A multigrid perspective on the parallel full approximation scheme in space and time

Matthias Bolten1 | Dieter Moser2 | Robert Speck2

1 Department of Mathematics, Universität Kassel, Kassel, Germany
2 Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Jülich, Germany

Correspondence
Dieter Moser, Institut für Geometrie und Praktische Mathematik, RWTH Aachen, Templergraben 55, Aachen 52056, Germany. Email: moser@igpm.rwth-aachen.de

Summary
For the numerical solution of time-dependent partial differential equations, time-parallel methods have recently been shown to provide a promising way to extend prevailing strong-scaling limits of numerical codes. One of the most complex methods in this field is the “Parallel Full Approximation Scheme in Space and Time” (PFASST). PFASST already shows promising results for many use cases and benchmarks. However, a solid and reliable mathematical foundation is still missing. We show that, under certain assumptions, the PFASST algorithm can be conveniently and rigorously described as a multigrid-in-time method. Following this equivalence, first steps towards a comprehensive analysis of PFASST using blockwise local Fourier analysis are taken. The theoretical results are applied to examples of diffusive and advective type.

KEYWORDS
high-performance computing, local Fourier analysis, multigrid, parallel-in-time, PFASST

1 INTRODUCTION

Due to the rapid increase of the number of cores in today’s and future high-performance computing (HPC) systems, the demand for new parallelization strategies has grown rapidly in the last few decades. When the speedup of a parallelization of the spatial dimensions is saturated, one general idea is to utilize parallelization of the temporal dimension. In the work of Burrage,1 we find a classification of such methods, divided into parallelization across the step, across the method, or across the problem.

Direct time-parallel methods mostly belong to the class of parallelization across the method; examples are certain parallel Runge–Kutta methods.2,3 Only modest parallel speedup is expected for these methods, because the number of processing units used for the parallelization are bound by, for example, the number of Runge–Kutta stage values. Other direct methods for parallel-in-time integration include revisionist integral deferred correction (RIDC) methods,4 ParaExp,5 tensor-product space-time solvers,6 or methods using the Laplace transformation.7

If a method decomposes the problem into subproblems that are solvable in a parallel manner and couples these subproblems using an iterative method, it typically belongs to the class of parallelizations across the step. More parallel-in-time integration methods were found in the area of multiple-shooting methods.11,12 Among them, in 2001, Parareal by Lions et al.13 renewed the interest in parallel-in-time methods and sparked many new papers in its field. The success of Parareal is accounted to its simplicity and applicability: Only a fine but expensive and a coarse but cheap propagator in time has to be provided by the user. Then, parallelization across the temporal dimension can be achieved in an iterative prediction–correction manner. In principle, the number of
processing units is not bounded (as, e.g., for parallel Runge–Kutta methods), but depends on the actual decomposition of the time domain.

Parareal influenced other methods (see the work of Gander et al. \cite{Gander}) or even inspired the design of new methods. In the work of Minion, \cite{Minion} the Parareal approach is coupled to iterative solvers of a collocation problem, the so-called spectral deferred correction (SDC) methods. This approach is extended to the “parallel full approximation scheme in space and time” (PFASST) in the work of Emmett et al. \cite{Emmett} PFASST adopts and evolves the characteristics of Parareal by interweaving its iterations with those of the local SDC scheme. In addition, PFASST uses ideas from the theory of nonlinear full approximation storage (FAS) \cite{FAS} multigrid methods.

Multigrid methods in general have a long-standing successful history and a solid mathematical basis. \cite{Multigrid_History} Regarding parallel-in-time integration, the first attempt using multigrid ideas dates back to Hackbusch in 1984 \cite{Hackbusch}; since then, multigrid methods were further developed and resulted, for example, in the multigrid waveform relaxation, \cite{Waveform_Relaxation} in multigrid reduction-in-time, \cite{Reduction_In_Time} or in classical space-time multigrid. \cite{Space_Time_Multigrid} All these classes are not strictly separated from each other. Often, methods may be reformulated to fit into a new class. A prominent example is Parareal itself: it was reformulated as a multiple-shooting method as well as a multigrid method in the work of Gander et al. \cite{Multiple_Shooting}, which in turn paved the way for a comprehensive analysis of Parareal.

This already shows the growing number and diversity of parallel-in-time methods. Finding PFASST’s place in this diversity of methods contributes to the understanding of PFASST by opening up the opportunity to use different mathematical tools from different fields. In particular, multigrid theory offers a variety of tools, such as local Fourier analysis, to estimate the convergence and to obtain a priori error bounds. Such a mathematical analysis is critical for understanding and comparing these algorithms and for their optimization for different applications.

The goal of this work is to formulate PFASST as a multigrid method and apply the mathematical tools of multigrid analysis for two standard problems, namely, the diffusion and advection problem. This sheds light on a general strategy for estimating the convergence rate of PFASST and hereby the number of iterations needed to achieve a certain precision. We would like to stress that we do not aim for a comprehensive study of the influence of all relevant parameters and algorithmic options nor can we expect to answer all questions concerning the convergence properties of PFASST in this work. Rather, we will provide a mathematical framework describing PFASST in a novel and convenient way so that all (or, at least, many) of these questions become more accessible.

To achieve this goal, we proceed as follows: In Section 2.1, we introduce the notation and preliminaries necessary to state a matrix formulation of PFASST and its building blocks. In particular, we introduce the collocation problem and the notation to deal with the nested multilevel structure of our setting. On this basis, we briefly describe SDC and the extension to multiple levels in matrix form in Sections 2.2 and 2.3. Then, we review PFASST in algorithmic form in Section 2.4, which we then convert into a novel matrix form in Sections 3.1–3.3. This matrix form facilitates the use of ideas from multigrid analysis in Sections 4–4.3, which leads to a block decomposition of the iteration matrix of PFASST. In Section 5, we introduce four strategies to estimate the convergence rate of PFASST. The work is closed with an outlook and a conclusion in Section 6.

# 2 | PFASST

We start with a brief introduction of the building blocks of PFASST from the perspective of linear iterative solvers. To this end, we restrict ourselves to linear autonomous ordinary differential equations and—for the multilevel parts—to two levels only. We will comment on these restrictions in Section 6.

## 2.1 | Preliminaries and notation

The starting point is the linear autonomous ordinary differential equation in the Picard formulation

\[ U(t) = U_0 + \int_{t_0}^{t} A U(\tau) d\tau, \quad t \in [t_0, T], \]  

(1)

where \( A \) is a discretized spatial operator, for example, stemming from a method of line discretization of a partial differential equation. For the discretization in the temporal dimension, the time domain \([t_0, T]\) is divided
into \( L \) subintervals. Each subinterval \([t_{i-1}, t_i]\), with \( i \in \{1, \ldots, L\} \), contains a set of \( M \) nodes \( \{\tau_1, \ldots, \tau_M\} \). We choose
\[
0 = t_0 < \ldots < t_L = T, \quad t_i \leq \tau_1 < \ldots < \tau_M = t_{i+1},
\]
\[
\Delta t = t_{i+1} - t_i, \quad \Delta \tau_m = \tau_{m+1} - \tau_m.
\] (2)

Each set of nodes \( \{\tau_1, \ldots, \tau_M\} \) is used as quadrature nodes for the numerical integration with rules such as, for example, Gauß–Radau or Gauß–Lobatto. Note that, throughout this work, the last quadrature node coincides with the right border of the particular subinterval, which simplifies the formal notation of the algorithm. The results presented here translate to other quadrature rules with minor modifications, though. Furthermore, if a mathematical entity like a set of numerical values or a certain matrix is time dependent and belongs to a subinterval \([t_i, t_{i+1}]\), we denote it, for example, by \( U_{[t_i, t_{i+1}]} \) (if it is not clear from the context).

Due to the nested structure and the distinct treatment of spatial and temporal dimensions, an appropriate notation is needed. Continuous functions are always represented by lower case letters; discretized and semidiscretized functions are the upper case version. Let \( u(t, x) \) be a function in space and time, defined on the domain \([t_0, T] \times \mathbb{R}, \) with \( T \in \mathbb{R}_+ \). For \( N \) degrees of freedom in space \( x_1, \ldots, x_N \), we use the notation
\[
U(t) = (u(t, x_1), u(t, x_2), \ldots, u(t, x_N))^T \in \mathbb{R}^N, \quad t \in [t_0, T]
\]
for semidiscretization in space. A full space-time discretization is denoted by
\[
U_{[t_0, t_i]} = \left(U(\tau_1), U(\tau_2), \ldots, U(\tau_M)\right)^T \in \mathbb{R}^{M \times N}, \quad \tau_i \in [t_{i-1}, t_i], l \in \{1, \ldots, L\},
\]
\[
\begin{align*}
U &= \left(U_{[t_0, t_{i-1}]}, \ldots, U_{[t_{i-1}, t_i]}\right)^T \in \mathbb{R}^{M \times N \times L}.
\end{align*}
\]

On each subinterval, a collocation problem is posed. It arises, when quadrature is used as a numerical counterpart to the integration in Equation 1. The basis for most quadrature formulations is the interpolation, easily expressed using the Lagrange polynomial basis \( \{\ell_i\}_{i=1}^M \) with
\[
\ell_i(s) := \prod_{k=1, k \neq i}^{M} \frac{s - \tau_k}{\tau_i - \tau_k}.
\] (3)

If we weight each Lagrange polynomial with the evaluation of the function \( f(t) \) at the point \( \tau_i \) and sum them up, we get the interpolation polynomial of the function \( f(t) \), which is exact on the nodes \( \{\tau_1, \ldots, \tau_M\} \). Note that the formulation in Equation 3 is not the preferable way to evaluate the Lagrange polynomials, because it becomes numerically unstable for a large number of nodes. A remedy for this problem is barycentric interpolation (cf. the work of Berrut et al.\textsuperscript{26}).

