}$	$\mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}$	$\mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}$
Figure 9 (left)	$1.20e^{-5}$	$3.57e^{-1}$	$1.19e^{-2}$	$4.87e^{-1}$
Figure 9 (right)	$3.14e^{-4}$	$6.24e^{-2}$	$5.14e^{-3}$	$2.67e^{-1}$

rows summarize the basic block iterations used as components to build the iterative PinT methods. While damped Block Jacobi (section 2.2.1) and ABJ (section 5.2) are more suitable for smoothing,[19] ABGS (section 2.2.2) is mostly used as a solver (e.g., to compute the CGC). This allows us to compare the convergence of each block iteration, and we illustrate this with the following examples.

We consider the Dahlquist problem with $\lambda := 2i - 0.2$, $u_0 = 1$. First, we decompose the simulation interval $[0, 2\pi]$ into $N = 10$ subintervals. Next, we choose a block discretization with $M = 5$ Lobatto–Legendre nodes, a collocation method on each block for fine integrator $\boldsymbol{\phi}$; see section 2.1.2. We build a coarse block discretization using $M_C = 3$, and define on each level an approximate integrator using Backward Euler. This allows us to define the $\tilde{\boldsymbol{\phi}}$, $\boldsymbol{\phi}_C$, and $\tilde{\boldsymbol{\phi}}_C$ integrators; see the legend of Table 3 for more details, where we show the maximum absolute error in time for each of the four propagators run sequentially. The high order collocation method with $M = 5$ nodes $\boldsymbol{\phi}^{-1}\boldsymbol{\chi}$ is the most accurate. The coarse collocation method with $M = 3$ nodes interpolated to the fine mesh is still more accurate than the Backward Euler method with $M = 5$ nodes $\tilde{\boldsymbol{\phi}}^{-\mathbf{1}}\boldsymbol{\chi}$ or the Backward Euler method with $M = 3$ interpolated to the fine mesh. Then we run all algorithms in Table 2, initializing the block variable iterate with the same random initial guess. The error for the last block variable with respect to the fine sequential solution is shown in Figure 9 (left). In addition, we show the same results in Figure 9 (right), but using the classical fourth order Runge–Kutta method as a fine propagator, second order Runge–Kutta (Heun

[19]Note that algorithms used as smoothers have $\mathbf{B}_0^1 = 0$, which is a necessary condition for parallel computation across all blocks.

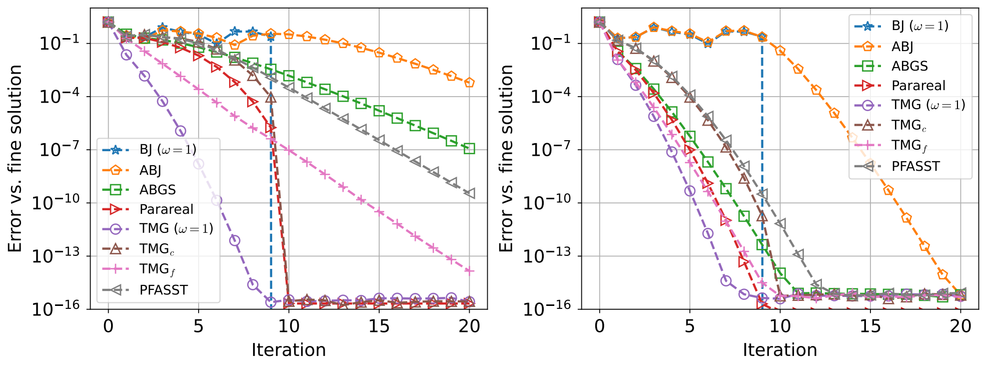


FIG. 9. *Comparison of iterative methods convergence using the GFM framework. Left: collocation as fine integrator. Right: fourth order Runge–Kutta method as fine integrator.* PARAREAL *($\omega=1$) and* PARAREAL *(TMG$_c$) denote a specific coarse propagator for* PARAREAL.

method) for the approximate integration operator, and equidistant points using a volume formulation as described in section 2.1.1. Note that PARAREAL, TMG $_{\omega=1}$, and TMG$_c$ are each PARAREAL algorithms using respectively $\tilde{\boldsymbol{\phi}}^{-1}\boldsymbol{\chi}$, $\mathbf{T}_C^F\boldsymbol{\phi}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}$, and $\mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_C^{-1}\mathbf{T}_F^C\boldsymbol{\chi}$ as coarse propagator $\mathcal{G}$ (see Table 3 for their discretization error).

