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 (PinT) algorithms have been proposed, like Parareal, PFASST, MGRIT, and Space-Time Multi-Grid (STMG). These methods have been described using different notations and the convergence estimates that are available for some of them 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 super-linearly and compare them directly numerically. Our framework also allows us to find new methods.

Key words. Parallel in Time (PinT) methods, Parareal, PFASST, MGRIT, space-time multigrid (STMG), generating functions, convergence estimates.

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

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 [11] or Ong and Schroder [33]. Four iterative algorithms have received significant attention, namely Parareal [28] (426 citat. since 2001)[2], PFASST [8] (228 citat. since 2012), MGRIT [10] (238 citat. since 2014) and a specific form of Space-Time Multi-Grid (STMG) [17] (122 citat. since 2016). Other algorithms have been proposed, *e.g.* the parallel implicit time-integrator PITA [9] (264 citat. since 2003) which is very similar to Parareal, the diagonalization technique [30] (50 citat. since 2008), RIDC [5] (108 citat. since 2010), ParaExp [12] (89 citat. since 2013) or REXI [36] (23 citat. since 2018).

Parareal, PFASST, MGRIT and STMG have all been benchmarked for large-

[*]Submitted to the editors DATE.

Funding: This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955701. The JU receives support from the European Union's Horizon 2020 research and innovation programme and 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, Switzerland.

[‡]Hamburg University of Technology, Germany (thibaut.lunet@tuhh.de).

[§]Forschungszentrum Jülich GmbH, Germany.

[1]See also https://www.parallel-in-time.org

[2]Number of citations since publication, according to Google Scholar in February 2022.

scale problems using large numbers of cores of high-performance computing systems [25, 27, 29, 38]. They represent 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 number of iterations, understanding convergence mechanisms and obtaining reliable error bounds for these iterative PinT methods is crucial. Individual analyses exist for Parareal [1, 13, 18, 34, 39], MGRIT [6, 24, 37], PFASST [2, 3], or STMG [17]. There are also a few combined analyses showing equivalences between Parareal and MGRIT [14, 18] or connections between MGRIT and PFASST [31]. However, no systematic comparison of convergence behaviour 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 [20]. 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 and STMG using the same notation. Then, we obtain rigorous comparable error bounds for all four methods by using the Generating Function Method (GFM) [26]. GFM has been used to analyze Parareal [13] and was used to relate Parareal and MGRIT [14]. But our use of GFM to derive common convergence bounds across multiple algorithms is novel.

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) or mixed type.

2.1. Blocks, block variables, and block operators.

We decompose the time interval $[0,T]$ into N time sub-intervals $[t_n, t_{n+1}]$ of uniform size Δt with $n \in \{0, ..., N-1\}$.

Definition 2.1 (time block). *A* time block *(or simply block) denotes the discretization of a time sub-interval* $[t_n, t_{n+1}]$ *using* one or several *grid points,*

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

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 sub-interval. 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 level 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 2.2.3.

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

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

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

Some iterative PinT methods like Parareal use values defined at the interfaces between sub-intervals $[t_n, t_{n+1}]$. Other algorithms, like PFASST, update solution values in the interior of blocks. In the first case, the block variable is the interface value with $M = 1$ and thus $\tau_1 = 1$ while in the second it consists of the volume values in the time block for $[t_n, t_{n+1}]$ 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 [21] or when computing Gauss quadrature formulas [16]. 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*

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

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

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

2.1.1. Example with Runge-Kutta methods. Consider the 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 [40, Chap. IV, Sec. 4] and instrumental when defining PFASST in Section 4. We show their representation with block operators. Starting from the Picard formulation for (2.1) in one time sub-interval $[t_n, t_{n+1}]$,

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

we choose a quadrature rule to approximate the integral. We consider here only Lobatto or Radau-II type quadrature nodes where the last quadrature node coincides with the right sub-interval boundary. This gives us the nodes $\tau_{n,m}$ of Definition 2.1 and we can 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

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

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[3]. The integration and transfer block operators from Definition 2.4 then become $\boldsymbol{\phi} := (\mathbf{I} - \mathbf{Q})$, $\boldsymbol{\chi} := \mathbf{H}$.

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 are iterative methods solving (2.12) by updating a vector $\boldsymbol{u}^k = [\boldsymbol{u}_1^k, \dots, \boldsymbol{u}_N^k]^T$ to $\boldsymbol{u}^{k+1}$ until some stopping criterion is satisfied. Furthermore, if the global iteration can be written for each line of the system separately, we call the update formula for one line a *block iteration.*

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

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

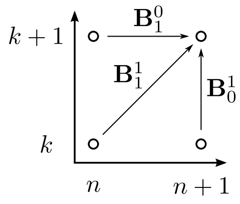
where $\mathbf{B}_0^1$, $\mathbf{B}_1^0$ and $\mathbf{B}_1^1$ are linear operators from $\mathbb{C}^M$ to $\mathbb{C}^M$ that satisfy the consistency condition[4]

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

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 [14] can also be considered for the study of MGRIT with FCF-relaxation; see also the discussion in Section 2.2.4.

[3]This specific form of the $\mathbf{H}$ matrix is induced by the use of Lobatto or Radau-II rules, which count the right interface of the time sub-interval as node. If one uses Radau-I or Gauss-type quadrature rules that do not use the right boundary as node, we get a denser matrix that extrapolates the interface value using interior volume values.

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



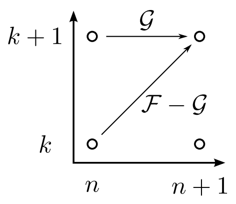


FIG. 1. *kn-graphs for a generic Primary Block Iteration (left) and* PARAREAL *(right).*

2.2.1. Example with Parareal. Consider the interface formulation (2.8) and define a coarse block operator $\boldsymbol{\phi}_{\boldsymbol{\Delta}} := R(\lambda\Delta t)^{-1}$ that approximates the fine block operator $\boldsymbol{\phi} := R(\lambda\Delta t/\ell)^{-\ell}$. Here $\boldsymbol{\phi}_{\boldsymbol{\Delta}}$ corresponds to a coarse grid in time, but nothing prevents the use of a lower order time integration method instead, or a combination of both. Then, we define as coarse and fine propagators

$$\mathcal{G} := \boldsymbol{\phi}_{\boldsymbol{\Delta}}^{\boldsymbol{-1}}\boldsymbol{\chi}, \quad \mathcal{F} := \boldsymbol{\phi}^{\boldsymbol{-1}}\boldsymbol{\chi}, \tag{2.15}$$

so that the PARAREAL update formula [28] yields

$$\boldsymbol{u}_{n+1}^{k+1} = \mathcal{F}\boldsymbol{u}_n^k + \mathcal{G}\boldsymbol{u}_n^{k+1} - \mathcal{G}\boldsymbol{u}_n^k = (\boldsymbol{\phi}^{-1} - \boldsymbol{\phi}_{\boldsymbol{\Delta}}^{-1})\boldsymbol{\chi}\boldsymbol{u}_n^k + \boldsymbol{\phi}_{\boldsymbol{\Delta}}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^{k+1}. \tag{2.16}$$

We recognize a primary block iteration with $\mathbf{B}_0^1 := 0$, $\mathbf{B}_1^0 := \mathcal{G}$ and $\mathbf{B}_1^1 := \mathcal{F} - \mathcal{G}$. Its kn-graph is shown in Figure 1 (right). Furthermore, the consistency condition (2.14) is satisfied, $(0 - \mathbf{Id}) \circ \mathcal{F} + \mathcal{G} + (\mathcal{F} - \mathcal{G}) = 0$. If we subtract $\boldsymbol{u}_{n+1}^k$ in (2.16) and multiply both sides by $\boldsymbol{\phi}$ and rearrange terms, we can write PARAREAL as the preconditioned fixed point iteration

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \mathbf{M}^{-1}(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k), \quad \mathbf{M} := \begin{pmatrix} \boldsymbol{\phi} & & \\ -\boldsymbol{\phi}\boldsymbol{\phi}_{\boldsymbol{\Delta}}^{\boldsymbol{-1}}\boldsymbol{\chi} & \boldsymbol{\phi} & \\ & \ddots & \ddots \end{pmatrix}. \tag{2.17}$$

This is a formulation of PARAREAL as approximate Gauss-Seidel preconditioning (with approximated lower diagonal) for solving (2.12).

2.2.2. Example with a block Jacobi relaxation. A damped block Jacobi iteration for the global problem (2.12) can be written as

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \omega\mathbf{D}^{-1}(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k), \tag{2.18}$$

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

$$\boldsymbol{u}_{n+1}^{k+1} = (1 - \omega)\boldsymbol{u}_{n+1}^k + \omega\boldsymbol{\phi}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^k, \tag{2.19}$$

which is a primary block iteration with $\mathbf{B}_1^0 = 0$. Its kn-graph is shown in Figure 2 (left). The consistency condition (2.14) is also satisfied, $((1 - \omega)\mathbf{Id} - \mathbf{Id}) \circ \boldsymbol{\phi}^{-1}\boldsymbol{\chi} + 0 + \omega\boldsymbol{\phi}^{-1}\boldsymbol{\chi} = 0$.

2.2.3. Example with coarse correction. As a last example, we consider the coarse correction only[5] of a two-level multi-grid method [23] applied to (2.12). Let

[5]The coarse correction is not convergent by itself without smoother.