Now, quadrature is nothing more than using the exact integration values of the interpolation polynomial as approximations for the integration of \( f(t) \). The following definition employs this strategy.

**Definition 1.** Let \( a \leq \tau_1 < \tau_2 < \ldots < \tau_M = b \) be the set of quadrature nodes on a subinterval \([a, b]\) with \( \Delta t = b - a \) and \( Q \) the quadrature matrix with entries
\[
q_{ij} = \frac{1}{\Delta t} \int_a^b \ell_i(r) \ell_j(r) dr, \quad i, j = 1, \ldots, M.
\]
We discretize Equation 1 at the quadrature nodes, using the matrix \( Q \) as approximation of the integral, and obtain this set of linear equations:
\[
U(\tau_i) = U(t_0) + \Delta t \sum_{j=1}^{M} q_{ij} A U(\tau_j), \quad i = 1, \ldots, M.
\]
Using the Kronecker product and the vector of ones \( \mathbf{1}_M \in \mathbb{R}^M \), we write this system of linear equations as
\[
U = U_0 + \Delta t (Q \otimes A) U, \quad \text{with } U_0 = \mathbf{1}_M \otimes U(t_0),
\]
or, more compactly,
\[
MU = (I - \Delta t Q \otimes A) U = U_0.
\] (4)

This problem is called the “collocation problem” on \([a, b]\).

The set of quadrature nodes determines the type of quadrature. Well-known quadrature rules are Gauß–Legendre, Gauß–Radau, and Gauß–Lobatto. These quadrature rules have a spectral order, which is reflected in the high order of the numerical solution of the collocation problem. Gauß–Radau and Gauß–Lobatto quadrature rules use quadrature nodes, which are in accordance with Equation 2. Due to the higher order, we will employ the Gauß–Radau quadrature in this work.
Finally, the PFASST algorithm is working on a hierarchy of discretizations. Given that it is common in multigrid theory to define the method on multiple levels via recursion and a two-level formulation, we focus on the two-level version with spatial coarsening only (i.e., PFASST is solving on a coarse and a fine level in space). For both levels, a separate set of operators and value vectors is needed. The coarse-level versions are simply denoted with a tilde (e.g., \( \tilde{A} \) is the coarse-level version of \( A \)).

### 2.2 SDC

Instead of directly solving the collocation problem on a subinterval, the SDC method utilizes a low-order method to generate an iterative solution that converges to the collocation solution \( U \). SDC was first introduced by Dutt et al.\(^\text{27} \) as an improvement of deferred correction methods.\(^\text{28} \) In the last decade, SDC was accelerated with generalized minimal residual method (GMRES) or other Krylov subspace methods,\(^\text{29} \) enhanced to a high-order splitting method,\(^\text{30–32} \) and found its way into the domain of time-parallel computing,\(^\text{33,34} \) in particular within PFASST.\(^\text{15,16} \)

Regarding the setting of this paper, we cast SDC as a stationary iterative method for the collocation problem as defined in Definition 1. This was pointed out earlier by various authors. For example, in the work of Weiser,\(^\text{35} \) this interpretation was used to optimize the convergence speed of SDC.

A general preconditioned stationary iterative method, denoted by

\[
U^{k+1} = U^k + P^{-1}(c - MU^k),
\]

is fully described by the preconditioner \( P \), the system matrix \( M \), and the right-hand side \( c \) of the linear equation under consideration. \( P \) has to be easy to invert, while being an accurate alternative for the system matrix \( M \). The SDC method follows this approach by replacing the full quadrature matrix \( Q \) by a lower triangular matrix \( Q_{\Delta} \). One simple way to generate a lower triangular matrix is to use the rectangle rule for quadrature instead of the Gauss–Radau rule. In the work of Weiser,\(^\text{35} \) an LU decomposition of \( Q \) provides a \( Q_{\Delta} \), which results in better convergence properties than the use of the simple rectangle rule while requiring the same computational effort.

The particular choice

\[
P_{SDC} = I - \Delta t Q_{\Delta} \otimes A \quad \text{and} \quad c = (U(t_0), U(t_0), \ldots, U(t_0))^T \in \mathbb{R}^{NM},
\]

then allows us to write SDC as an iterative method for the system matrix \( M \) as defined in Definition 1, where the right-hand side is given by the initial values \( U(t_0) \) of the ODE spread on each node. If SDC is used on another subinterval than the first, the right-hand side consists of a numerical approximation of \( U(t_1) \) spread on each node. In order to start the iteration, an initial iteration vector \( U^0 \) is needed. For SDC, the right-hand side is an apparent choice for the initial iteration vector. With these choices, one iteration is equivalent to one SDC sweep.\(^\text{35,36} \) The iteration matrix of SDC is simply given by

\[
T_{SDC} = I - P_{SDC}^{-1}M = I - (I - \Delta t Q_{\Delta} \otimes A)^{-1}(I - \Delta t Q \otimes A).
\]

Note that if we just use the lower triangular part of the \( Q \) matrix as \( Q_{\Delta} \), the iterative method mimics a Gauss–Seidel iteration. With \( Q_{\Delta} \) being a simpler integration rule or stemming from the LU decomposition of \( Q \) instead of the lower triangular part of \( Q \), we characterize SDC as an approximative Gauss–Seidel iteration.

### 2.3 Multilevel SDC

The next step towards PFASST is the introduction of multiple levels in space. This leads to the so-called “multi-level spectral deferred corrections” method (MLSDC), first introduced and studied by Speck et al.\(^\text{37} \) Here, SDC iterations (called “sweeps” in this context) are performed alternately on a fine and on a coarse level in order to shift work load to coarser (i.e., cheaper) levels. These cheaper levels are obtained, for example, by reducing the degrees of freedom in space or the order of the quadrature rule in time. Therefore, MLSDC requires suitable interpolation and restriction operators \( T^F_c \) and \( T^C_c \), and a coarse-grid correction in order to transfer information between the different levels. As a consequence, MLSDC can be written as an FAS-multigrid-like
iteration. Like SDC, it solves the collocation problem in an iterative manner, using the same initial iteration vector. For our purpose, we derive a two-level version of one MLSDC step \( U^k \rightarrow U^{k+1} \) from Speck et al.\(^{37}\) as follows:

1. Perform \( n_F \) fine SDC sweeps using the values \( U^k \) according to Equation 5. This yields provisional values \( U^* \).

2. Sweep from fine to coarse:
   
   (a) Restrict the fine values \( U^* \) to the coarse values \( \bar{U}^k \).
   
   (b) Compute the FAS correction \( r^k = \bar{M}U^k - T^c_FMU^* \).
   
   (c) Perform \( n_C \) coarse SDC sweeps on the collocation problem

   \[
   (I - \Delta r \bar{Q} \otimes \bar{A}) \bar{U} = \bar{U}_0 + r^k,
   \]

   with \( \bar{U}^k \) as starting value. This yields new values \( \bar{U}^{k+1} \).

3. Sweep from coarse to fine: Compute the interpolated coarse correction \( \delta^k = T^c_F \left( \bar{U}^{k+1} - T^c_F U^* \right) \) and add it to \( U^* \) to obtain \( U^{k+1} \).

Note that we use the FAS correction strategy here to match the description of Speck et al.\(^{37}\) This is just a question of notation, because, in the linear case, using this correction strategy is equivalent to the standard coarse-grid correction.\(^{18}\) Note further that we will only perform one fine and one coarse SDC sweep in each MLSDC iteration (i.e., \( n_F = n_C = 1 \)). The next lemma shows that, similarly for SDC, we can cast this algorithm as a preconditioned stationary iterative method.