The TMG iteration converges fastest, since it uses the most accurate block integrators on both levels; cf. Table 3. Keeping the same CGC but approximating the smoother, TMG$_f$ improves the first iterations, but convergence for later iterations is closer to PFASST. This suggests that convergence for later iterations is mostly governed by the accuracy of the smoother since both TMG$_f$ and PFASST use ABJ. This is corroborated by the comparison of PFASST and TMG$_c$, which differ only in their choice of smoother. While the exact Block Jacobi relaxation makes TMG$_c$ converge after $k=N$ iterations (a well-known property of PARAREAL), using the ABJ smoother means that PFASST does not share this property.

On the other hand, the first iterations are also influenced by the CGC accuracy. The iteration error is very similar for PFASST and TMG$_c$, which have the same CGC. This is more pronounced when using the fourth order Runge–Kutta method for $\boldsymbol{\phi}$, as we see in Figure 9 (right). Early iteration errors are similar for two-level methods that use the same CGC (TMG/TMG$_f$, and PFASST/ TMG$_c$). Similarities of the first iteration errors can also be observed for PARAREAL and ABGS. Both algorithms use the same $\mathbf{B}_0^1$ operator; see Table 2. This suggests that early iteration errors are mostly governed by the accuracy of $\mathbf{B}_0^1$, which is corroborated by the two-level methods (TMG and TMG$_f$ use the same $\mathbf{B}_0^1$ operator, as PFASST and TMG$_c$).

Remark 6.1. An important aspect of this analysis is that it compares *only the convergence* of each algorithm, and not their overall computational cost. For instance, PFASST and TMG$_c$ appear to be equivalent for the first iterations, but the block iteration of PFASST is cheaper than TMG$_c$, because an approximate block integrator is used for relaxation. To account for this and build a model for computational efficiency, the GFM framework would need to be combined with a model for computational cost of the different parts in the block iterations. Such a study is beyond the scope of this paper but is the subject of ongoing work.

7. Conclusion.

We have shown that the generating function method (GFM) can be used to compare convergence of different iterative PinT algorithms. To do so, we formulated popular methods like Parareal, PFASST, MGRIT, and TMG in a common framework based on the definition of a primary block iteration. The GFM analysis showed that all these methods eventually converge superlinearly[20] due to the evolution nature of the problems. We confirmed this by numerical experiments, and our Python code is publically available [37].

Our analysis opens up further research directions. For example, studying multistep block iterations like MGRIT with FCF-relaxation and more complex two-level methods without Assumption 4.3 would be a useful extension of the GFM framework. Similarly, an extension to multilevel versions of STMG, PFASST, and MGRIT would be very valuable. Finally, in practice PinT methods are used to solve space-time problems. The GFM framework should be able to provide convergence bounds in this case as well, potentially even for nonlinear problems, considering GFM was used successfully to study Parareal applied to nonlinear systems of ODEs [21].

Appendix A. Error bounds for primary block iterations.

A.1. Incomplete primary block iterations.

First, we consider

$$\text{(PBI-1)}: \quad \boldsymbol{u}_{n+1}^{k+1} = \mathbf{B}_0^1\left(\boldsymbol{u}_n^{k+1}\right) + \mathbf{B}_0^0\left(\boldsymbol{u}_n^k\right), \tag{A.1}$$

$$\text{(PBI-2)}: \quad \boldsymbol{u}_{n+1}^{k+1} = \mathbf{B}_1^0(\boldsymbol{u}_{n+1}^k) + \mathbf{B}_0^0\left(\boldsymbol{u}_n^k\right), \tag{A.2}$$

$$\text{(PBI-3)}: \quad \boldsymbol{u}_{n+1}^{k+1} = \mathbf{B}_1^0(\boldsymbol{u}_{n+1}^k) + \mathbf{B}_0^1\left(\boldsymbol{u}_n^{k+1}\right), \tag{A.3}$$

where one block operator is zero. (PBI-1) corresponds to Parareal, (PBI-2) to Block Jacobi SDC, and (PBI-3) to Block Gauss–Seidel SDC. We recall the notation

$$\alpha := \left\|\mathbf{B}_0^0\right\|, \quad \beta := \left\|\mathbf{B}_0^1\right\|, \quad \gamma := \left\|\mathbf{B}_1^0\right\|. \tag{A.4}$$