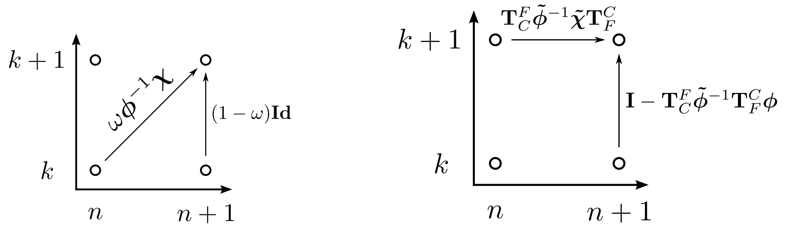

$k+1$
k
n
$n+1$
$\omega\phi^{-1}\chi$
$(1-\omega)\mathbf{Id}$
$\mathbf{T}_C^F\tilde{\phi}^{-1}\chi\mathbf{T}_F^C$
$\mathbf{I}-\mathbf{T}_C^F\tilde{\phi}^{-1}\mathbf{T}_F^C\phi$

FIG. 2. *kn-graphs for the block Jacobi relaxation (left) and the coarse correction (right).*

$\tilde{\boldsymbol{u}}$ be a coarse block variable on the coarse time block $(\tilde{\tau}_m)_{1\leq m\leq \tilde{M}}$ with $\tilde{M} < M$ and consider the associated coarse problem

$$\tilde{\mathbf{A}}\tilde{\boldsymbol{u}} := \begin{pmatrix} \tilde{\boldsymbol{\phi}} & & & \\ -\tilde{\boldsymbol{\chi}} & \tilde{\boldsymbol{\phi}} & & \\ & \ddots & \ddots & \\ & & \tilde{\boldsymbol{\chi}} & \tilde{\boldsymbol{\phi}} \end{pmatrix} \begin{bmatrix} \tilde{\boldsymbol{u}}_1 \\ \tilde{\boldsymbol{u}}_2 \\ \vdots \\ \tilde{\boldsymbol{u}}_N \end{bmatrix} = \begin{bmatrix} \mathbf{T}_F^C\boldsymbol{\chi}(u_0\mathbb{1}) \\ 0 \\ \vdots \\ 0 \end{bmatrix} =: \tilde{\boldsymbol{f}} \tag{2.20}$$

built from (2.12). Here, $\mathbf{T}_F^C$ is a block restriction operator, i.e. a transfer matrix from a fine (F) to coarse (C) representation. Similarly we also have a block prolongation operator $\mathbf{T}_C^F$, i.e. a transfer matrix from a coarse (C) to a fine (F) representation. A coarse correction step applied to (2.12) can be represented as

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \bar{\mathbf{T}}_C^F\tilde{\mathbf{A}}^{-1}\bar{\mathbf{T}}_F^C(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k), \tag{2.21}$$

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 writing (2.21) in two steps,

$$\tilde{\mathbf{A}}\boldsymbol{d} = \bar{\mathbf{T}}_F^C(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k), \tag{2.22}$$

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \bar{\mathbf{T}}_C^F\boldsymbol{d}, \tag{2.23}$$

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

$$\boldsymbol{d}_{n+1} = \tilde{\boldsymbol{\phi}}^{-1}\mathbf{T}_F^C\boldsymbol{\chi}\boldsymbol{u}_n^k - \tilde{\boldsymbol{\phi}}^{-1}\mathbf{T}_F^C\boldsymbol{\phi}\boldsymbol{u}_{n+1}^k + \tilde{\boldsymbol{\phi}}^{-1}\tilde{\boldsymbol{\chi}}\boldsymbol{d}_n, \tag{2.24}$$

$$\boldsymbol{u}_{n+1}^{k+1} = \boldsymbol{u}_{n+1}^k + \mathbf{T}_C^F\boldsymbol{d}_{n+1}. \tag{2.25}$$

Now we need the following simplifying assumption.

Assumption 2.6. The successive application of prolongation $\mathbf{T}_C^F$ followed by restriction $\mathbf{T}_F^C$ leaves the coarse block variables unchanged, *i.e.*

$$\mathbf{T}_F^C \circ \mathbf{T}_C^F = \mathbf{Id}. \tag{2.26}$$

This condition is satisfied in many situations (*e.g.* restriction with standard injection, and polynomial interpolation)[6]. Using it in (2.25) for block index n yields

$$\boldsymbol{d}_n = \mathbf{T}_F^C\left(\boldsymbol{u}_n^{k+1} - \boldsymbol{u}_n^k\right). \tag{2.27}$$

[6]In some situations, *e.g.* when the transpose of the prolongation is used for restriction, we do not get the identity but an invertible matrix. The same simplification can be done, except one must take into account the inverse of $(\mathbf{T}_F^C\mathbf{T}_C^F)$.

Inserting $\boldsymbol{d}_n$ into (2.24) on the right and the resulting $\boldsymbol{d}_{n+1}$ into (2.25) leads to

$$\boldsymbol{u}_{n+1}^{k+1} = (\mathbf{I} - \mathbf{T}_C^F \tilde{\boldsymbol{\phi}}^{-1} \mathbf{T}_F^C \boldsymbol{\phi}) \boldsymbol{u}_{n+1}^k + \mathbf{T}_C^F \tilde{\boldsymbol{\phi}}^{-1} \tilde{\boldsymbol{\chi}} \mathbf{T}_F^C \boldsymbol{u}_n^{k+1} + \mathbf{T}_C^F \tilde{\boldsymbol{\phi}}^{-1} \boldsymbol{\Delta}_{\boldsymbol{\chi}} \boldsymbol{u}_n^k, \tag{2.28}$$

with $\boldsymbol{\Delta}_{\boldsymbol{\chi}} := \mathbf{T}_F^C \boldsymbol{\chi} - \tilde{\boldsymbol{\chi}} \mathbf{T}_F^C$. To simplify further, we make the following assumption.

Assumption 2.7. We consider operators $\mathbf{T}_F^C$, $\boldsymbol{\chi}$ and $\tilde{\boldsymbol{\chi}}$ such that

$$\boldsymbol{\Delta}_{\boldsymbol{\chi}} = \mathbf{T}_F^C \boldsymbol{\chi} - \tilde{\boldsymbol{\chi}} \mathbf{T}_F^C = 0. \tag{2.29}$$

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 assumption is important to define PFASST (*cf.* Section 4.3 and see Bolten et al. [2, Remark 1] for more details) and simplifies the analysis of STMG (*cf.* Section 5), as both methods use this block iteration. Then, (2.28) reduces to

$$\boldsymbol{u}_{n+1}^{k+1} = (\mathbf{I} - \mathbf{T}_C^F \tilde{\boldsymbol{\phi}}^{-1} \mathbf{T}_F^C \boldsymbol{\phi}) \boldsymbol{u}_{n+1}^k + \mathbf{T}_C^F \tilde{\boldsymbol{\phi}}^{-1} \mathbf{T}_F^C \boldsymbol{\chi} \boldsymbol{u}_n^{k+1}. \tag{2.30}$$

Again, we recognize a primary block iteration for which the kn-graph is given in Figure 2 (right). It satisfies the consistency condition[7] (2.14), $((\mathbf{I} - \mathbf{T}_C^F \tilde{\boldsymbol{\phi}}^{-1} \mathbf{T}_F^C \boldsymbol{\phi}) - \mathbf{I}) \circ \boldsymbol{\phi}^{-1} \boldsymbol{\chi} + \mathbf{T}_C^F \tilde{\boldsymbol{\phi}}^{-1} \mathbf{T}_F^C \boldsymbol{\chi} = 0$.

2.2.4. General remarks. We can represent many existing iterative PinT algorithms (Parareal, MGRIT with F-relaxation, PFASST, STMG with two-levels, ...) as primary block iterations. However, some methods can not be written like this, *e.g.* MGRIT with FCF-relaxation, Parareal with overlap or methods with more than two levels. For those, the block iteration does not only link two successive block variables with time index $n+1$ and n, but more than two, with time indices $n+1$, n, $n-1$, ... The generating function method can be extended to such cases [14], but we focus here on primary block iterations and leave extensions for future work.

Some iterative PinT algorithms can also consist of combinations of two or more block iterations, as it is the case for PFASST (*cf.* Section 4.3) and STMG (*cf.* Section 5). But we also show in those analyses that it is possible to reduce combinations of several block iterations into one primary block iteration. In this paper, we therefore only focus on primary block iterations for simplicity.

2.3. Generating function and error bound for a block iteration.

Definition 2.8 (Generating function). *The* generating function *associated with a primary block iteration is the power series*

$$\rho_k(\zeta) := \sum_{n=0}^{\infty} e_{n+1}^k \zeta^{n+1} \tag{2.31}$$

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 $\mathbb{C}^M$.

[7]Note that the consistency condition is satisfied even without assumption 2.7.