**Lemma 1.** Let \( T^c_F \in \mathbb{R}^{NM \times NM} \) and \( T^c_F \in \mathbb{R}^{NM \times NM} \) be the prolongation and restriction operators that transfer information between the coarse and fine levels. We describe the same problem on a fine space-time grid with the system matrix \( M \) and on a coarse space-time grid with \( \bar{M} \). For both levels, we use an iterative method, which is characterized by \( \bar{P} \) and \( \bar{P} \) to solve \( MU = \bar{c} \) and \( \bar{M}U = T^c_F \bar{c} = \bar{c} \), respectively. Then, a combination of both methods using coarse-grid correction can be written as

\[
U^{k+1} = U^k + T^c_F \bar{P}^{-1}_{\text{SDC}} T^c_F \left( U^0 - MU^k \right)
\]

\[
U^{k+1} = U^k + T^c_F \bar{P}^{-1}_{\text{SDC}} T^c_F \left( U^0 - MU^k \right).
\]

It is possible to write Equation 8 in the form of Equation 5, using a new preconditioner \( P_{\text{MLSDC}} \), where

\[
P^{-1}_{\text{MLSDC}} = T^c_F \bar{P}^{-1}_{\text{SDC}} T^c_F + P^{-1}_{\text{SDC}} - P^{-1}_{\text{SDC}} M T^c_F \bar{P}^{-1}_{\text{SDC}} T^c_F.
\]

Following Equation 7 yields the MLSDC iteration matrix

\[
T_{\text{MLSDC}} = I - P^{-1}_{\text{MLSDC}} M.
\]

**Proof.** Let \( U^k \) be the result of the last iteration on the fine level. For the proof, we start in the middle of the algorithm. First, we compute the FAS correction

\[
r^k = \bar{M}T^c_F U^k - T^c_F MU^k
\]

and use it to modify \( \bar{P} \) for the next iteration on the coarse level. We start the iteration on the coarse level with

\[
\bar{U}^{k+1} = \bar{U}^k + \bar{P}^{-1}_{\text{SDC}} \left( \bar{c} + r^k - \bar{M}U^k \right) = T^c_F U^k + \bar{P}^{-1}_{\text{SDC}} T^c_F \left( \bar{c} - MU^k \right),
\]

with the restricted value \( \bar{U}^k = T^c_F U^k \). Then, we compute the coarse correction

\[
\delta^k = T^c_F \left( \bar{U}^{k+1} - T^c_F U^k \right)
\]

and obtain the half-step

\[
U^{k+\frac{1}{2}} = U^k + \delta^k = U^k + T^c_F \bar{P}^{-1}_{\text{SDC}} T^c_F \left( U^0 - MU^k \right)
\]

after some algebraic manipulations. Using this half-step for the next iteration on the fine level gives Equation 8. Simple algebraic manipulations, after inserting the half-step into the second step, yield the preconditioner (9), which immediately leads to the iteration matrix (10).\( \square \)

For the matrix formulation, it is irrelevant whether the MLSDC step starts with the computation on the fine or the coarse level. To comply with the literature, we leave the algorithm of MLSDC in the original order, while changing the order for the matrix formulation.
As a part of PFASST, MLSDC corresponds to the computation performed on each subinterval. Adding a communication framework between the MLSDC iterations performed on each subinterval leads to PFASST. However, adding the communication framework yields a structure similar to the one we have seen in Lemma 1.

2.4 The PFASST algorithm

The time-parallel PFASST algorithm in its final form was introduced by Emmett et al.\textsuperscript{16} as a combination of SDC methods\textsuperscript{27} with Parareal\textsuperscript{13} using an FAS correction strategy to allow for efficient spatial coarsening along the level hierarchy.

First, we explain PFASST on the basis of the schematic representation in Figure 1. This is followed by a simplified two-level algorithmic representation as it is found in the literature.\textsuperscript{16} First of all, we see the time domain, decomposed into subintervals, on the x-axis. On the y-axis, we see the elapsed computational time. Each processor is assigned to a subinterval, where it performs SDC sweeps with values \( U^k_{t_1,t_l} \) for the preceding processor. Note that depending on the collocation nodes, this value is computed from an extrapolation of all values \( U^k_{t_1,t_l} \) for the solver on the coarse and fine levels, respectively. These blocks are connected through FAS corrections to the subjacent MLSDC iterations and sends intermediate results on each level to the next processor. The blue and red blocks represent the communication framework between the processors. Before starting with the actual PFASST iterations, a prediction phase, represented by the first blue blocks near the x-axis, computes suitable initial values for the iterations to come.

On the basis of the schematic representation and the full algorithm description in the work of Emmett et al.\textsuperscript{16} we state a two-level version without the prediction phase. Let \( U^k_{t_1,t_l,m} \) be the value on the \( l \)th subinterval at the \( k \)th iteration and the \( m \)th node. We have

\[
F^k_{[t_1,t_l]} = [AU^k_{[t_1,t_l]1}, \ldots, AU^k_{[t_1,t_l]M_l}] \quad \text{and} \quad U^k_{[t_1,t_l]} = [U^k_{[t_1,t_l]1}, \ldots, U^k_{[t_1,t_l]M_l}],
\]

where \( M_l \) is the number of nodes on the \( l \)th interval. An upper bar, for example, \( U^k_{l-1} \), indicates that this value was sent by the preceding processor. Note that depending on the collocation nodes, this value is computed from an extrapolation of all \( U^k_{[t_1,t_l]1}, \ldots, U^k_{[t_1,t_l]M_l} \). If the last quadrature node coincides with the right interval border, only \( U^k_{[t_1,t_l]} \) is sent forward. These values are used as a new right-hand side to the collocation problem on the following subinterval. Denote the initial values for each subinterval by \( U^0_{[t_1,t_l]} \). Prepared with this notations, we are ready to formulate the PFASST algorithm:

1. Go down to the coarse level:

(a) Restrict the fine values \( U^k_{[t_1,t_l]} \) to the coarse values \( U^k_{[t_1,t_{l-1}]} \) and compute \( F^k_{[t_1,t_{l-1}]} \).
(b) Compute FAS correction \( \tau^k \), using \( F^k_{[t_1,t_l]} \) and \( F^k_{[t_1,t_{l-1}]} \).
(c) If \( l > 0 \), then receive the new initial value \( \tilde{U}^k_l \) from processor \( P_{l-1} \) and compute \( \tilde{F}^k_{[t_1,t_l]} \), else use the initial value of the ODE.
(d) Perform \( n_C \) SDC sweeps with values \( \tilde{U}^k_{[t_1,t_l]}, \tilde{F}^k_{[t_1,t_l]} \) and the FAS correction \( \tau^k \). This yields new values \( \tilde{U}^{k+\frac{1}{2}}_{[t_1,t_l]} \) and \( \tilde{F}^{k+\frac{1}{2}}_{[t_1,t_l]} \).
(e) Send \( \tilde{U}^{k+\frac{1}{2}}_{[t_1,t_l]} \) to processor \( P_{l+1} \) if \( l < N - 1 \). This will be received as the new initial condition \( \tilde{U}^k_{t_1} \) for the solver on the coarse level.

![Schematic representation of the PFASST algorithm with two levels and four processes](image-url)
2. Return to the fine level:
   (a) Interpolate the coarse correction $\delta^k \approx \tilde{U}^{k+\frac{1}{2}}_{[t_{l-1}, t_l]} - \tilde{U}^k_{[t_{l-1}, t_l]}$ and add to $U^k_{[t_{l-1}, t_l]}$, yielding $U^{k+\frac{1}{2}}_{[t_{l-1}, t_l]}$. Recompute $F^{k+\frac{1}{2}}_{[t_{l-1}, t_l]}$.
   (b) If $l > 0$, then receive the new initial value $\tilde{U}^k_{[t_{l-1}, t_l]}$ from processor $P_{l-1}$, else take the initial value of the ODE.
   (c) Interpolate coarse correction vector $\delta^k = \tilde{U}^{k+\frac{1}{2}}_{t_l} - \tilde{U}^k_{t_l}$ and add it to $U^k_{t_l}$, yielding $U^{k+\frac{1}{2}}_{t_l}$. Recompute $F^{k+\frac{1}{2}}_{[t_{l-1}, t_l]}$.
   3. Perform $n_F$ fine SDC sweeps using the values $U^{k+\frac{1}{2}}_{[t_{l-1}, t_l]}$ and $F^{k+\frac{1}{2}}_{[t_{l-1}, t_l]}$. This yields values $U^{k+1}_{[t_{l-1}, t_l]}$ and $F^{k+1}_{[t_{l-1}, t_l]}$.
   4. Send $U^{k+1}_{[t_{l-1}, t_l]}M_l$ to processor $P_{l+1}$ if $l < N - 1$. This will be used as initial value $\tilde{U}^{k+1}_{t_{l+1}}$ in the next iteration on the fine level.

This form of the PFASST algorithm is suitable for implementation, but not for the mathematical analysis. It is especially difficult to capture how the parts influence each other. To overcome this limitation, we now change the perspective: Instead of building the algorithm in a “vertical” way (i.e., MLSDC on each subinterval), we look at all intervals at once in a “horizontal” way (i.e., we analyze how the different components of PFASST act on the full time-domain $[t_0, T]$).

### 3 | A MULTIGRID PERSPECTIVE

In this section, the perspective is shifted from solvers on one specific subinterval to the interaction of the solvers on the whole time domain $[t_0, T]$. We begin with stating the composite collocation problem.