Application of Lemma 2.7 gives the recurrence relations

$$\text{(PBI-1)}: \quad \rho_{k+1}(\zeta) \le \frac{\alpha\zeta}{1-\beta\zeta}\rho_k(\zeta) \Longrightarrow \rho_k(\zeta) \le \alpha^k \left(\frac{\zeta}{1-\beta\zeta}\right)^k \rho_0(\zeta), \tag{A.5}$$

$$\text{(PBI-2)}: \quad \rho_{k+1}(\zeta) \le (\gamma + \alpha\zeta)\rho_k(\zeta) \Longrightarrow \rho_k(\zeta) \le \gamma^k \left(1 + \frac{\alpha}{\gamma}\zeta\right)^k \rho_0(\zeta), \tag{A.6}$$

$$\text{(PBI-3)}: \quad \rho_{k+1}(\zeta) \le \frac{\gamma}{1-\beta\zeta}\rho_k(\zeta) \Longrightarrow \rho_k(\zeta) \le \gamma^k \frac{1}{(1-\beta\zeta)^k}\rho_0(\zeta) \tag{A.7}$$

for the corresponding generating functions. Using definition[21] (2.30) for δ, we find that $\rho_0(\zeta) \le \delta \sum_{n=0}^{\infty} \zeta^{n+1}$. By using the binomial series expansion

$$\frac{1}{(1-\beta\zeta)^k} = \sum_{n=0}^{\infty} \binom{n+k-1}{n} (\beta\zeta)^n \tag{A.8}$$

[20]This is due to the factorial term stemming from the binomial sums in the estimates (2.32)–(2.35).

[21]The definition of δ as maximum error for $n \in \{0, \ldots, N\}$ can be extended to $n \in \mathbb{N}$, as the error values for $n > N$ do not matter and can be set to zero.

for $k > 0$ and the Newton binomial sum, we obtain for the three block iterations

$$\text{(PBI-1)}: \quad \rho_k(\zeta) \le \delta\alpha^k\zeta\left[\sum_{n=0}^{\infty}\binom{n+k-1}{n}\beta^n\zeta^{n+k}\right]\left[\sum_{n=0}^{\infty}\zeta^n\right], \tag{A.9}$$

$$\text{(PBI-2)}: \quad \rho_k(\zeta) \le \delta\gamma^k\zeta\left[\sum_{n=0}^{k}\binom{k}{n}\left(\frac{\alpha}{\gamma}\right)^n\zeta^n\right]\left[\sum_{n=0}^{\infty}\zeta^n\right], \tag{A.10}$$

$$\text{(PBI-3)}: \quad \rho_k(\zeta) \le \delta\gamma^k\zeta\left[\sum_{n=0}^{\infty}\binom{n+k-1}{n}\beta^n\zeta^n\right]\left[\sum_{n=0}^{\infty}\zeta^n\right]. \tag{A.11}$$

Error bound for PBI-1. We simplify the expression using

$$\left[\sum_{n=0}^{\infty}\binom{n+k-1}{n}\beta^n\zeta^{n+k}\right] = \left[\sum_{n=k}^{\infty}\binom{n-1}{n-k}\beta^{n-k}\zeta^n\right], \tag{A.12}$$

and then the series product formula

$$\left[\sum_{n=0}^{\infty}a_n\zeta^n\right]\left[\sum_{n=0}^{\infty}b_n\zeta^n\right] = \sum_{n=0}^{\infty}c_n\zeta^n, \quad c_n = \sum_{i=0}^{n}a_ib_{n-i}, \tag{A.13}$$

with $b_n = 1$ and

$$a_n = \begin{cases} 0 \text{ if } n < k, \\ \binom{n-1}{n-k}\beta^{n-k} \text{ otherwise.} \end{cases} \tag{A.14}$$