LEMMA 2.9. *The generating function for the primary block iteration* (2.13) *satisfies*

$$\rho_{k+1}(\zeta) \leq \frac{\gamma + \alpha\zeta}{1 - \beta\zeta}\rho_k(\zeta), \tag{2.32}$$

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

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

$$\boldsymbol{u}_{n+1}^{k+1} - \boldsymbol{u}_{n+1} = \mathbf{B}_0^1(\boldsymbol{u}_{n+1}^k) + \mathbf{B}_1^0\left(\boldsymbol{u}_n^{k+1}\right) + \mathbf{B}_1^1\left(\boldsymbol{u}_n^k\right) - \boldsymbol{\psi}(\boldsymbol{u}_n). \tag{2.33}$$

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

$$\boldsymbol{u}_{n+1}^{k+1} - \boldsymbol{u}_{n+1} = \mathbf{B}_0^1(\boldsymbol{u}_{n+1}^k - \boldsymbol{u}_{n+1}) + \mathbf{B}_1^0\left(\boldsymbol{u}_n^{k+1} - \boldsymbol{u}_n\right) + \mathbf{B}_1^1\left(\boldsymbol{u}_n^k - \boldsymbol{u}_n\right). \tag{2.34}$$

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

$$e_{n+1}^{k+1} \leq \gamma e_{n+1}^k + \beta e_n^{k+1} + \alpha e_n^k \tag{2.35}$$

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

$$\sum_{n=0}^{\infty} e_{n+1}^{k+1}\zeta^{n+1} \leq \gamma\sum_{n=0}^{\infty} e_{n+1}^k\zeta^{n+1} + \beta\sum_{n=0}^{\infty} e_n^{k+1}\zeta^{n+1} + \alpha\sum_{n=0}^{\infty} e_n^k\zeta^{n+1}. \tag{2.36}$$

Using Definition 2.8 and that $e_0^k = 0$ for all k we find

$$\rho_{k+1}(\zeta) \leq \gamma\rho_k(\zeta) + \beta\zeta\sum_{n=1}^{\infty} e_n^{k+1}\zeta^n + \alpha\zeta\sum_{n=1}^{\infty} e_n^k\zeta^n. \tag{2.37}$$

Shifting indices leads to

$$(1 - \beta\zeta)\rho_{k+1}(\zeta) \leq (\gamma + \alpha\zeta)\rho_{k+1}(\zeta) \tag{2.38}$$

and concludes the proof. ☐

THEOREM 2.10. *Consider the primary block iteration* (2.13) *and let*

$$\delta := \max_{n\in[\![1,N]\!]} \left\|\boldsymbol{u}_n^0 - \boldsymbol{u}_n\right\| \tag{2.39}$$

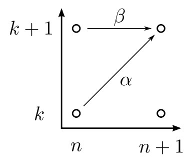
be the maximum error of the initial guess over all blocks. Then, using the notations of Lemma 2.9, *we have for* $k > 0$

$$e_{n+1}^k \leq \theta_{n+1}^k(\alpha, \beta, \gamma)\delta, \tag{2.40}$$

where θ_{n+1}^k *is a bounding function defined as follows:*

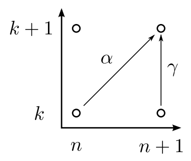
- *if* only $\gamma = 0$, *then*

$$\theta_{n+1}^k = \frac{\alpha^k}{(k-1)!}\sum_{i=0}^{n-k}\prod_{l=1}^{k-1}(i+l)\beta^i, \tag{2.41}$$



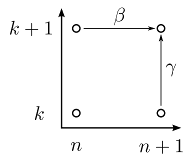
- *if* only $\beta = 0$, *then*

$$\theta^k_{n+1} = \begin{cases} (\gamma+\alpha)^k \text{ if } k \le n, \\ \delta\gamma^k \sum_{i=0}^{n} \binom{k}{i} \left(\frac{\alpha}{\gamma}\right)^i \text{ otherwise,} \end{cases} \tag{2.42}$$



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

$$\theta^k_{n+1} = \frac{\gamma^k}{(k-1)!} \sum_{i=0}^{n} \prod_{l=1}^{k-1} (i+l)\beta^i, \tag{2.43}$$



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

$$\theta^k_{n+1} = \gamma^k \sum_{i=0}^{\min(n,k)} \sum_{l=0}^{n-i} \binom{k}{i} \binom{l+k-1}{l} \left(\frac{\alpha}{\gamma}\right)^i \beta^l. \tag{2.44}$$

The proof uses Lemma 2.9, bounds the generating function at iterate $k = 0$ by

$$\rho_0(\zeta) \le \delta \sum_{n=0}^{\infty} \zeta^{n+1}, \tag{2.45}$$

which covers arbitrary initial guesses for defining starting values $\boldsymbol{u}^0_n$ for each block. For specific initial guesses, $\rho_0(\zeta)$ can be bounded differently, see e.g. [13, Proof of Th. 1]. The error bound is then computed by coefficient identification after expansion into power series: the rather technical proof can be found in Appendix A.

In some numerical examples shown below, we find that the estimate from Theorem 2.10 in volume can be not quite sharp, *cf.* Section 4.2.3. If the last time point of the blocks coincides with the right bound of the sub-interval, it is helpful to define the *interface error* at the right boundary point of the n^{th} block as

$$\bar{e}^k_{n+1} := |\bar{u}^k_{n+1} - \bar{u}_{n+1}|, \tag{2.46}$$

where $\bar{u}$ is the last element of the block variable $\boldsymbol{u}$. We then multiply (2.34) by $e^T_M = [0, \dots, 0, 1]$ to get

$$e^T_M(\boldsymbol{u}^{k+1}_{n+1} - \boldsymbol{u}_{n+1}) = \boldsymbol{b}^1_0(\boldsymbol{u}^{k+1}_{n+1} - \boldsymbol{u}_{n+1}) + \boldsymbol{b}^0_1(\boldsymbol{u}^{k+1}_{n} - \boldsymbol{u}_{n}) + \boldsymbol{b}^1_1(\boldsymbol{u}^{k}_{n} - \boldsymbol{u}_{n}), \tag{2.47}$$

where $\boldsymbol{b}^j_i$ is the last line of the block operator $\mathbf{B}^j_i$. Taking the absolute value on both sides, we recognize the interface error $\bar{e}^{k+1}_{n+1}$ on the left hand side. By neglecting the error from interior points and using the triangle inequality, we get the approximation[8]

$$\bar{e}^{k+1}_{n+1} \approx \bar{\gamma}\bar{e}^k_{n+1} + \bar{\beta}\bar{e}^{k+1}_n + \bar{\alpha}\bar{e}^k_n, \tag{2.48}$$

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

COROLLARY 2.11 (Interface error approximation). *Defining for the initial interface error the bound* $\bar{\delta} := \max_{n\in[\![1,N]\!]} \left\|\bar{u}^0_n - \bar{u}_n\right\|$, *we obtain for the interface error the approximation*

$$\bar{e}^{k+1}_{n+1} \approx \bar{\theta}^k_{n+1} := \theta^k_{n+1}(\bar{\alpha}, \bar{\beta}, \bar{\delta}), \tag{2.49}$$

with θ^k_{n+1} *defined in Theorem 2.10.*

[8]For an interface block iteration ($M = 1, \tau_1 = 1$), (2.48) becomes a strict inequality and thus Corollary 2.11 a rigorous convergence bound.

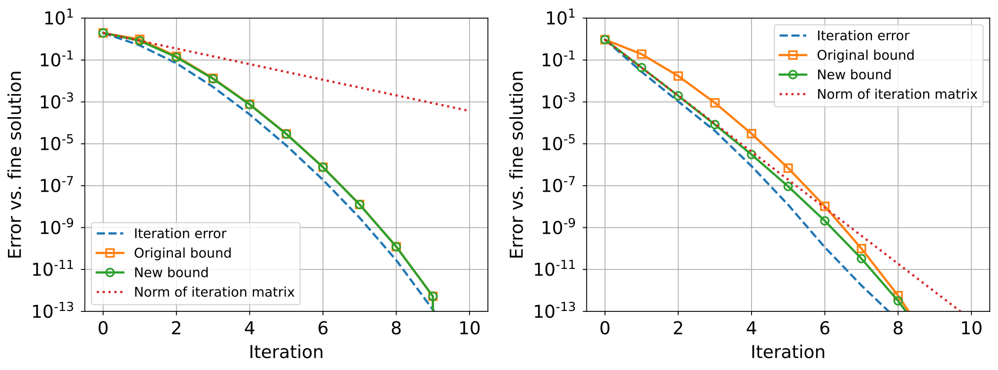


FIG. 3. *Comparison of error bounds for* PARAREAL *with two different* λ *values for* (2.1). *Left:* $\lambda = i$, *right:* $\lambda = -1$; *the error is computed with the* L^∞ *norm.*

Proof. Result follows as in the proof of Lemma 2.9 using approximate relations.∎

3. Writing Parareal as block iteration.

In the convergence analysis of PARAREAL for non-linear problems in [13, Th. 1], a double recurrence of the form $e_{n+1}^{k+1} \leq \alpha e_n^k + \beta e_n^{k+1}$ is obtained for the error, where α and β come from Lipschitz constants and local truncation error bounds. If we use the PARAREAL setting described in Section 2.2.1 for the Dahlquist model problem, where $\alpha = \|\mathcal{F} - \mathcal{G}\|$ and $\beta = \|\mathcal{G}\|$, [13, Th. 1] gives the bound

$$e_{n+1}^k \leq \delta \frac{\alpha^k}{k!} \bar{\beta}^{n-k} \prod_{l=1}^{k} (n+1-l), \quad \bar{\beta} = \max(1, \beta). \tag{3.1}$$

This is different from our new error bound

$$e_{n+1}^k \leq \delta \frac{\alpha^k}{(k-1)!} \sum_{i=0}^{n-k} \prod_{l=1}^{k-1} (i+l) \beta^i \tag{3.2}$$

when applying Theorem 2.10 with $\gamma = 0$ to the block iteration of PARAREAL. The difference stems from an upper bound approximation in the proof of [13, Th. 1] which leads to the simpler bound in (3.1). The two bounds are equal when $\beta = 1$, but for $\beta \neq 1$, the error bound in (3.2) is sharper. To illustrate this, we use the interface formulation of Section 2.1.1 for PARAREAL from Section 2.2.1. We set $M := 1$, $\tau_1 := 1$ and use the block operators

$$\boldsymbol{\phi} := R(\lambda \Delta t/\ell)^{-\ell}, \quad \boldsymbol{\chi} := 1, \quad \boldsymbol{\phi}_{\boldsymbol{\Delta}} := R_\Delta(\lambda \Delta t/\ell_\Delta)^{-\ell_\Delta}. \tag{3.3}$$