**Definition 2.** Let the interval $[t_0, T]$ be decomposed as in Equation 2 into $L$ subintervals $[t_l, t_{l+1}]$. On each subinterval, a collocation problem in the form of Equation 4, denoted by $M_{[t_l, t_{l+1}]}$, is posed. The collocation matrix on the whole time domain is then defined as

$$
M_{[t_0, T]} = \begin{pmatrix}
M_{[t_0, t_1]} \\
-\mathbf{H} & M_{[t_1, t_2]} \\
& \ddots & \ddots \\
& & -\mathbf{H} & M_{[t_{L-1}, T]} \\
\end{pmatrix} \in \mathbb{R}^{NM}, \text{ with}
$$

$$
\mathbf{N} = \begin{pmatrix}
0 & 0 & \cdots & 1 \\
0 & 0 & \cdots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1 \\
\end{pmatrix} \in \mathbb{R}^M, \text{ and thus } \mathbf{H} = \mathbf{N} \otimes \mathbf{I}_N \in \mathbb{R}^{NM}.
$$

The operator $\mathbf{N}$ handles how the new starting value for the upcoming interval is produced. Furthermore, stacking together

$$
e^{(t_{l+1})} = \begin{cases}
U_0, & \text{for } l = 0 \\
0, & \text{for } l > 0
\end{cases} \in \mathbb{R}^{NM},
$$

forms the right-hand side $e^{(t_{l+1})}$ for the “composite collocation problem”

$$
M_{[t_0, T]} \begin{pmatrix}
U_{[t_0, t_1]} \\
U_{[t_1, t_2]} \\
\vdots \\
U_{[t_{L-1}, T]}
\end{pmatrix} = \begin{pmatrix}
U_0 \\
0 \\
\vdots 
\end{pmatrix} = e^{(t_{L+1})}.
$$

Along with this definition, the block structure of our problem becomes evident. On the diagonal of the new collocation matrix, we find blocks of size $NM$, each of them being associated with the subintervals $[t_l, t_{l+1}]$. The operators on the subdiagonal deal with the communication between two adjacent subintervals. When designing iterative solvers for the composite collocation problem, we also want to exploit this block structure. Therefore, the next two sections are dedicated to the block versions of an approximate Jacobi and an approximate Gauss–Seidel iteration, and both will emerge from the interpretation of SDC as an approximate Gauss–Seidel iterative solver. When the two methods are correctly interlaced, this results in PFASST as we will see in Section 3.3.

### 3.1 | The approximative block Gauss–Seidel solver

The classical Gauss–Seidel solver is a splitting method, which incorporates the lower triangular part of the system matrix as preconditioning matrix. Obviously, this strategy is possible in principle for the composite collocation problem, as defined in
Definition 2, but this would neglect the particular block structure of the problem. Therefore, we now construct a block version of the SDC iteration, following its description as an approximative Gauß–Seidel solver.

Assume that we perform one SDC sweep on each subinterval via

$$U^{k+1}_{[t_{l-1}, t_l]} = U^k_{[t_{l-1}, t_l]} + P^{-1}_{[t_{l-1}, t_l]} \left( c^{k+1}_{[t_{l-1}, t_l]} - M_{[t_{l-1}, t_l]} U^k_{[t_{l-1}, t_l]} \right),$$  \hspace{1cm} (13)

where $P_{[t_{l-1}, t_l]}$ denotes the SDC preconditioner (6) and $c^k_{[t_{l-1}, t_l]}$ is the right-hand side on the $l$th subinterval in the $k$th iteration. In order to pass the last value forward in time to the next subinterval, we can use the matrix $N$. Therefore, the right-hand side of the collocation problem can be written as

$$\begin{align*}
    c^k_{[t_{l-1}, t_l]} &= \begin{bmatrix} U_0, & \ldots, & U_0 \end{bmatrix}, \quad \text{for } l = 0 \text{ and } k > 0, \\
    c^k_{[t_{l-1}, t_l]} &= \begin{bmatrix} \hat{U}_l^k, & \ldots, & \hat{U}_l^k \end{bmatrix} = HU^k_{[t_{l-1}, t_l]} \quad \text{for } l > 0 \text{ and } k > 1.
\end{align*}$$  \hspace{1cm} (14)

For some initial iteration vector $U^0_{[t_{l-1}, t_l]}$, stemming, for example, from copying the initial value on each node of each subinterval (“spreading”), we can write this process compactly as a single approximate Gauß–Seidel step over the whole time domain.

**Lemma 2.** Let $M_{[t_0, T]}$ be the matrix of a composite collocation problem (see Definition 2). Using SDC on each subinterval and passing the results via Equation 14 corresponds to

$$U^{k+1}_{[t_0, T]} = U^k_{[t_0, T]} + P^{-1}_{[t_0, T]} \left( c^k_{[t_0, T]} - M_{[t_0, T]} U^k_{[t_0, T]} \right),$$  \hspace{1cm} (15)

with

$$U^k_{[t_0, T]} = \begin{bmatrix} U^k_{[t_0, t_1]} \\ U^k_{[t_1, t_2]} \\ \vdots \\ U^k_{[t_{l-1}, T]} \end{bmatrix} \in \mathbb{R}^{NML}, \quad c^k_{[t_0, T]} = \begin{bmatrix} U_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{NML},$$

and

$$P_{[t_0, T]} = \begin{bmatrix} P_{[t_0, t_1]} & -H & 0 & \cdots & 0 \\ -H & P_{[t_1, t_2]} & -H & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ -H & 0 & \cdots & -H & P_{[t_{l-1}, T]} \end{bmatrix} \in \mathbb{R}^{NML \times NML}.$$

**Proof.** We multiply Equation 13 with $P_{[t_{l-1}, t_l]}$ from the left and Equation 15 with $P_{[t_0, T]}$ from the left. Comparing the resulting terms line by line reveals the equivalence. \hfill \Box

This Gauß–Seidel-like iteration can be found in Figure 1: Here, after each blue block that represents SDC sweeps on the coarse level, the values $U^k_l$ are passed forward in time, providing new initial values for the sweep on the next interval. Thus, the iteration on the coarse level can be identified with an approximate block Gauß–Seidel iteration for the composite collocation problem.

### 3.2 The approximative block Jacobi solver

The communication, emerging from the use of the approximate block Gauß–Seidel solver, is blocking. Each processor has to wait for its predecessor. Hence, this is a purely serial approach. A simple way to avoid the blocking communication is to use an approximate block Jacobi solver, omitting the subdiagonal blocks responsible for the communication.

Assume that we perform a step similar to Equation 13, but we use the right-hand side

$$\begin{align*}
    c^k_{[t_{l-1}, t_l]} &= \begin{bmatrix} U_0, & \ldots, & U_0 \end{bmatrix}, \quad \text{for } l = 0 \text{ and } k > 0, \\
    c^k_{[t_{l-1}, t_l]} &= \begin{bmatrix} \hat{U}_l^{k-1}, & \ldots, & \hat{U}_l^{k-1} \end{bmatrix} = HU^{k-1}_{[t_{l-1}, t_l]} \quad \text{for } l > 0 \text{ and } k > 1
\end{align*}$$  \hspace{1cm} (16)

instead. This means that not the result of the current but of the previous iteration of the preceding interval is used.

In the first iteration, the result of the prediction phase is used. Using the simple spreading prediction phase, this is easily achieved by choosing $U^0_l = U_0$ for $l > 0$. 
Lemma 3. Let $M_{[t_0,T]}$ be the matrix of a composite collocation problem (see Definition 2). Then, using SDC on each subinterval and passing the results via Equation 16 corresponds to

$$U_{[t_0,T]}^{k+1} = U_{[t_0,T]}^k + \hat{P}_{[t_0,T]}^{-1} \left( c_{[t_0,T]} - M_{[t_0,T]} U_{[t_0,T]}^k \right)$$

with

$$\hat{P}_{[t_0,T]} = \begin{pmatrix} \hat{P}_{[t_0,t_1]} & \hat{P}_{[t_1,t_2]} & \cdots & \hat{P}_{[t_{L-1},T]} \end{pmatrix},$$

and $U_{[t_0,T]}^k$, $c_{[t_0,T]}$ defined as in Lemma 2.

Proof. Similar to the proof in Lemma 2, a blockwise comparison yields the equivalence. In particular, the influence of the subdiagonal of $M_{[t_0,T]}$ on the communication is revealed by a blockwise view on Equation 17 as follows:

$$U_{[t_0,t_1]}^{k+1} = U_{[t_0,t_1]}^k + \hat{P}_{[t_0,t_1]}^{-1} \left( U_0 - M_{[t_0,t_1]} U_{[t_0,t_1]}^k \right), \quad \text{for } l = 0$$

$$U_{[t_l,t_{l+1}]}^{k+1} = U_{[t_l,t_{l+1}]}^k + \hat{P}_{[t_l,t_{l+1}]}^{-1} \left( \hat{H} U_{[t_l,t_{l+1}]}^k - M_{[t_l,t_{l+1}]} U_{[t_l,t_{l+1}]}^k \right), \quad \text{for } l > 1.$$

The values $NU_{[t_{L-1},T]}^k$ are equivalent to $1_M \otimes \bar{U}^{k-1}$. □

It is evident that due to the block-diagonal structure of $\hat{P}_{[t_0,T]}$, one block Jacobi iteration may be performed concurrently on $L$ computing units. This approach corresponds to the sweeps on the fine (red) blocks in Figure 1: These sweeps can be performed in parallel, because they do not depend on the previous subinterval at the same iteration. Therefore, the iteration on the fine level can be identified with an approximate block Jacobi iteration for the composite collocation problem (12).