From this we get

$$c_n = \sum_{i=k}^{n}\binom{i-1}{i-k}\beta^{i-k} = \sum_{i=0}^{n-k}\binom{i+k-1}{i}\beta^i = \sum_{i=0}^{n-k}\frac{\prod_{l=1}^{k-1}(i+l)}{(k-1)!}\beta^i, \tag{A.15}$$

using the convention that the product reduces to one when there are no terms in it. Identifying coefficients in the power series and rearranging terms yields for $k > 0$

$$\boxed{\text{(PBI-1)}: \quad e_{n+1}^k \le \delta\frac{\alpha^k}{(k-1)!}\sum_{i=0}^{n-k}\prod_{l=1}^{k-1}(i+l)\beta^i.} \tag{A.16}$$

Following an idea by Gander and Hairer [21], we can also consider the error recurrence $e_{n+1}^{k+1} \le \alpha e_n^k + \bar{\beta}e_n^{k+1}$, $\bar{\beta} = \max(1,\beta)$. Using the upper bound $\sum_{n=0}^{\infty}\zeta^n = \frac{1}{1-\zeta} \le \frac{1}{1-\bar{\beta}\zeta}$, for the initial error, we avoid the series product and get $\rho_k(\zeta) \le \delta\alpha^k\frac{\zeta^k}{(1-\bar{\beta})^{k+1}}$ as bound on the generating function. We then obtain the simpler error bound

$$e_{n+1}^k \le \delta\frac{\alpha^k}{k!}\bar{\beta}^{n-k}\prod_{l=1}^{k}(n+1-l) \tag{A.17}$$

as in the proof of [21, Thm. 1].

*Error bound for PBI-*2. We use (A.13) again with $b_n = 1$ to get

$$a_n = \begin{cases} \binom{k}{n}\left(\frac{\alpha}{\gamma}\right)^n & \text{if } n \le k, \\ 0 & \text{otherwise.} \end{cases} \tag{A.18}$$

From this we get $c_n = \sum_{i=0}^{\min(n,k)} \binom{k}{i}\left(\frac{\alpha}{\gamma}\right)^i$, which yields for $k > 0$ the error bound

$$\boxed{\text{(PBI-2)}: \quad e_{n+1}^k \le \begin{cases} \delta(\gamma+\alpha)^k & \text{if } k \le n, \\ \delta\gamma^k \sum_{i=0}^{n} \binom{k}{i}\left(\frac{\alpha}{\gamma}\right)^i & \text{otherwise.} \end{cases}} \tag{A.19}$$

*Error bound for PBI-*3. We use (A.13) with $b_n = 1$ for the series product to get

$$a_n = \binom{n+k-1}{n}\beta^n = \frac{\prod_{l=1}^{k-1}(n+l)}{(k-1)!}\beta^n, \tag{A.20}$$

which yields the error bound

$$\boxed{\text{(PBI-3)}: \quad e_{n+1}^k \le \delta \frac{\gamma^k}{(k-1)!} \sum_{i=0}^{n} \prod_{l=1}^{k-1} (i+l)\beta^i} \tag{A.21}$$

for $k > 0$.

A.2. Full primary block iteration. We now consider a primary block iteration (2.13) with all block operators nonzero,

$$\text{(PBI-Full)}: \quad \boldsymbol{u}_{n+1}^{k+1} = \mathbf{B}_1^0\left(\boldsymbol{u}_{n+1}^k\right) + \mathbf{B}_0^1\left(\boldsymbol{u}_n^{k+1}\right) + \mathbf{B}_0^0\left(\boldsymbol{u}_n^k\right), \tag{A.22}$$

with α, β, and γ defined in (A.4). Applying Lemma 2.7 leads to

$$\rho_{k+1}(\zeta) \le \frac{\gamma+\alpha\zeta}{1-\beta\zeta}\rho_k(\zeta) \quad \Longrightarrow \quad \rho_k(\zeta) \le \left(\frac{\gamma+\alpha\zeta}{1-\beta\zeta}\right)^k \rho_0(\zeta). \tag{A.23}$$