We solve (2.1) for $\lambda \in \{i, -1\}$ with $t \in [0, 2\pi]$ using $L := 10$ blocks, $\ell := 10$ fine time steps per block with the standard 4^{th} order Runge-Kutta method for $\boldsymbol{\phi}$ and $\ell_\Delta = 5$ coarse time steps per block with Forward Euler for $\boldsymbol{\phi}_{\boldsymbol{\Delta}}$[9]. Figure 3 shows the resulting error (dashed line), the original error bound (3.1), and the new bound (3.2). We also plot the linear bound obtained from the L^∞ norm of the iteration matrix

$$\mathbf{R}_{\text{PARAREAL}} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A}, \quad \mathbf{M} = \begin{pmatrix} \boldsymbol{\phi} & & \\ -\boldsymbol{\phi}\boldsymbol{\phi}_{\boldsymbol{\Delta}}^{-1}\boldsymbol{\chi} & \boldsymbol{\phi} & \\ & \ddots & \ddots \end{pmatrix} \tag{3.4}$$

[9]This version of PARAREAL can also be seen as a particular two-level method, which is covered in Section 5.

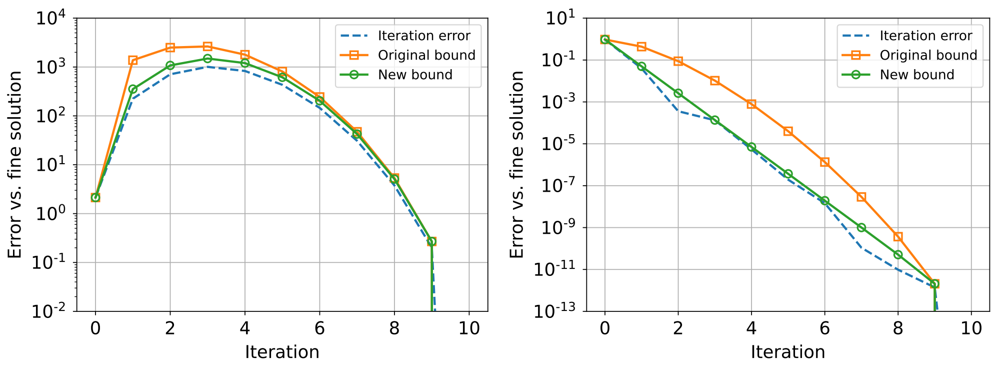


FIG. 4. *Comparison of error bounds for* PARAREAL *with two different λ values for* (2.1)*. Left: $\lambda = 4i$, right: $\lambda = -4$; the error is computed with the L^∞ norm.*

of the global iteration (2.17). 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 super-linear 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 converges well for imaginary $\lambda = i$ since the coarse solver uses $\bar{\ell}_\Delta = 50$ points per wavelength. Figure 4 shows the same setup for a four times larger λ and we see the well documented deterioration of convergence for imaginary λ/hyperbolic problems [18, 15], also well captured by the bounds, while Parareal still converges quickly for larger negative real λ.

4. Writing PFASST and related algorithms as block iteration.

Spectral Deferred Corrections (SDC) [7] are an essential ingredient of PFASST. We provide a simplified description of SDC for the Dahlquist problem (2.1) and the steps leading to PFASST.

4.1. 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 transmission operators

$$\boldsymbol{\phi} := (\mathbf{I} - \mathbf{Q}), \quad \boldsymbol{\chi} := \mathbf{H} \quad \Longrightarrow \quad (\mathbf{I} - \mathbf{Q})\boldsymbol{u}_{n+1} = \mathbf{H}\boldsymbol{u}_n. \tag{4.1}$$

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

$$\mathbf{Q}_\Delta = \lambda \Delta t \left(\int_0^{\tau_m} l_{\Delta,j}(s) ds \right), \tag{4.2}$$

where $l_{\Delta,j}$ is an approximation of the Lagrange polynomial l_j. Usually, $\mathbf{Q}_\Delta$ is lower triangular [35, Sec 3] and easy to invert. This approximation is used to build the preconditioned iteration

$$\boldsymbol{u}_{n+1}^{k+1} = \boldsymbol{u}_{n+1}^{k} + [\mathbf{I} - \mathbf{Q}_\Delta]^{-1} \left(\mathbf{H}\boldsymbol{u}_n - (\mathbf{I} - \mathbf{Q})\boldsymbol{u}_{n+1}^{k} \right) \tag{4.3}$$

for solving (4.1). Setting $\boldsymbol{\phi}_{\boldsymbol{\Delta}} := \mathbf{I} - \mathbf{Q}_\Delta$, we see that SDC solves the global problem (2.12) block by block. To invert the $\boldsymbol{\phi}$ operator in each block, it uses the preconditioned iteration

$$\boldsymbol{u}_{n+1}^{k+1} = \left[\mathbf{I} - \boldsymbol{\phi}_{\boldsymbol{\Delta}}^{-1} \boldsymbol{\phi} \right] \boldsymbol{u}_{n+1}^{k} + \boldsymbol{\phi}_{\boldsymbol{\Delta}}^{-1} \boldsymbol{\chi} \boldsymbol{u}_n. \tag{4.4}$$

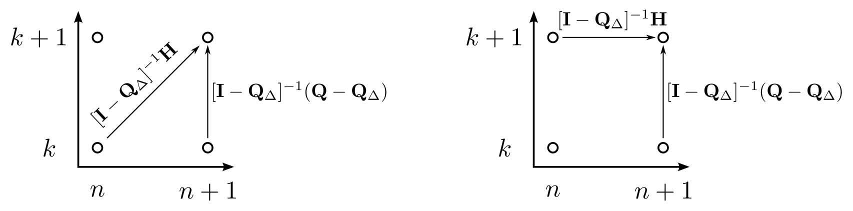

$k+1$
k
n
$n+1$
$[\mathbf{I}-\mathbf{Q}_\Delta]^{-1}\mathbf{H}$
$[\mathbf{I}-\mathbf{Q}_\Delta]^{-1}(\mathbf{Q}-\mathbf{Q}_\Delta)$
$k+1$
k
n
$n+1$
$[\mathbf{I}-\mathbf{Q}_\Delta]^{-1}\mathbf{H}$
$[\mathbf{I}-\mathbf{Q}_\Delta]^{-1}(\mathbf{Q}-\mathbf{Q}_\Delta)$

FIG. 5. *kn-graphs for Block Jacobi SDC (left) and Block Gauss-Seidel SDC (right).*

SDC runs sequentially block by block. The key idea of PFASST is to solve the whole system (2.12) at once while introducing concurrency via specific global preconditioners that we describe next.

4.2. Algorithmic components for PFASST.

4.2.1. Approximate Block Jacobi with SDC. 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 $\boldsymbol{\phi}_{\boldsymbol{\Delta}}$ as introduced above. This leads to the *global* preconditioned iteration

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \mathbf{P}_{Jac}^{-1}(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k), \quad \mathbf{P}_{Jac} = \begin{bmatrix} \boldsymbol{\phi}_{\boldsymbol{\Delta}} & & \\ & \boldsymbol{\phi}_{\boldsymbol{\Delta}} & \\ & & \ddots \end{bmatrix}. \tag{4.5}$$

This is equivalent to the block Jacobi relaxation in Section 2.2.2 with $\omega = 1$, except that the block operator $\boldsymbol{\phi}$ is approximated by $\boldsymbol{\phi}_{\boldsymbol{\Delta}}$. Using the SDC block operators (4.1) gives the block updating formula

$$\boldsymbol{u}_{n+1}^{k+1} = \boldsymbol{u}_{n+1}^k + [\mathbf{I} - \mathbf{Q}_\Delta]^{-1}\left(\mathbf{H}\boldsymbol{u}_n^k - (\mathbf{I} - \mathbf{Q})\boldsymbol{u}_{n+1}^k\right), \tag{4.6}$$

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

$$\begin{aligned} \mathbf{B}_0^1 &:= \mathbf{I} - [\mathbf{I} - \mathbf{Q}_\Delta]^{-1}(\mathbf{I} - \mathbf{Q}) = [\mathbf{I} - \mathbf{Q}_\Delta]^{-1}(\mathbf{Q} - \mathbf{Q}_\Delta), \\ \mathbf{B}_1^1 &:= [\mathbf{I} - \mathbf{Q}_\Delta]^{-1}\mathbf{H}, \quad \mathbf{B}_1^0 := 0. \end{aligned} \tag{4.7}$$

Its kn-graph is shown in Figure 5 (left). This block iteration can be written in the more generic form

$$\boldsymbol{u}_{n+1}^{k+1} = \left[\mathbf{I} - \boldsymbol{\phi}_{\boldsymbol{\Delta}}^{-1}\boldsymbol{\phi}\right]\boldsymbol{u}_{n+1}^k + \boldsymbol{\phi}_{\boldsymbol{\Delta}}^{-1}\boldsymbol{\chi}\boldsymbol{u}_n^k. \tag{4.8}$$

This is similar to (4.4) except that we use the current iterate $\boldsymbol{u}_n^k$ from the previous block and not the converged solution $\boldsymbol{u}_n$. Since this block iteration does not explicitly depend on the use of SDC, we denote it as *Approximate Block Jacobi* (ABJ).