3.3 Assembling PFASST

In Section 2.3, multigrid elements were already introduced to SDC to form MLSDC. The same ideas apply when we now interlace both iterative block solvers aforementioned to introduce a novel representation of PFASST. In order to achieve more parallelism, we compute the approximate Gauss–Seidel iteration step on the coarse level and the approximate block Jacobi iteration step on the fine level, so that the more cost-intensive work is done in parallel. As the following theorem shows, it is now possible to write PFASST in the form of Equation 8, and we are able to state an iteration matrix, which has the form of a two-level multigrid iteration matrix.

Theorem 1. Let $T_F^C$ and $T_C^F$ be blockwise defined transfer operators, which treat the subintervals independently from each other, and let

$$\left\{ P_{[t_0,t_1]}, \ldots, P_{[t_{L-1},T]} \right\} \quad \text{and} \quad \left\{ \hat{P}_{[t_0,t_1]}, \ldots, \hat{P}_{[t_{L-1},T]} \right\}$$

be sets of preconditioner for the fine and coarse levels, respectively, describing SDC sweeps on $[t_l,t_{l+1}]$ for $l \in \{0, \ldots, L-1\}$ and $t_L = T$. Let $M_{[t_0,T]}$ be the composite collocation matrix of Definition 2 and $H, \hat{H}$ be the operations to compute the initial value for the following subinterval. Then, the linear two-level version of PFASST can be written in matrix form as

$$U_{[t_0,T]}^{k+1,1} = U_{[t_0,T]}^k + T_F^C \hat{P}_{[t_0,T]}^{-1} T_C^F \left( c_{[t_0,T]} - M_{[t_0,T]} U_{[t_0,T]}^k \right),$$

$$U_{[t_0,T]}^{k+1,2} = U_{[t_0,T]}^k + \hat{P}_{[t_0,T]}^{-1} \left( c_{[t_0,T]} - M_{[t_0,T]} U_{[t_0,T]}^{k+1} \right),$$

(18)
with $\hat{\Pi}_{[0,T]}$, as in Lemma 2, and $\tilde{\Pi}_{[0,T]}$, as in Lemma 3. In addition, define $H$ and $\tilde{H}$ such that $\tilde{H}^{\text{C}} = T^{\text{C}}_F H$ and $e_{[0,T]} = [U^0, 0, \ldots, 0]$ as well as $\tilde{e}_{[0,T]} = T^{\text{C}}_F \tilde{e}_{[0,T]}$. Finally, the PFASST iteration matrix is given by

$$T_{\text{PFASST}} = \left( 1 - \hat{\Pi}_{[0,T]}^{-1} M_{[0,T]} \right) \left( 1 - T^{\text{C}}_F \hat{\Pi}_{[0,T]}^{-1} T^{\text{C}}_F M_{[0,T]} \right). \tag{19}$$

**Proof.** We compare systematically each step of PFASST with the computations on the subinterval found in Equation 18, which expands into

$$r^k_{[0,T]} = \tilde{M}_{[0,T]} T^{\text{C}}_F U^k_{[0,T]} - T^{\text{C}}_F M_{[0,T]} U^k_{[0,T]} \tag{20}$$

$$\tilde{U}^{k+1}_{[0,T]} = \tilde{U}^k_{[0,T]} + \hat{\Pi}^{-1} \left( \tilde{e}_{[0,T]} + \tau^k_{[0,T]} - \tilde{M}_{[0,T]} \tilde{U}^k_{[0,T]} \right) \tag{21}$$

$$U^{k+\frac{1}{2}}_{[0,T]} = U^k_{[0,T]} + T^{\text{C}} F \left( \tilde{U}^{k+1}_{[0,T]} - T^{\text{C}} F U^k_{[0,T]} \right) \tag{22}$$

$$U^{k+1}_{[0,T]} = U^{k+\frac{1}{2}}_{[0,T]} + \hat{\Pi}^{-1} \left( \tilde{e}_{[0,T]} - M_{[0,T]} U^{k+\frac{1}{2}}_{[0,T]} \right) \tag{23}$$

From top to bottom, we have the computation of the FAS $\tau$-correction of the right-hand side, the SDC sweep on the coarse level, the FAS $\delta$-correction of the updated value, and the SDC sweep on the fine level. PFASST’s communication between the subintervals has been already derived in Lemmas 2 and 3. The evaluations of right-hand side, in the form of $F$ and $\tilde{F}$, are included in the matrix vector multiplications with $M_{[0,T]}$ and $\tilde{M}_{[0,T]}$, respectively.

The computation of the FAS correction $r^k_{[0,T]}$ as in Equation 20 differs from the formula (11), which we derived for MLSDC, as the latter is only applied on one subinterval. This leads to the FAS correction vector of Equation 20 having additional terms

$$L = \tilde{H} T^{\text{C}}_F - T^{\text{C}}_F H \tag{24}$$

with

$$\tau_{[0,T]} = \left( \tau_{[0,1]}, \tau_{[1,2]}, \ldots, \tau_{[N-1,1]} \right) + L U^k_{[0,1]}, \ldots, \tau_{[N-1,1]} + LU^k_{[N-1,1]} \right)^T. \tag{25}$$

However, by requirement, we have $L = 0$, and in Remark 1, we will investigate how this requirement is met. The iteration matrix is the result of simple algebraic manipulations.

In contrast to Lemma 1 for MLSDC, we now have an additional requirement, which necessitates the following remark.

**Remark 1.** When only coarsening in space is performed, our interpolation operator has the form $T^{\text{C}}_F = I_M \otimes \tilde{T}^{\text{C}}_F$, where $\tilde{T}^{\text{C}}_F$ is the restriction operator in space. In this case, it yields

$$L = \tilde{H} \cdot (I_M \otimes \tilde{T}^{\text{C}}_F) - (I_M \otimes \tilde{T}^{\text{C}}_F) \cdot H = N \otimes (\tilde{T}^{\text{C}}_F - T^{\text{C}}_F) = 0.$$

and therefore, the condition is met. When we additionally have coarsening in the number of quadrature nodes, our restriction operator takes the form $T^{\text{C}}_F = \tilde{T}^{\text{C}}_F \otimes \tilde{T}^{\text{C}}_F$, where $\tilde{T}^{\text{C}}_F$ is the restriction operator in time. In this case, it yields

$$L = (\tilde{N} \tilde{T}^{\text{C}}_F - \tilde{T}^{\text{C}}_F N) \otimes \tilde{T}^{\text{C}}_F,$$

and we need further assumptions about the restriction operator in time. Let $t_{ij}$ be the $j$th entry of the $i$th row of $\tilde{T}^{\text{C}}_F$. Due to the aforementioned assumptions, $L = 0$ translates to

$$0 = \tilde{T}^{\text{C}}_F N \cdot \tilde{T}^{\text{C}}_F,$$

and

$$0 = \begin{pmatrix} 0 & \cdots & 0 & \sum_{j=1}^{M} t_{i,j} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \sum_{j=1}^{M} t_{i,j} \end{pmatrix} - \begin{pmatrix} t_{1,1} & \cdots & t_{1,M-1} & t_{1,M} \\ \vdots & \ddots & \vdots \\ t_{M,1} & \cdots & t_{M,M-1} & t_{M,M} \end{pmatrix}.$$
Hence, we require that
\[ t_{M,j} = 0 \quad \forall \ j \in \{1, \ldots, M-1\} \quad \text{and} \quad t_{M,M} = \sum_{j=1}^{M} t_{ij} \quad \forall \ i \in \{1, \ldots, \tilde{M}\}. \]

If the restriction \( \tilde{T}_F \) of a constant vector yields a constant vector with the same values but a smaller dimension, we infer that
\[ \sum_{j=1}^{M} t_{ij} = 1 \quad \forall \ i \in \{1, \tilde{M}\} \]
and hence \( t_{M,M} = 1 \). This requirement is met when the restriction in time projects the last node of the fine level onto the last node of the coarse level. It holds, for example, for the linear restriction or simple injection, as long as \( \tilde{t}_M = t_M \).

The hierarchy of discretizations on which PFASST is working and the exchange of information between those levels using FAS and coarse-grid correction obviously indicates a strong similarity to classical multigrid methods. This relation is in particular emphasized by the iteration matrix, also known as the error propagator. Standard multigrid methods are typically described and analyzed by their iteration matrix \( T_{MG} \), which reads
\[ T_{MG}(\nu, \eta) = (I - P^{-1}_{\text{post}}) \nu (I - T_C^{\tilde{M}})^{-1} T_F \tilde{C} M (I - P^{-1}_{\text{pre}}) \eta, \]
for \( \nu \) post- and \( \eta \) pre-smoothing steps. The expression in the middle is the coarse-grid correction, and \( \tilde{M} \) is a coarse version of \( M \). In a standard two-grid algorithm, the problem is solved directly at the coarsest level. In practice, it is also legitimate to use the approximate solution in the form of \( \tilde{P}^{-1} \). PFASST does exactly this. Under the conditions of Theorem 1, the comparison of Equation 26 and Equation 19 yields that PFASST can be readily interpreted as a multigrid algorithm with one post-smoothing iteration and no pre-smoothing steps. We point out that this does not prove that PFASST actually behaves like a classical multigrid method on elliptic PDEs in terms of convergence and robustness. In particular, properties, such as smoothing and approximation, are not necessarily satisfied, and the analysis of the algorithm in this respect is left for future work.