Combining the calculations performed for PBI-2 and PBI-3, we obtain

$$\rho_k(\zeta) \le \delta\zeta\gamma^k \left[\sum_{n=0}^{k} \binom{k}{n}\left(\frac{\alpha}{\gamma}\right)^n \zeta^n\right] \left[\sum_{n=0}^{\infty} \binom{n+k-1}{n}\beta^n\zeta^n\right] \left[\sum_{n=0}^{\infty} \zeta^n\right] \tag{A.24}$$

$$= \delta\zeta\gamma^k \left[\sum_{n=0}^{k} \binom{k}{n}\left(\frac{\alpha}{\gamma}\right)^n \zeta^n\right] \left[\sum_{n=0}^{\infty}\sum_{i=0}^{n} \binom{i+k-1}{i}\beta^i\zeta^n\right]. \tag{A.25}$$

Then using (A.13) with

$$a_n = \begin{cases} \binom{k}{n}\left(\frac{\alpha}{\gamma}\right)^n & \text{if } n \le k, \\ 0 & \text{otherwise,} \end{cases} \qquad b_n = \sum_{i=0}^{n} \binom{i+k-1}{i}\beta^i, \tag{A.26}$$

we obtain

$$\rho_k(\zeta) \le \delta\zeta\gamma^k \sum_{n=0}^{\infty} c_n\zeta^n, \text{ with } c_n = \sum_{i=0}^{\min(n,k)} \sum_{l=0}^{n-i} \binom{k}{i}\binom{l+k-1}{l}\left(\frac{\alpha}{\gamma}\right)^i \beta^l. \tag{A.27}$$

From this we can identify the error bound

$$\boxed{\text{(PBI-Full)}: \quad e_{n+1}^{k} \leq \delta\gamma^{k} \sum_{i=0}^{\min(n,k)} \sum_{l=0}^{n-i} \binom{k}{i} \binom{l+k-1}{l} \left(\frac{\alpha}{\gamma}\right)^{i} \beta^{l}.} \tag{A.28}$$

Reproducibility of computational results.. This paper has been awarded the "SIAM Reproducibility Badge: code and data available", as a recognition that the authors have followed reproducibility principles valued by SISC and the scientific computing community. Code and data that allow readers to reproduce the results in this paper are available at https://github.com/parallel-in-Time/gfm.

Acknowledgment. We greatly appreciate the very detailed feedback from the anonymous reviewers. It helped a lot to improve the organization of the paper and to make it more accessible.

REFERENCES

[1] E. Aubanel, *Scheduling of tasks in the Parareal algorithm*, Parallel Comput., 37 (2011), pp. 172–182.

[2] G. Bal, *On the convergence and the stability of the parareal algorithm to solve partial differential equations*, in Domain Decomposition Methods in Science and Engineering, Lect. Notes Comput. Sci. Eng. 40, R. Kornhuber, et al., eds., Springer, Berlin, 2005, pp. 426–432.

[3] M. Bolten, D. Moser, and R. Speck, *A multigrid perspective on the parallel full approximation scheme in space and time*, Numer. Linear Algebra Appl., 24 (2017), e2110.

[4] M. Bolten, D. Moser, and R. Speck, *Asymptotic convergence of the parallel full approximation scheme in space and time for linear problems*, Numer. Linear Algebra Appl., 25 (2018), e2208.

[5] J. Burmeister and G. Horton, *Time-parallel multigrid solution of the Navier-Stokes equations*, in Multigrid Methods III, Springer, Berlin, 1991, pp. 155–166.

[6] A. J. Christlieb, C. B. Macdonald, and B. W. Ong, *Parallel high-order integrators*, SIAM J. Sci. Comput., 32 (2010), pp. 818–835.

[7] P.-H. Cocquet and M. J. Gander, *How large a shift is needed in the shifted Helmholtz preconditioner for its effective inversion by multigrid?*, SIAM J. Sci. Comput., 39 (2017), pp. A438–A478.

[8] V. Dobrev, T. Kolev, N. Petersson, and J. Schroder, *Two-Level Convergence Theory for Multigrid Reduction in Time (MGRIT)*, Technical report LLNL-JRNL-692418, Lawrence Livermore National Laboratory, 2016.

[9] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT, 40 (2000), pp. 241–266.

[10] H. C. Elman, O. G. Ernst, and D. P. O'leary, *A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations*, SIAM J. Sci. Comput., 23 (2001), pp. 1291–1315.