4.2.2. Approximate Block Gauss-Seidel with SDC. We can also consider Block Gauss-Seidel type preconditioning

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \tilde{\mathbf{P}}_{GS}^{-1}(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k), \quad \tilde{\mathbf{P}}_{GS} = \begin{bmatrix} \boldsymbol{\phi}_{\boldsymbol{\Delta}} & & \\ -\boldsymbol{\chi} & \boldsymbol{\phi}_{\boldsymbol{\Delta}} & \\ & \ddots & \ddots \end{bmatrix} \tag{4.9}$$

for the global problem (2.12). Similar to PARAREAL, this is an approximate Block Gauss-Seidel preconditioner, except that here the diagonal blocks are approximated, instead of the lower-diagonal blocks as in Parareal.[10] Using the SDC block operators gives the block updating formula

$$\boldsymbol{u}_{n+1}^{k+1} = \boldsymbol{u}_{n+1}^{k} + [\mathbf{I} - \mathbf{Q}_\Delta]^{-1} \left(\mathbf{H}\boldsymbol{u}_{n}^{k+1} - (\mathbf{I} - \mathbf{Q})\boldsymbol{u}_{n+1}^{k}\right), \tag{4.10}$$

which we call *Block Gauss-Seidel SDC*. This is again a primary block iteration with

$$\begin{aligned} &\mathbf{B}_0^1 := \mathbf{I} - [\mathbf{I} - \mathbf{Q}_\Delta]^{-1}(\mathbf{I} - \mathbf{Q}) = [\mathbf{I} - \mathbf{Q}_\Delta]^{-1}(\mathbf{Q} - \mathbf{Q}_\Delta), \\ &\mathbf{B}_1^1 := 0, \quad \mathbf{B}_1^0 := [\mathbf{I} - \mathbf{Q}_\Delta]^{-1}\mathbf{H}. \end{aligned} \tag{4.11}$$

Its kn-graph is shown in Figure 5 (right). For generic $\boldsymbol{\phi}$ and $\boldsymbol{\phi}_{\boldsymbol{\Delta}}$ we get

$$\boldsymbol{u}_{n+1}^{k+1} = \left[\mathbf{I} - \boldsymbol{\phi}_{\boldsymbol{\Delta}}^{-1}\boldsymbol{\phi}\right] \boldsymbol{u}_{n+1}^{k} + \boldsymbol{\phi}_{\boldsymbol{\Delta}}^{-1}\boldsymbol{\chi}\boldsymbol{u}_{n}^{k+1}. \tag{4.12}$$

This formula uses the updated iterate $\boldsymbol{u}_n^{k+1}$ in contrast to (4.4) which uses the converged solution $\boldsymbol{u}_n$ or (4.8) which uses the old iterate $\boldsymbol{u}_n^k$. Since this block iteration does not explicitly depend on the use of SDC, we denote it as *Approximate Block Gauss-Seidel* (ABGS).

4.2.3. Analysis and numerical experiments. Since Block Jacobi SDC can be written as primary block iteration, we can directly apply Theorem 2.10 with $\beta = 0$ to get the error bound

$$e_{n+1}^k \leq \begin{cases} \delta(\gamma+\alpha)^k \text{ if } k \leq n \\ \delta\gamma^k \displaystyle\sum_{i=0}^{n} \binom{k}{i} \left(\frac{\alpha}{\gamma}\right)^i \text{ otherwise,} \end{cases} \tag{4.13}$$

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 in our case. 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 the 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, applying Theorem 2.10 gives with $\alpha = 0$

$$e_{n+1}^k \leq \delta \frac{\gamma^k}{(k-1)!} \sum_{i=0}^{n} \prod_{l=1}^{k-1} (i+l)\beta^i, \tag{4.14}$$

where $\gamma := \left\|[\mathbf{I} - \mathbf{Q}_\Delta]^{-1}(\mathbf{Q} - \mathbf{Q}_\Delta)\right\|$, $\beta := \left\|[\mathbf{I} - \mathbf{Q}_\Delta]^{-1}\mathbf{H}\right\|$. Here the algorithm contracts already in early iterations if γ is small enough. Since the γ coefficient is the same for both Block Gauss-Seidel SDC and Block Jacobi-SDC, both algorithms have a similar convergence rate when they contract.

We illustrate this with the following example. Let $\lambda := i$, $u_0 := 1$, and let the time interval $[0, \pi]$ be divided into $L = 10$ sub-intervals. Inside each sub-interval, we use one step of the collocation method from Section 2.1.2 with $M := 10$ Lobatto quadrature nodes of a Legendre distribution [19]. This gives us block variables of size $M = 10$ and we chose $\mathbf{Q}_\Delta$ as the matrix defined by a Backward Euler step to build the $\boldsymbol{\phi}_{\boldsymbol{\Delta}}$ operator. The starting value $\boldsymbol{u}^0$ for the iteration is initialized with random numbers starting from the same seed. Figure 6 (left) shows the numerical error for the last block using the L^∞ norm, the bound obtained with the GFM method and

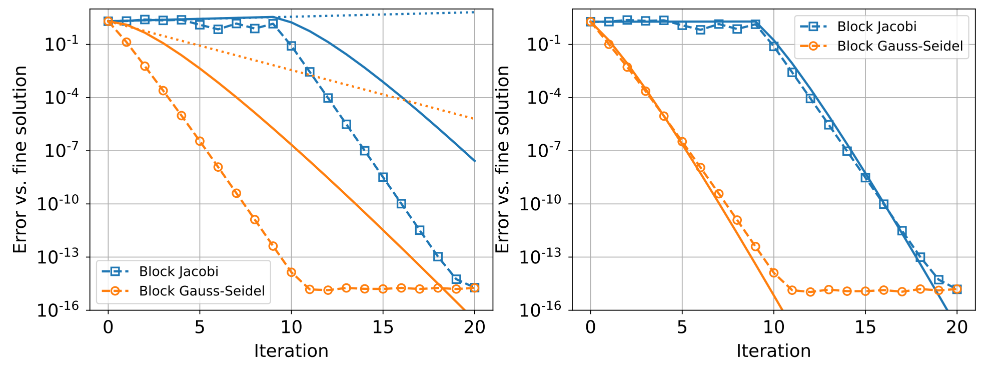


FIG. 6. *Comparison of numerical errors with GFM bounds for Block Jacobi SDC and Block Gauss-Seidel SDC. Left : error on the block variables (dashed), GFM bounds (solid), linear bound from the iteration matrix (dotted). Right : error estimate using the interface approximation from Corollary 2.11.*

the linear bound using the norm of the global iteration matrix. As for PARAREAL in Section 3, the GFM bound is similiar to the iteration matrix bound for the first few iterations, but much tighter for the 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 2.11 gives a very good estimate of the error at the interface, see Figure 6 (right).

4.3. PFASST. We give a simplified description of PFASST [8] applied to the Dahlquist problem (2.1). While we consider only the two-level variant, the algorithm can use more levels [8, 38]. In a nutshell, one PFASST iteration is the combination of a block Jacobi SDC iteration on the fine level followed by a block Gauss-Seidel SDC iteration on the coarse level [8, Sec. 3.2].[11] In particular, this corresponds to doing only one SDC sweep on the coarse level. To write PFASST as a block iteration, we first describe the coarse level as in Section 2.2.3. From that we can form the $\tilde{\mathbf{Q}}$ quadrature matrix associated to the coarse nodes and the coarse $\tilde{\mathbf{H}}$ matrix. This leads to the definition of the $\tilde{\boldsymbol{\phi}}$ and $\tilde{\boldsymbol{\chi}}$ operators for the coarse level, combined with the transfer operators $\mathbf{T}_F^C$ and $\mathbf{T}_C^F$. Then, we approximate $\tilde{\mathbf{Q}}$ by a $\tilde{\mathbf{Q}}_\Delta$ matrix, which allows us to define an approximate operator $\tilde{\boldsymbol{\phi}}_{\boldsymbol{\Delta}}$ for $\tilde{\boldsymbol{\phi}}$ on the coarse level.

First, we use the iteration formula for Block Jacobi SDC from Section 4.2.1 as update for the intermediary solution (denoted with iteration index $k + 1/2$),

$$\boldsymbol{u}_{n+1}^{k+1/2} = [\mathbf{I} - \mathbf{Q}_\Delta]^{-1}(\mathbf{Q} - \mathbf{Q}_\Delta)\boldsymbol{u}_{n+1}^k + [\mathbf{I} - \mathbf{Q}_\Delta]^{-1}\mathbf{H}\boldsymbol{u}_n^k. \tag{4.15}$$

Then, as described in [3, Sec. 2.2], the block Gauss-Seidel iteration applied to the

[10] Note that an exact Block Gauss-Seidel preconditioner for (2.12) uses the original block triangular matrix $\mathbf{A}$, and thus converges in one iteration by integrating all blocks with $\boldsymbol{\phi}$ sequentially.