However, this does not prohibit an analysis based on the tools that are usually used for multigrid schemes.

## 4 LOCAL FOURIER ANALYSIS FOR PFASST

The most common tool for analysis and design of multigrid algorithms is the local Fourier analysis (LFA), initially introduced by Brandt\(^{17} \) (cf. the work of Trottenberg et al.\(^{18} \)). It simplifies the problem by making assumptions such as periodic domains and constant coefficients. The goal of LFA is, in the rigorous case, the computation and usually the estimation of the spectral radius of the iteration matrix and its building blocks. If there is no Toeplitz structure to exploit in the building blocks of the iteration matrix or if the boundary conditions have a strong influence, LFA fails to give accurate predictions. To account for this problem, it is reasonable to use a combination of LFA and algebraic computations, as it was first introduced by Friedhoff et al.\(^{39} \) in the form of “semi-algebraic mode analysis” (SAMA). The motivation behind SAMA is the large gap between the theoretical analysis and the actual performance of multigrid methods for parabolic equations and time-parallel methods. Friedhoff et al. demonstrated that SAMA enables accurate predictions of the short-term behavior and asymptotic convergence factors.

In this work, we focus on two prototype problems, namely, the diffusion and advection in one dimension, to show how, in principle, PFASST can be analyzed using the framework we describe in this work. We will use periodicity in space to maintain agreement with the assumptions of LFA.

The usual approach to LFA is to define and work with Fourier symbols for each operator. These Fourier symbols represent the behavior of the operators on the grid functions
\[ \varphi_\theta(x) = \exp(i\theta x/h), \quad x \in [0, 1], \quad \theta \in [0, 2\pi), \]
for distinct frequencies \( \theta \). The observation, how the different grid functions are damped or changed on different grids and under different operations, is a central point of LFA.

However, in our analysis, we will make use of the matrix notation and henceforth avoid the use of explicit Fourier symbols, but rather perform a block diagonalization of the matrices of PFASST. The goal is the blockwise diagonalization of the iteration matrix of PFASST. Later on, each block will be associated with a discrete frequency. Therefore, we will be able to state which frequency is damped or changed to which extend.
Due to the periodicity in space, parts of the iteration matrix consist of circulant matrices. A circulant matrix is a special kind of Toeplitz matrix where each row vector is rotated one element to the right relative to the preceding row vector. More precisely, we denote

$$C = \begin{pmatrix} c_0 & c_1 & \cdots & c_{N-1} \\ c_{N-1} & c_0 & c_1 & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ c_1 & \cdots & c_{N-1} & c_0 \end{pmatrix}.$$  \hspace{1cm} (28)

It has the eigenvalues $\lambda_k$ and eigenvectors $\psi_k$ for $k = 0, \ldots, N-1$ with

$$\lambda_k = \sum_{j=0}^{N-1} c_j \exp \left( \frac{i 2\pi k j}{N} \right) \quad \text{and} \quad \psi_k = \frac{1}{\sqrt{N}} \left[ \exp \left( \frac{i 2\pi k 0}{N} \right), \exp \left( \frac{i 2\pi k 1}{N} \right), \ldots, \exp \left( \frac{i 2\pi k (N-1)}{N} \right) \right]^T.$$  \hspace{1cm} (29)

This also means that with the transformation matrix $\Psi$, which is orthogonal and consists of the eigenvectors, it holds

$$\left( \Psi^T C \Psi \right)_{jk} = \lambda_j.$$  \hspace{1cm} (30)

For two diagonalizable matrices $A, B$ with the same eigenvector space, it holds

$$\Psi^T (A + B) \Psi = \Psi^T A \Psi + \Psi^T B \Psi = D^{(A)} + D^{(B)},$$

$$\Psi^T A B \Psi = \Psi^T A \Psi \Psi^T B \Psi = D^{(A)} D^{(B)},$$

$$\Psi^T A^{-1} \Psi = (D^{(A)})^{-1}.$$ \hspace{1cm} (31)

Here, $D^{(A)}$ denotes the diagonal matrix containing the eigenvalues of $A$. Furthermore, for the Kronecker product, we have $P^{-1} A \otimes B P = B \otimes A$, where $P$ is a suitable permutation matrix. These rules will be used extensively by the following algebraic manipulations.

### 4.1 Transforming the PFASST iteration matrix

The PFASST algorithm has three layers it works on. The first layer is the spatial space, the second consists of the quadrature nodes, and the third is the temporal structure given by the subintervals. All layers are interweaved: We illustrate this by rewriting the system matrix $M_{[t_0, t_1]}$ under the assumption that we have the same problem (i.e., the same discretization of the same operator) on each subinterval

$$M_{[t_0, t_1]} = I_L \otimes I_M \otimes I_N - \Delta t \cdot I_L \otimes Q \otimes A - E \otimes N \otimes I_N,$$  \hspace{1cm} (32)

where $N$ is again the number of degrees of freedom in the spatial dimension, $M$ the number of nodes per subinterval, and $L$ the number of subintervals. Also, a new operator $E \in \mathbb{R}^{L \times L}$ is introduced, which has ones on the first subdiagonal and zeros elsewhere. In each term, the layers are separated by the Kronecker product, and through the summation of those parts, we interweave them again. Our transformation aims at the layer where each matrix is diagonalizable by $\Psi$. With the help of the permutation matrix $P$, we define a transformation matrix $F$, which affects and reorders the layers, as

$$F = P \cdot (I_L \otimes I_M \otimes \Psi), \quad F^{-1} = (I_L \otimes I_M \otimes \Psi^T) \cdot P^{-1},$$

and therefore

$$F^{-1} M_{[t_0, t_1]} F = I_N \otimes (I_L \otimes I_M - E \otimes N) - \Delta t \cdot D^{(A)} \otimes I_L \otimes Q.$$

This yields diagonal matrices on the layer for the spatial dimension, so that we can write

$$F^{-1} M_{[t_0, t_1]} F = \text{diag} \left( B_1^{(M_{[t_0, t_1]}^M)}, \ldots, B_N^{(M_{[t_0, t_1]}^M)} \right)$$

with $B_j^{(M_{[t_0, t_1]}^M)} = I_L \otimes I_M - E \otimes N - \Delta t \cdot \lambda_j I_L \otimes Q$.

We call the resulting blocks “time collocation blocks,” highlighting the dimension and components of the blocks. We note that this is actually SAMA in a different notation.

The aforementioned transformation strategy leads to a block structure for all matrices that emerge in the formulation of PFASST, particularly for the iteration matrix. Here, the interpolation and restriction matrices need special attention, though.
4.1.1 Transforming interpolation and restriction

In this section, we focus on interpolation and restriction operators, which are designed for two special isometric periodic grids with an even number of fine grid points. Between these two grids, we define a special class of interpolation and restriction pairs.

**Definition 3.** Let \( \mathbf{C} \in \mathbb{R}^{N/2 \times N/2} \) be a circulant matrix, with the associated eigenvalues \( \{\lambda_k\}_{k=1}^{N/2} \), and let the fine grid \( X \) and coarse grid \( \bar{X} \) be defined as

\[
X = [x_1, \ldots, x_N] \quad \text{and} \quad \bar{X} = [\bar{x}_1, \ldots, \bar{x}_{N/2}], \text{with } x_{2j-1} = \bar{x}_j \text{ for all } j \in \{1, \ldots, \frac{N}{2}\}.
\]

Let \( \mathcal{W}(\cdot, \cdot) : \mathbb{R}^{N/2 \times N/2} \times \mathbb{R}^{N/2 \times N/2} \mapsto \mathbb{R}^{N \times N/2} \) be an “interweaving” operator, which stacks together the rows of two matrices subsequently, beginning with the first row of the first matrix, followed by the first row of the second matrix, and finally ending with the last row of the second matrix. Then, we define the class of circulant interweaved interpolation (“CI-interpolation”) operators as

\[
\Pi = \left\{ \mathbf{T}_F^C : \exists \mathbf{C} \in \mathbb{R}^{N/2 \times N/2} \quad \text{circulant and} \quad \mathbf{C} \cdot \mathbf{1} = \mathbf{1}, \mathbf{T}_F^C = \mathcal{W}(\mathbf{I}_{N/2}, \mathbf{C}) \right\}
\]

and the class of circulant interweaved restriction (“CI-restriction”) operators as

\[
\Pi^T = \left\{ \mathbf{T}_F^C : \mathbf{c}(\mathbf{T}_F^C)^T \in \Pi, \mathbf{c} \in \mathbb{R} \right\}.
\]

Due to the circulant nature of the interweaved matrices, we are able to state a transformation analytically.