[11] M. Emmett and M. Minion, *Toward an efficient parallel in time method for partial differential equations*, Commun. Appl. Math. Comput. Sci., 7 (2012), pp. 105–132.

[12] O. G. Ernst and M. J. Gander, *Why it is difficult to solve Helmholtz problems with classical iterative methods*, in Numerical Analysis of Multiscale Problems, Springer, Berlin, 2012, pp. 325–363.

[13] O. G. Ernst and M. J. Gander, *Multigrid methods for Helmholtz problems: A convergent scheme in 1d using standard components*, Direct Inverse Probl. Wave Propagation Appl., 14 (2013).

[14] R. D. Falgout, S. Friedhoff, Tz. V. Kolev, S. P. MacLachlan, and J. B. Schroder, *Parallel time integration with multigrid*, SIAM J. Sci. Comput., 36 (2014), pp. C635–C661.

[15] 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.

[16] S. Friedhoff, R. Falgout, T. Kolev, S. MacLachlan, and J. Schroder, *A multigrid-in-time algorithm for solving evolution equations in parallel*, in Proceedings of the 16th Copper Mountain Conference on Multigrid Methods, 2013.

[17] M. J. Gander, *Analysis of the Parareal algorithm applied to hyperbolic problems using characteristics*, Bol. Soc. Esp. Mat. Apl., 42 (2008), pp. 21–35.

[18] M. J. Gander, 50 *years of time parallel time integration*, in Multiple Shooting and Time Domain Decomposition Methods, T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, eds., Springer, Berlin, 2015, pp. 69–114.

[19] M. J. Gander, I. G. Graham, and E. A. Spence, *Applying GMRES to the Helmholtz equation with shifted Laplacian preconditioning: What is the largest shift for which wavenumber-independent convergence is guaranteed?*, Numer. Math., 131 (2015), pp. 567–614.

[20] M. J. Gander and S. Güttel, *PARAEXP: A parallel integrator for linear initial-value problems*, SIAM J. Sci. Comput., 35 (2013), pp. C123–C142.

[21] M. J. Gander and E. Hairer, *Nonlinear convergence analysis for the Parareal algorithm*, in Domain Decomposition Methods in Science and Engineering, Lect. Notes in Comput. Sci. Eng. 60, O. B. Widlund and D. E. Keyes, eds., Springer, Berlin, 2008, pp. 45–56.

[22] M. J. Gander, F. Kwok, and H. Zhang, *Multigrid interpretations of the Parareal algorithm leading to an overlapping variant and MGRIT*, Comput. Vis. Sci., 19 (2018), pp. 59–74.

[23] M. J. Gander and T. Lunet, *Toward error estimates for general space-time discretizations of the advection equation*, Comput. Vis. Sci., 23 (2020), pp. 1–14.

[24] M. J. Gander and T. Lunet, *ParaStieltjes: Parallel computation of Gauss quadrature rules using a Parareal-like approach for the Stieltjes procedure*, Numer. Linear Algebra Appl., 28 (2021), e2314.

[25] M. J. Gander and M. Neumüller, *Analysis of a new space-time parallel multigrid algorithm for parabolic problems*, SIAM J. Sci. Comput., 38 (2016), pp. A2173–A2208.

[26] M. J. Gander and S. Vandewalle, *Analysis of the Parareal time-parallel time-integration method*, SIAM J. Sci. Comput., 29 (2007), pp. 556–578.

[27] W. Gautschi, *Orthogonal Polynomials: Computation and Approximation*, Oxford University Press, New York, 2004.

[28] S. Götschel, M. Minion, D. Ruprecht, and R. Speck, *Twelve ways to fool the masses when giving parallel-in-time results*, in Springer Proc. Math. Stat., Springer, Berlin, 2021, pp. 81–94.

[29] S. Günther, L. Ruthotto, J. B. Schroder, E. C. Cyr, and N. R. Gauger, *Layer-parallel training of deep residual neural networks*, SIAM J. Math. Data Sci., 2 (2020), pp. 1–23.

[30] W. Hackbusch, *Parabolic multi-grid methods*, in Proceedings of the Sixth International Symposium on Computing Methods in Applied Sciences and Engineering, VI, North-Holland, Amsterdam, 1984, pp. 189–197.