[11] A different order is given in [3, Sec. 2.2] (first Block Gauss-Seidel, then Block Jacobi), but both can be seen as the same iteration, only differing by the associated initialization process. Here we use the order that is more convenient for analysis.

coarse level can be written as a preconditioned iteration for the global system (2.12)

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^{k+1/2} + \bar{\mathbf{T}}_C^F \tilde{\mathbf{P}}_{GS}^{-1} \bar{\mathbf{T}}_F^C (\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^{k+1/2}), \quad \tilde{\mathbf{P}}_{GS} = \begin{bmatrix} \tilde{\boldsymbol{\phi}}_{\boldsymbol{\Delta}} & & \\ -\tilde{\boldsymbol{\chi}} & \tilde{\boldsymbol{\phi}}_{\boldsymbol{\Delta}} & \\ & \ddots & \ddots \end{bmatrix}. \tag{4.16}$$

This is the same iteration as we obtained for the coarse grid correction in Section 2.2.3, except that the coarse operator $\tilde{\boldsymbol{\phi}}$ has been replaced by $\tilde{\boldsymbol{\phi}}_{\boldsymbol{\Delta}}$. Assumption 2.7 holds, since the use of Lobatto or Radau-II nodes leads to the specific form of the $\mathbf{H}$ matrix in (2.11), which implies

$$\boldsymbol{\Delta}_{\boldsymbol{\chi}} = \mathbf{T}_F^C \mathbf{H} - \tilde{\mathbf{H}} \mathbf{T}_F^C = 0. \tag{4.17}$$

Using similar computations and the block operators defined for collocation and SDC (*cf.* Section 2.1.2 and Section 4.1) we obtain the block iteration

$$\boldsymbol{u}_{n+1}^{k+1} = [\mathbf{I} - \mathbf{T}_C^F (\mathbf{I} - \tilde{\mathbf{Q}}_\Delta)^{-1} \mathbf{T}_F^C (\mathbf{I} - \mathbf{Q})] \boldsymbol{u}_{n+1}^{k+1/2} + \mathbf{T}_C^F (\mathbf{I} - \tilde{\mathbf{Q}}_\Delta)^{-1} \mathbf{T}_F^C \mathbf{H} \boldsymbol{u}_n^{k+1} \tag{4.18}$$

by substitution into (2.30). Finally, the combination of the two gives

$$\begin{aligned} \boldsymbol{u}_{n+1}^{k+1} = {} & (\mathbf{I} - \mathbf{T}_C^F \tilde{\boldsymbol{\phi}}_{\boldsymbol{\Delta}}^{-1} \mathbf{T}_F^C \boldsymbol{\phi}) [\mathbf{I} - \mathbf{Q}_\Delta]^{-1} (\mathbf{Q} - \mathbf{Q}_\Delta) \boldsymbol{u}_{n+1}^k \\ & + (\mathbf{I} - \mathbf{T}_C^F [\mathbf{I} - \tilde{\mathbf{Q}}_\Delta]^{-1} \mathbf{T}_F^C (\mathbf{I} - \mathbf{Q})) [\mathbf{I} - \mathbf{Q}_\Delta]^{-1} \mathbf{H} \boldsymbol{u}_n^k \\ & + \mathbf{T}_C^F (\mathbf{I} - \tilde{\mathbf{Q}}_\Delta)^{-1} \tilde{\mathbf{H}} \mathbf{T}_F^C \boldsymbol{u}_n^{k+1}. \end{aligned} \tag{4.19}$$

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

$$\begin{aligned} \boldsymbol{u}_{n+1}^{k+1} = {} & [\mathbf{I} - \mathbf{T}_C^F \tilde{\boldsymbol{\phi}}_{\boldsymbol{\Delta}}^{-1} \mathbf{T}_F^C \boldsymbol{\phi}] (\mathbf{I} - \boldsymbol{\phi}_{\boldsymbol{\Delta}}^{-1} \boldsymbol{\phi}) \boldsymbol{u}_{n+1}^k \\ & + (\mathbf{I} - \mathbf{T}_C^F \tilde{\boldsymbol{\phi}}_{\boldsymbol{\Delta}}^{-1} \mathbf{T}_F^C \boldsymbol{\phi}) \boldsymbol{\phi}_{\boldsymbol{\Delta}}^{-1} \boldsymbol{\chi} \boldsymbol{u}_n^k + \mathbf{T}_C^F \tilde{\boldsymbol{\phi}}_{\boldsymbol{\Delta}}^{-1} \mathbf{T}_F^C \boldsymbol{\chi} \boldsymbol{u}_n^{k+1}. \end{aligned} \tag{4.20}$$

This is again a primary block iteration, but in contrast to the previously described block iterations, all block operators are non-zero.

4.3.1. Analysis and numerical experiments. Applying Theorem 2.10 to (4.19) gives for PFASST the error bound

$$e_{n+1}^k \leq \delta \gamma^k \sum_{i=0}^{\min(n,k)} \sum_{l=0}^{n-i} \binom{k}{i} \binom{l+k-1}{l} \left(\frac{\alpha}{\gamma}\right)^i \beta^l, \tag{4.21}$$

with $\gamma := ||(\mathbf{I} - \mathbf{T}_C^F \tilde{\boldsymbol{\phi}}_{\boldsymbol{\Delta}}^{-1} \mathbf{T}_F^C \boldsymbol{\phi})(\mathbf{I} - \boldsymbol{\phi}_{\boldsymbol{\Delta}}^{-1} \boldsymbol{\phi})||$, $\beta := ||\mathbf{T}_C^F (\mathbf{I} - \tilde{\mathbf{Q}}_\Delta)^{-1} \tilde{\mathbf{H}} \mathbf{T}_F^C||$, 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 $L = 10$ sub-intervals. Inside each sub-interval we use $M := 6$ Lobatto-Legendre nodes on the fine level and $\tilde{M} := 2$ Lobatto nodes on the coarse level. The $\mathbf{Q}_\Delta$ and $\tilde{\mathbf{Q}}_\Delta$ operators are defined with Backward Euler. Figure 7 (left) compares the measured numerical error with the GFM bound and the linear bound from the iteration matrix. As in Section 4.2.3, 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. As shown in Figure 7 (right) we get a significantly better estimation of the numerical error when we use the interface approximation from Corollary 2.11. For the GFM bound we have $(\alpha, \beta, \gamma) = (0.16, 1, 0.19)$,

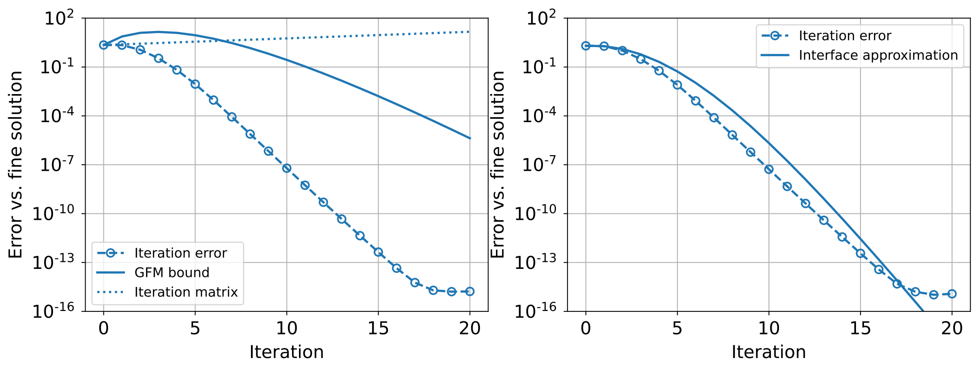


FIG. 7. *Comparison of numerical errors with GFM bounds for* PFASST. *Left : error bound using volume values. Right : estimate using the interface approximation.*

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 where $\gamma = 0$. This similarity between PFASST and PARAREAL will be discussed in more detail in the next section.

5. Writing two-level time multi-grid as block iteration. The idea of time multi-grid goes back to the 1980s and 1990s [4, 22, 32]. PARAREAL itself can be interpreted as a time multi-grid method [18], and this insight inspired various developments of multi-level methods, in particular MGRIT and STMG. In this section, we show how to write two-level algorithms applied to the Dahlquist problem as block iterations. Since we do not consider spatial dimensions, we refer to this class of methods as Time Multi-Grid (TMG). In particular, we will show that all the other two-level iterative methods, including PFASST, can be expressed as TMG. The extension of this analysis to more levels and comparison with multi-level MGRIT is left for future work.

Gander and Neumüller introduce STMG for discontinuous Galerkin approximations in time [17], which leads to a similar system as (2.12). We describe the two-level approach for general time discretizations, following this multi-level description [17, Sec. 3]. Consider a coarse problem defined as in Section 2.2.3 and a damped block Jacobi smoother as in Section 2.2.2 with a given relaxation parameter ω. Then, a two-level TMG iteration requires the following steps:

1. ν_1 pre-relaxation steps with block Jacobi smoother, see Section 2.2.2,
2. a coarse correction using the exact coarse grid operators,
3. ν_2 steps with the block Jacobi smoother, see Section 2.2.2,

each corresponding to a block iteration. 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 is obtained when

- Assumption 2.7 is verified, which implies $\boldsymbol{\Delta_\chi} = 0$,
- only one pre-relaxation step is used, $\nu_1 = 1$,
- and no post-relaxation step is considered, $\nu_2 = 0$.