**Lemma 4.** Let \( \mathbf{T}_F^C \) be a CI-interpolation and \( \mathbf{T}_C^F \) the associated CI-restriction operator, \( \mathbf{Ψ} \) the transformation matrix for \( N \) grid points, and \( \mathbf{Ψ}_C \) the transformation matrix for \( N/2 \) grid points. Then, it holds

\[
\mathbf{Ψ}^T \mathbf{T}_C^F \mathbf{Ψ}_C = \begin{pmatrix}
\hat{d}_0 & \ldots & \hat{d}_{N/2-1} \\
\vdots & \ddots & \vdots \\
\hat{d}_0 & \ldots & \hat{d}_{N/2-1}
\end{pmatrix}
\]

and

\[
\mathbf{Ψ}^T \mathbf{T}_C^F \mathbf{Ψ}_C = \begin{pmatrix}
d_0 & \ldots & \hat{d}_0 \\
\vdots & \ddots & \vdots \\
\hat{d}_0 & \ldots & \hat{d}_{N/2-1}
\end{pmatrix}
\]

The values on the diagonal depend solely on the circulant matrix \( \mathbf{C} \) and its eigenvalues \( \lambda_k^{(C)} \) for \( k \in \{0, N/2 - 1\} \). More precisely, we have

\[
d_k = \frac{1 + \lambda_k^{(C)} \exp\left(-i \frac{2\pi}{N} k\right)}{\sqrt{2}} \quad \text{and} \quad \hat{d}_k = \frac{1 - \lambda_k^{(C)} \exp\left(-i \frac{2\pi}{N} k\right)}{\sqrt{2}}.
\]

**Proof.** Using the properties of the interweaving operator, we have

\[
\mathbf{T}_C^F \cdot \mathbf{Ψ}_C = \mathcal{W}(\mathbf{I}_{\frac{N}{2}}, \mathbf{C}_{\frac{N}{2}}) \mathbf{Ψ}_C = \mathcal{W}(\mathbf{Ψ}_C, \mathbf{C}_{\frac{N}{2}} \mathbf{Ψ}_C).
\]

Using the eigenvector–eigenvalue relation (29) of the two circulant matrices \( \mathbf{C} \) and \( \mathbf{I} \) (see Section 4, for the computation of the kth column), we have

\[
[\mathbf{T}_C^F \cdot \mathbf{Ψ}_C]_{-k} = \sqrt{\frac{2}{N}} \begin{pmatrix}
\exp(i4\pi/Nk \cdot 0) \\
\lambda_k \exp(i4\pi/Nk \cdot 0) \\
\exp(i4\pi/Nk \cdot (N/2 - 1)) \\
\lambda_k \exp(i4\pi/Nk \cdot (N/2 - 1))
\end{pmatrix}.
\]

Multiplying Equation 35 by the eigenvector \( \mathbf{Ψ}_k \) from the left yields

\[
[\mathbf{T}_C^F \cdot \mathbf{Ψ}_C]_{-k} = \frac{d_k}{\sqrt{N}} \begin{pmatrix}
\exp(2i\pi/Nk \cdot 0) \\
\ldots \\
\exp(2i\pi/Nk \cdot (N-1)) \\
\end{pmatrix} + \frac{\hat{d}_k}{\sqrt{N}} \begin{pmatrix}
\exp(2i\pi/N(N/2 + k) \cdot 0) \\
\ldots \\
\exp(2i\pi/N(N/2 + k) \cdot (N-1))
\end{pmatrix}.
\]

Then, solve the resulting equation row-wise to find that it has the unique solution (37).

Depending on the structure of \( \mathbf{C} \), we are able to state further simplifications for \( d_k \) and \( \hat{d}_k \), as we see in the following remark.
Remark 2. For the first simplification, let \( m \leq N/2 \) and let the entries of \( C \) be, so that

\[
c_l = \begin{cases} 
    c_{N-l}, & l \in \{1, \ldots, m\}, \\
    0, & l > m.
\end{cases}
\]

In this case, \( C \) is generated by an symmetric stencil with an odd number of entries. Then, it holds

\[
d_k = d_{N/2-k} \quad \text{and} \quad \hat{d}_k = \hat{d}_{N/2-k}.
\]

The second simplification applies for a CI-interpolation and CI-restriction operator with \( C \cdot 1 = 1 \) (i.e., the operators should conserve constants). This yields \( \lambda_0^{(C)} = 1 \), from where follows

\[
d_0 = 0 \quad \text{and} \quad \hat{d}_0 = \sqrt{2}.
\]

We now use Lemma 4 to transform the coarse-grid correction. For the interpolation operator, we obtain diagonal entries \( \{d_0, \hat{d}_0, \ldots, d_{N/2-1}, \hat{d}_{N/2-1}\} \) and for the restriction operator the diagonal entries \( \{\hat{f}_0, f_0, \ldots, \hat{f}_{N/2-1}, f_{N/2-1}\} \). These entries may coincide if the same circulant matrix \( C \) is used for the construction of both operators. Furthermore, we transform the inverse of the system matrix \( \hat{A}^{-1} \) in the spatial dimension into a diagonal matrix consisting of the eigenvalues \( \{\hat{\lambda}_0^{-1}, \ldots, \hat{\lambda}_{N/2-1}^{-1}\} \) of \( \hat{A}^{-1} \). Then, we obtain

\[
\Psi^T C A^{-1} F C \Psi = \Psi^T C F \Psi C \Psi \Psi^T C F \Psi
\]

\[
= \frac{1}{2} \begin{pmatrix} 
    d_0 & \cdots & \hat{d}_{N/2-1} \\
    \hat{d}_0 & \ddots & \ddots \\
    \vdots & \ddots & \ddots \\
    \hat{d}_{N/2-1} & \cdots & \hat{d}_0
\end{pmatrix} \begin{pmatrix} 
    \hat{\lambda}_0^{-1} & \cdots & \hat{\lambda}_{N/2-1}^{-1} \\
    \hat{\lambda}_{N/2-1}^{-1} & \ddots & \ddots \\
    \vdots & \ddots & \ddots \\
    \hat{\lambda}_0^{-1} & \cdots & \hat{\lambda}_0^{-1}
\end{pmatrix} \begin{pmatrix} 
    f_0 & \cdots & \hat{f}_{N/2-1} \\
    \hat{f}_0 & \ddots & \ddots \\
    \vdots & \ddots & \ddots \\
    \hat{f}_{N/2-1} & \cdots & \hat{f}_0
\end{pmatrix}
\]

\[
= \frac{1}{2} \begin{pmatrix} 
    d_0 \hat{\lambda}_0^{-1} f_0 & \cdots & d_0 \hat{\lambda}_0^{-1} \hat{f}_0 \\
    d_0 \hat{\lambda}_0^{-1} \hat{f}_0 & \ddots & \ddots \\
    \vdots & \ddots & \ddots \\
    d_0 \hat{\lambda}_0^{-1} \hat{f}_0 & \cdots & d_0 \hat{\lambda}_0^{-1} f_0
\end{pmatrix} \begin{pmatrix} 
    \hat{\lambda}_0^{-1} \hat{f}_0 & \cdots & \hat{\lambda}_0^{-1} f_0 \\
    \hat{\lambda}_0^{-1} f_0 & \ddots & \ddots \\
    \vdots & \ddots & \ddots \\
    \hat{\lambda}_0^{-1} f_0 & \cdots & \hat{\lambda}_0^{-1} \hat{f}_0
\end{pmatrix}
\]

The values are now scattered over three diagonals. By using the appropriate permutation matrix, we can gather them into new blocks:

\[
P^{-1} \Psi^T C A^{-1} F C \Psi P = \text{diag} \left( B_0, \ldots, B_{N/2-1} \right), \tag{38}
\]

where \( B_f = \begin{pmatrix} d_0 \hat{\lambda}_0^{-1} f_0 & d_0 \hat{\lambda}_0^{-1} \hat{f}_0 \\
    d_0 \hat{\lambda}_0^{-1} \hat{f}_0 & d_0 \hat{\lambda}_0^{-1} f_0 \end{pmatrix} \in \mathbb{R}^{2 \times 2}. \tag{39}
\]

In this structure, we find the classical mode-mixing property of interpolation and restriction operators. This well-known property of standard multigrid iterations interweaves pairs of one low and one high frequency, the “harmonics.”