[31] W. Hackbusch, *Multi-grid Methods and Applications*, Vol. 4, Springer, Berlin, 2013.

[32] A. Hessenthaler, B. S. Southworth, D. Nordsletten, O. Röhrle, R. D. Falgout, and J. B. Schroder, *Multilevel convergence analysis of multigrid-reduction-in-time*, SIAM J. Sci. Comput., 42 (2020), pp. A771–A796.

[33] C. Hofer, U. Langer, M. Neumüller, and R. Schneckenleitner, *Parallel and robust preconditioning for space-time isogeometric analysis of parabolic evolution problems*, SIAM J. Sci. Comput., 41 (2019), pp. A1793–A1821.

[34] D. E. Knuth, *The Art of Computer Programming. 1. Fundamental Algorithms*, Addison-Wesley, Reading, MA, 1975.

[35] M. Lecouvez, R. D. Falgout, C. S. Woodward, and P. Top, *A parallel multigrid reduction in time method for power systems*, in Proceedings of the Power and Energy Society General Meeting, IEEE, 2016, pp. 1–5.

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

[37] T. Lunet, *Parallel-in-time/gfm: SISC*, https://doi.org/10.5281/zenodo.7871102, 2023.

[38] T. Lunet, J. Bodart, S. Gratton, and X. Vasseur, *Time-parallel simulation of the decay of homogeneous turbulence using Parareal with spatial coarsening*, Comput. Vis. Sci., 19 (2018), pp. 31–44.

[39] Y. Maday and E. M. Rønquist, *Parallelization in time through tensor-product space-time solvers*, C. R. Math., 346 (2008), pp. 113–118.

[40] M. L. Minion, R. Speck, M. Bolten, M. Emmett, and D. Ruprecht, *Interweaving PFASST and parallel multigrid*, SIAM J. Sci. Comput., 37 (2015), pp. S244–S263.

[41] S. Murata, N. Satofuka, and T. Kushiyama, *Parabolic multi-grid method for incompressible viscous flows using a group explicit relaxation scheme*, Comput. & Fluids, 19 (1991), pp. 33–41.

[42] B. W. Ong and J. B. Schroder, *Applications of time parallelization*, Comput. Vis. Sci., 23 (2020).

[43] E. M. Rønquist and A. T. Patera, *Spectral element multigrid.* I. *Formulation and numerical results*, J. Sci. Comput., 2 (1987), pp. 389–406.

[44] D. Ruprecht, *Wave propagation characteristics of parareal*, Comput. Vis. Sci., 19 (2018), pp. 1–17.

[45] D. Ruprecht and R. Krause, *Explicit parallel-in-time integration of a linear acoustic-advection system*, Comput. & Fluids, 59 (2012), pp. 72–83.

[46] D. Ruprecht and R. Speck, *Spectral deferred corrections with fast-wave slow-wave splitting*, SIAM J. Sci. Comput., 38 (2016), pp. A2535–A2557.

[47] M. Schreiber, P. S. Peixoto, T. Haut, and B. Wingate, *Beyond spatial scalability limitations with a massively parallel method for linear oscillatory problems*, Internat. J. High Perform. Comput. Appl., 32 (2018), pp. 913–933.

[48] B. S. Southworth, *Necessary conditions and tight two-level convergence bounds for Parareal and multigrid reduction in time*, SIAM J. Matrix Anal. Appl., 40 (2019), pp. 564–608.

[49] R. Speck, D. Ruprecht, R. Krause, M. Emmett, M. L. Minion, M. Winkel, and P. Gibbon, *A massively space-time parallel N-body solver*, in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, IEEE, 2012, pp. 92:1–92:11.

[50] G. A. Staff and E. M. Rønquist, *Stability of the Parareal algorithm*, in Domain Decomposition Methods in Science and Engineering, Lect. Notes Comput. Sci. Eng. 40, R. Kornhuber, et al., eds., Springer, Berlin, 2005, pp. 449–456.

[51] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, New York, 2001.

[52] G. Wanner and E. Hairer, *Solving Ordinary Differential Equations* II, Springer, Berlin, 1996.