In this case, the two-level iteration reduces to the two block updates

$$\boldsymbol{u}_{n+1}^{k+1/2} = (1-\omega)\boldsymbol{u}_{n+1}^{k} + \omega\boldsymbol{\phi}^{-1}\boldsymbol{\chi}\boldsymbol{u}_{n}^{k}, \tag{5.1}$$

$$\boldsymbol{u}_{n+1}^{k+1} = \left(\mathbf{I} - \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}^{-1}\mathbf{T}_F^C\boldsymbol{\phi}\right)\boldsymbol{u}_{n+1}^{k+1/2} + \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}^{-1}\tilde{\boldsymbol{\chi}}\mathbf{T}_F^C\boldsymbol{u}_{n}^{k+1}, \tag{5.2}$$

using $k+1/2$ as intermediate index. Combining (5.1) and (5.2) leads to the primary block iteration

$$\boldsymbol{u}_{n+1}^{k+1} = \left(\mathbf{I} - \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}^{-1}\mathbf{T}_F^C\boldsymbol{\phi}\right)\left[(1-\omega)\boldsymbol{u}_{n+1}^{k} + \omega\boldsymbol{\phi}^{-1}\boldsymbol{\chi}\boldsymbol{u}_{n}^{k}\right] + \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}^{-1}\tilde{\boldsymbol{\chi}}\mathbf{T}_F^C\boldsymbol{u}_{n}^{k+1}. \tag{5.3}$$

If $\omega \neq 1$, then all block operators in this primary block iteration are non-zero, and applying Theorem 2.10 leads to a similar convergence bound as the one obtained for PFASST in Section 4.3.1.

For $\omega = 1$ we get the simplified iteration

$$\boldsymbol{u}_{n+1}^{k+1} = \left(\boldsymbol{\phi}^{-1}\boldsymbol{\chi} - \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}^{-1}\mathbf{T}_F^C\boldsymbol{\chi}\right)\boldsymbol{u}_{n}^{k} + \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}^{-1}\tilde{\boldsymbol{\chi}}\mathbf{T}_F^C\boldsymbol{u}_{n}^{k+1}, \tag{5.4}$$

and the following result:

PROPOSITION 5.1. *Consider a coarse grid correction as in Definition 2.2.3, such that the prolongation and restriction operators (in time) satisfy Assumption 2.6. If Assumption 2.7 is also verified and only one Jacobi pre-relaxation step with parameter $\omega = 1$ is used before the coarse grid correction, then the two-level* TMG *is equivalent to a* PARAREAL *method, where the coarse solver $\mathcal{G}$ uses the same time integrator as the fine solver $\mathcal{F}$ but with larger time steps, i.e.*

$$\mathcal{G} := \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}^{-1}\mathbf{T}_F^C\boldsymbol{\chi}. \tag{5.5}$$

This is the same result as [18, Theorem 3.1] but presented here in the context of our GFM framework and the definition of PARAREAL given in Section 2.2.1. In particular, it shows that the simplified two-grid iteration on (2.12) is equivalent to the preconditioned fixed-point iteration (2.17) of PARAREAL, if some preliminary conditions are met and the notation $\boldsymbol{\phi}_{\boldsymbol{\Delta}}^{-1} := \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}^{-1}\mathbf{T}_F^C$ is used for the approximate integration operator[12]. However, PARAREAL is usually viewed as an iteration acting on solution values located at the block interfaces, while the two-grid correction as defined here acts on values inside the block, namely volume values.

The second important aspect of this analysis lays in the similarity between this simplified two-level TMG with $\omega = 1$, and the PFASST algorithm as described in Section 4.3. We have the following result:

PROPOSITION 5.2. *The two-level (linear)* PFASST *algorithm is equivalent to a two-level* TMG *satisfying the following conditions:*

1. *Assumptions 2.6 and 2.7 are verified,*
2. TMG *combines one Jacobi smoothing with a coarse grid correction s.t:*
   (a) *the relaxation uses $\omega = 1$ and an approximate integration operator $\boldsymbol{\phi}_{\boldsymbol{\Delta}}$,*
   (b) *the coarse grid correction uses an approximate integration operator $\tilde{\boldsymbol{\phi}}_{\boldsymbol{\Delta}}$.*

This result is equivalent to [2, Theorem 1], but presented here in the GFM framework context. It puts TMG and PFASST in a common group of *elementary two-level time multigrid* methods, which differs on the use (or not) of approximate integration operators during relaxation or coarse grid correction. We summarize those differences

[12]Note that, even if $\mathbf{T}_C^F\tilde{\boldsymbol{\phi}}^{-1}\mathbf{T}_F^C$ is not invertible, this abuse of notation is possible as (2.17) requires an approximation of $\boldsymbol{\phi}^{-1}$ rather than an approximation of $\boldsymbol{\phi}$ itself.

Correction \ Relaxation	Exact $\boldsymbol{\phi}$	Approximate $\boldsymbol{\phi}_{\boldsymbol{\Delta}}$
Exact $\tilde{\boldsymbol{\phi}}$	TMG ($\omega = 1$)	TMG$_f$
Approximate $\tilde{\boldsymbol{\phi}}_{\boldsymbol{\Delta}}$	TMG$_c$	PFASST

Table 1
Classification of the elementary two-level time multigrid methods.

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

Table 2
Summary of all the methods we analyzed, and their block iteration operators. Note that TMG with $\omega = 1$ and TMG$_c$ *correspond to* Parareal *with some specific coarsening.*

in Table 1, and add also two composite approaches that naturally result from this distinction:

- a two level TMG where *only* the coarse correction uses an approximate integration operator $\tilde{\boldsymbol{\phi}}_{\boldsymbol{\Delta}}$ (TMG$_c$),
- a two level TMG where *only* the relaxation on the fine level uses an approximate integration operator $\boldsymbol{\phi}_{\boldsymbol{\Delta}}$ (TMG$_f$).

This gives us then four different two-level algorithms, and we compare their block iteration and convergence in Section 6. Note that the TMG$_c$ algorithm can, similar to TMG, be described as a Parareal algorithm using an approximate integration operator and larger time step for the coarse propagator, that is

$$\mathcal{G} := \mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_{\boldsymbol{\Delta}}^{-1}\mathbf{T}_F^C\boldsymbol{\chi} \tag{5.6}$$

As a matter of fact, the version of Parareal used for numerical experiments in Section 3 is equivalent to TMG$_c$.

6. Comparison of iterative PinT algorithms.

We summarize all methods analyzed in this paper in Table 2. After choosing a given block decomposition and associated block operators, one just has to define approximate integration operators, and possibly a coarse block discretization, to build any of the algorithms.

We illustrate this for the Dahlquist problem with $\lambda := 2i - 0.2$, $u_0 = 1$ and decompose the simulation interval $[0, 2\pi]$ into $N = 10$ sub-intervals. We choose a block discretization with $M = 5$ points and $\tilde{M} = 3$ for the coarse block, with Lobatto-Legendre points for both levels. We use one step of a collocation method on each block for the fine integrator (see Section 2.1.2) and a Backward Euler discretization for the approximate block operators on each level, as for SDC (see Section 4.1). In Table 3, we give the maximum absolute error in time for each possible propagator, if

	$\boldsymbol{\phi}^{-1}\boldsymbol{\chi}$	$\boldsymbol{\phi}_{\boldsymbol{\Delta}}^{-1}\boldsymbol{\chi}$	$\mathbf{T}_C^F\tilde{\boldsymbol{\phi}}^{-1}\mathbf{T}_F^C\boldsymbol{\chi}$	$\mathbf{T}_C^F\tilde{\boldsymbol{\phi}}_{\boldsymbol{\Delta}}^{-1}\mathbf{T}_F^C\boldsymbol{\chi}$
Figure 8 (left)	$1.20e^{-5}$	$3.57e^{-1}$	$1.19e^{-2}$	$4.87e^{-1}$
Figure 8 (right)	$3.14e^{-4}$	$6.24e^{-2}$	$5.14e^{-3}$	$2.67e^{-1}$

TABLE 3
Maximum error over time for each block propagator, when run sequentialy.

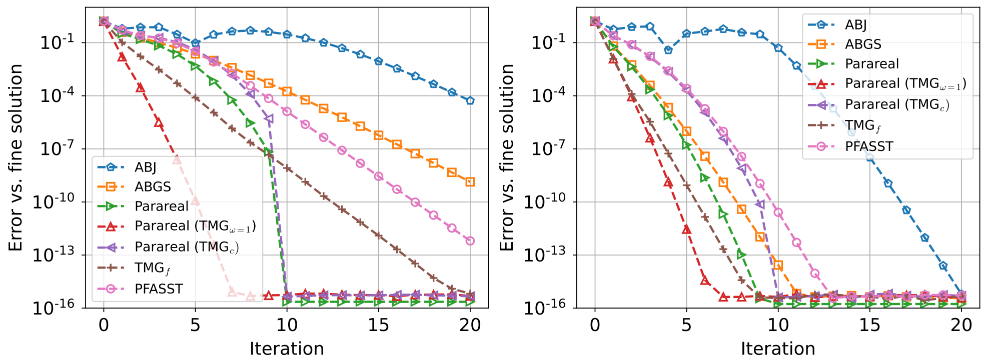


FIG. 8. *Comparison of iterative methods convergence using the GFM framework. Left: collocation as base fine integrator. Right: 4th order Runge-Kutta method as base fine integrator.* PARAREAL *(*TMG$_{\omega=1}$*) and* PARAREAL *(*TMG$_c$*) denote a specific coarsening for* PARAREAL *(see Section 5).*

used sequentially (data for Figure 8, left). Then we run all algorithms described in Table 2, initializing the block variable iterate with the same random initial guess. We show the error for the last block variable with respect to the fine sequential solution in Figure 8 (left). In addition, we show the same results, but using the classical 4th order Runge-Kutta method as fine propagator, 2nd order Runge-Kutta for the approximate integration operator and equidistant points using a volume formulation as described in Section 2.1.1 (Figure 8, right).

We observe that the TMG iteration converges fastest, due to the use of the most accurate block integrator for both levels (see Table 3). Keeping the same coarse grid correction but approximating the smoother (TMG$_f$) allows to improve the first iterations, but convergence for later iterations is closer to PFASST. This indicates that convergence for later iterations is mostly governed by the accuracy of the smoother (*i.e.* ABJ for both TMG$_f$ and PFASST). This is corroborated by the comparison of PFASST and TMG$_c$, which differ only by the type of smoother; while the exact Block Jacobi relaxation induces convergence of TMG$_c$ after $k = N$ iterations (a well known property of PARAREAL), using the ABJ smoother makes it impossible for PFASST to get this property.

On the other hand, the first iterations are also influenced by the coarse grid correction accuracy. We see that the iteration error is very similar for PFASST and TMG$_c$ which have the same coarse grid correction. This is more pronounced when using the 4th order Runge-Kutta method as base fine integrator, as we see in Figure 8 (right). Early iteration errors are similar for two-level methods that use the same coarse grid correction (TMG/ TMG$_f$, and PFASST/ TMG$_c$).

A similar behavior can be observed for the pair PARAREAL and ABGS. Looking at Table 2, we see that both algorithms use the same $\mathbf{B}_1^0$ operator. This suggests that the error for early iterations is mostly governed by the accuracy of the block operator $\mathbf{B}_1^0$, especially since this can also be observed for the elementary two-level methods (TMG and TMG$_f$ use the same $\mathbf{B}_1^0$ operator, as PFASST and TMG$_c$ do).

Remark 6.1. From a convergence point of view, PFASST and TMG$_c$ appear to be equivalent for the first iterations. But one should note that the block iteration of PFASST is cheaper than TMG$_c$, because an approximate block integrator is used for relaxation. Here we do not compare methods with respect to computational, as it is not in the scope of this paper. A model for cost, left for future work, could be combined with these convergence estimates to build a unified model for computational efficiency.

7. Conclusion. We have shown that the Generating Function Method (GFM) is an excellent mathematical tool to compare many iterative PinT algorithms, and it was this tool that led us to write all these algorithms in the same framework, so they can directly be compared. The GFM also showed that all these methods converge eventually super-linearly, which is due to the evolution nature of the problems. This is also observed in our numerical experiments, which were produced with a Python code that is publically available[13].

Our GFM framework also opens up many further research directions to explore, for example the study of multi-step block iterations like MGRIT with FCF-relaxation, and more complex two-level methods without Assumption 2.7. Also an extension of GFM to the multi-level versions of STMG, PFASST and MGRIT would be very valuable. Finally, all these methods are used in practice to solve true space-time problems, and the GFM should be capable to give convergence bounds for all methods in this situation as well, even for non-linear problems, as was shown for non-linear systems of ODEs [13].

Appendix A. Error bounds for *Primary Block Iterations*.

A.1. Incomplete Primary Block Iterations. First, we consider

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

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

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

where one block operator is zero (*e.g.* (PBI-1) corresponds to Parareal, (PBI-2) to Block Jacobi SDC and (PBI-3) to Block Gauss-Seidel SDC). To simplify notation, let

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

Then, application of Lemma 2.9 gives the recurrence relations

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

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

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

[13]https://github.com/Parallel-in-Time/gfm

for the corresponding generating functions. Setting $\delta := \max_{n\in\mathbb{N}} \left(e_{n+1}^0\right)$, we find that $\rho_0(\zeta) \leq \delta \sum_{n=0}^{\infty} \zeta^{n+1}$. By using the binomial series expansion

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

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

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

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

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

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

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

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. This yields for $k > 0$ the error bound

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

Following an idea by Gander [13], we can also consider the error recurrence $e_{n+1}^{k+1} \leq \alpha e_n^k + \bar{\beta} e_n^{k+1}$, $\bar{\beta} = \max(1, \beta)$. Using the upper bound $\sum_{n=0}^{\infty} \zeta^n = \frac{1}{1-\zeta} \leq \frac{1}{1-\beta\zeta}$, for the initial error, we avoid the series product and get $\rho_k(\zeta) \leq \delta\alpha^k \frac{\zeta^k}{(1-\bar{\beta})^{k+1}}$, as bound on the generating function. We then obtain the simpler error bound

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

as in the proof of [13, Th. 1].

Error bound for PBI-2. We again use (A.13) with $b_n = 1$ for the series product and

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

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

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

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

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

which yields for $k > 0$ the error bound

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

A.2. Full Primary Block Iteration. We now consider a primary block iteration with all block operators non-zero,

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

with α, β and γ defined in (A.4). Applying Lemma 2.9 leads to

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

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

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

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

Then using (A.13) with

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

we obtain

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

From this we can identify the error bound

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

REFERENCES

[1] 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, R. Kornhuber and et al., eds., vol. 40 of Lecture Notes in Computational Science and Engineering, Berlin, 2005, Springer, pp. 426–432.

[2] M. Bolten, D. Moser, and R. Speck, *A multigrid perspective on the parallel full approximation scheme in space and time*, Numerical Linear Algebra with Applications, 24 (2017), p. e2110.

[3] M. Bolten, D. Moser, and R. Speck, *Asymptotic convergence of the parallel full approximation scheme in space and time for linear problems*, Numerical linear algebra with applications, 25 (2018), p. e2208.

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

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

[6] V. Dobrev, T. Kolev, N. Petersson, and J. Schroder, *Two-level convergence theory for multigrid reduction in time (MGRIT)*, tech. report, LLNL-JRNL-692418, 2016.

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

[8] M. Emmett and M. Minion, *Toward an efficient parallel in time method for partial differential equations*, Comm. App. Math. and Comp. Sci., 7 (2012), pp. 105–132.

[9] C. Farhat and M. Chandesris, *Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid-structure applications*, International Journal for Numerical Methods in Engineering, 58 (2003), pp. 1397–1434.

[10] S. Friedhoff, R. Falgout, T. Kolev, S. MacLachlan, and J. Schroder, *A multigrid-in-time algorithm for solving evolution equations in parallel*, in Sixteenth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, United States, 2013.

[11] 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, 2015, pp. 69–114.

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

[13] M. J. Gander and E. Hairer, *Nonlinear convergence analysis for the Parareal algorithm*, in Domain Decomposition Methods in Science and Engineering, Lecture Notes in Computational Science and Engineering, O. B. Widlund and D. E. Keyes, eds., vol. 60, Springer, 2008, pp. 45–56.

[14] M. J. Gander, F. Kwok, and H. Zhang, *Multigrid interpretations of the Parareal algorithm leading to an overlapping variant and MGRIT*, Computing and Visualization in Science, 19 (2018), pp. 59–74.

[15] 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.

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

[17] M. J. Gander and M. Neumuller, *Analysis of a new space-time parallel multigrid algorithm for parabolic problems*, SIAM Journal on Scientific Computing, 38 (2016), pp. A2173–A2208.

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

[19] W. Gautschi, *Orthogonal polynomials: computation and approximation*, Oxford University Press, 2004.

[20] S. Götschel, M. Minion, D. Ruprecht, and R. Speck, *Twelve ways to fool the masses when giving parallel-in-time results*, in Springer Proceedings in Mathematics & Statistics, Springer International Publishing, 2021, pp. 81–94.

[21] S. Gunther, L. Ruthotto, J. B. Schroder, E. C. Cyr, and N. R. Gauger, *Layer-parallel training of deep residual neural networks*, SIAM Journal on Mathematics of Data Science, 2 (2020), pp. 1–23.

[22] W. Hackbusch, *Parabolic multi-grid methods*, in Proc. of the sixth int'l. symposium on Computing methods in applied sciences and engineering, VI, North-Holland Publishing Co., 1984, pp. 189–197.

[23] W. Hackbusch, *Multi-grid methods and applications*, vol. 4, Springer Science & Business Media, 2013.

[24] 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 Journal on Scientific Computing, 42 (2020), pp. A771–A796.

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

[26] D. E. Knuth, *The art of computer programming. 1. Fundamental algorithms*, Addison-Wesley, 1975.

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

[28] 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.

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

[30] Y. Maday and E. M. Rønquist, *Parallelization in time through tensor-product space–time solvers*, Comptes Rendus Mathematique, 346 (2008), pp. 113–118.

[31] 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.

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

[33] B. W. Ong and J. B. Schroder, *Applications of time parallelization*, Computing and Visualization in Science, 23 (2020).

[34] D. Ruprecht, *Wave propagation characteristics of parareal*, Computing and Visualization in Science, 19 (2018), pp. 1–17.

[35] D. Ruprecht and R. Speck, *Spectral deferred corrections with fast-wave slow-wave splitting*, SIAM Journal on Scientific Computing, 38 (2016), pp. A2535–A2557.

[36] M. Schreiber, P. S. Peixoto, T. Haut, and B. Wingate, *Beyond spatial scalability limitations with a massively parallel method for linear oscillatory problems*, The International Journal of High Performance Computing Applications, 32 (2018), pp. 913–933.

[37] B. S. Southworth, *Necessary conditions and tight two-level convergence bounds for Parareal and multigrid reduction in time*, SIAM Journal on Matrix Analysis and Applications, 40 (2019), pp. 564–608.

[38] 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, SC '12, Los Alamitos, CA, USA, 2012, IEEE Computer Society Press, pp. 92:1–92:11.

[39] G. A. Staff and E. M. Rønquist, *Stability of the Parareal algorithm*, in Domain Decomposition Methods in Science and Engineering, Lecture Notes in Computational Science and Engineering, R. Kornhuber and et al, eds., vol. 40, Springer, 2005, pp. 449–456.

[40] G. Wanner and E. Hairer, *Solving ordinary differential equations II*, Springer Berlin Heidelberg, 1996.