### 4.2 | Transforming the full iteration matrix

The iteration matrix of PFASST can now be transformed into a block matrix with \( N/2 \) blocks of size \( 2LM \). Each block is associated with a harmonic pair of the spatial problem and therefore with one high and one low spatial frequency. In contrast, the smoother alone is decomposed into \( N \) blocks, which may be associated with only one single frequency. This is summarized in the following theorem.
Theorem 2. Let us have an iteration matrix in the form of Equation 19 with

\[ T = (I - P^{-1}M) \left( I - T_F^C \tilde{P}^{-1} T_F^C M \right), \]

where \( M \) is the collocation matrix, \( T_F^C, T_C^L \) are two circulant interweaved transfer operators, and \( P, \tilde{P} \) are two preconditioners with a matrix in the spatial layer, which is diagonalizable and has the same eigenvector space as the spatial system matrix \( A \). Then, there exists a transformation \( F \) so that

\[
F^{-1}TF = \text{diag} \left( B_0^{(S)}, \ldots, B_{\frac{N-1}{2}}^{(S)} \right) \in \mathbb{R}^{LMN \times LMN}, \text{ with } \]

\[
P_k^{(S)} = \begin{pmatrix}
I - \left( B_k^{(P)} \right)^{-1} B_k^{(M)} & & \\
& I - \left( B_k^{(P)} \right)^{-1} B_k^{(M)} \\
& & I - \left( B_k^{(P)} \right)^{-1} B_k^{(M)}
\end{pmatrix} \in \mathbb{R}^{2LM \times LM}, \text{ for } k = 0, \ldots, N-1
\]

\[
P_k^{(CGC)} = \begin{pmatrix}
I - f_k d_k \left( B_k^{(P)} \right)^{-1} B_k^{(M)} & -\tilde{f}_k d_k \left( B_k^{(P)} \right)^{-1} B_k^{(M)} & \\
& I - f_k d_k \left( B_k^{(P)} \right)^{-1} B_k^{(M)} \\
& & I - f_k d_k \left( B_k^{(P)} \right)^{-1} B_k^{(M)}
\end{pmatrix} \in \mathbb{R}^{2LM \times LM}, \text{ for } k = 0, \ldots, N-1
\]

with matrices \( B_k^{(P)}, B_k^{(M)} \in \mathbb{R}^{LM \times LM} \) for \( k = 0, \ldots, N-1 \) and \( B_k^{(P)} \in \mathbb{R}^{LM \times LM} \) for \( k = 0, \ldots, \frac{N}{2} - 1 \), solely depending on the eigenvalues of \( A \) and \( \tilde{A} \), where

\[
\Psi^T \Psi = \text{diag} \left( B_0^{(P)}, \ldots, B_{\frac{N-1}{2}}^{(P)} \right), \text{ with } B_j^{(P)} = I_L \otimes I_M - \Delta t \cdot \lambda_j^{(A)} I_L \otimes Q_\Delta.
\]

\[
\Psi^T M \Psi = \text{diag} \left( B_0^{(M)}, \ldots, B_{\frac{N-1}{2}}^{(M)} \right), \text{ with } B_j^{(M)} = I_L \otimes I_M - E \otimes N - \Delta t \cdot \lambda_j^{(A)} I_L \otimes Q.
\]

\[
\Psi^T \tilde{P} \Psi = \text{diag} \left( B_0^{(P)}, \ldots, B_{\frac{N-1}{2}}^{(P)} \right), \text{ with } B_j^{(P)} = I_L \otimes I_M - E \otimes N - \Delta t \cdot \lambda_j^{(A)} I_L \otimes Q_\Delta.
\]

We call \( B_j^{(M)}, B_j^{(P)}, \) and \( B_j^{(P)} \) basic blocks. The matrix \( Q_\Delta \in \mathbb{R}^{M \times M} \) is a lower triangular matrix approximating \( Q \) (see Section 2.2).

**Proof.** The proof is rather straightforward. The matrices \( P, M, \) and \( \tilde{P} \) have three layers, separated by Kronecker products like in Equation 32. Applying the transformation in the spatial dimension leads to the basic blocks (43). Similar to Equation 39, we choose the adequate permutation matrices on the layers of subintervals and quadrature nodes, to get the blocks of harmonics. Also, each block of the post-smoother is associated with a mode; hence, we stack harmonic pairs together to \( B_k^{(S)} \) in order to match them with the blocks of the coarse-grid correction \( B_k^{(CGC)} \), performed by the same permutation matrix.

This theorem makes it possible, at least semialgebraically, to analyze the convergence properties of PFASST by computing the spectral radius of each block \( B_k^{(S)}, B_k^{(CGC)} \). Until this point, the choice of the particular problem and the operators yields a rigorous transformation. Hence, the blocks and the full iteration matrix of PFASST have the same eigenvalues. This translates to computing \( N/2 \) eigenvalue decompositions of matrices of size \( 2ML \times 2ML \). As we can see in Equation 43, the basic blocks consist of \( I_L \) and \( E \) on the first layer. However, it is not directly possible to apply the transformation strategy presented earlier to this layer. For an empirical study like LFA, though, only estimates of the spectral radii are needed. This is mainly due to the fact that even the exact spectral radius does not reflect the direct numerical behavior of the method exactly, but rather asymptotically. In the following section, we therefore give up the rigorosity of the transformation in order to find a decomposition of the time collocation blocks into \( L \) much smaller blocks of size \( 2M \times 2M \).

### 4.3 Assuming periodicity in time

To enable the further decomposition of the basic blocks, we exchange in the matrix formulation

\[
E = \begin{pmatrix}
0 & 0 & \cdots & 0 \\
1 & 0 & & \\
\vdots & \ddots & \ddots & \\
0 & 1 & 0
\end{pmatrix} \quad \text{with} \quad \hat{E} = \begin{pmatrix}
0 & 0 & \cdots & 0 \\
1 & 0 & & \\
\vdots & \ddots & \ddots & \\
0 & 1 & 0
\end{pmatrix},
\]

(44)
which introduces time periodicity to the problem and makes the matrix circulant. Hence, it becomes easy to transform

$$[\Psi^{-1} \hat{E}]_{jj} = \exp \left(-i2\pi \frac{j}{L} \right),$$

which makes the time collocation blocks $B_{k,j}^{(M)}$, $B_{k,j}^{(P)}$, and $B_{k,j}^{(E)}$ further decomposable into $NL$ or $NL/2$ blocks of size $M \times M$ or $2M \times 2M$, respectively. This leads directly to the following theorem that can be proved using straightforward computations similar to the ones used before.

**Theorem 3.** Let us have the identical requirements as in Theorem 2, except the use of $\hat{E}$ instead of $E$. Then, there exists a transformation $\hat{F}$ such that

$$\hat{F}^{-1} \hat{F} = \text{diag} \left(B_{0,0}^{(S)}, B_{0,0}^{(CGC)}, B_{0,1}^{(S)}, B_{0,1}^{(CGC)}, \ldots, B_{m,L-1}^{(S)}, B_{m,L-1}^{(CGC)} \right),$$

where the blocks are first iterated over the steps (second index) and then over the collocation nodes (first index). They are defined as

$$B_{k,j}^{(S)} = \begin{pmatrix} I - (B_{k,j}^{(P)})^{-1} (B_{k,j}^{(M)})^{-1} & I - (B_{k,j}^{(P)})^{-1} (B_{k,j}^{(M)})^{-1} \\ I - (B_{k,j}^{(P)})^{-1} (B_{k,j}^{(M)})^{-1} & 0 \end{pmatrix} \in \mathbb{R}^{M \times 2M}$$

and

$$B_{k,j}^{(CGC)} = \begin{pmatrix} I - f_{k} \Delta t (B_{k,j}^{(P)})^{-1} (B_{k,j}^{(M)})^{-1} & I - f_{k} \Delta t (B_{k,j}^{(P)})^{-1} (B_{k,j}^{(M)})^{-1} \\ I - f_{k} \Delta t (B_{k,j}^{(P)})^{-1} (B_{k,j}^{(M)})^{-1} & 0 \end{pmatrix} \in \mathbb{R}^{M \times 2M},$$

with matrices $B_{k,j}^{(P)}, B_{k,j}^{(M)} \in \mathbb{R}^{M \times M}$ for $k = 0, \ldots, N-1, j = 0, \ldots, L-1$ and $B_{k,j}^{(P)} \in \mathbb{R}^{M \times M}$ for $k = 0, \ldots, N-1, j = 0, \ldots, L-1$, solely depending on the eigenvalues of $A$ and $\tilde{A}$, with

$$\Psi^T \Psi = \text{diag} \left( B_{0,0}^{(P)}, \ldots, B_{N-1,L-1}^{(P)} \right),$$

with $B_{k,j}^{(P)} = I - \lambda_{k}^{(A)} \Delta t Q_{\Delta}$,

$$\Psi^T M \Psi = \text{diag} \left( B_{0,0}^{(M)}, \ldots, B_{N-1,L-1}^{(M)} \right),$$

with $B_{k,j}^{(M)} = I - \lambda_{k}^{(A)} \Delta t Q - \exp \left(-i2\pi \frac{j}{L} \right) N$,

$$\Psi^T \tilde{P} \Psi = \text{diag} \left( B_{0,0}^{(P)}, \ldots, B_{N-1,L-1}^{(P)} \right),$$

with $B_{k,j}^{(P)} = I - \lambda_{k}^{(A)} \Delta t Q_{\Delta} - \exp \left(-i2\pi \frac{j}{L} \right) N$. (49)

We denote those blocks as “collocation blocks” in contrast to the time collocation blocks of Theorem 2.

This leaves us with $NL/2$ blocks of size $2M \times 2M$. We identify the matrices $Q$ and $Q_{\Delta}$ as the atomic part of the whole matrix formulation. Further decompositions may only be performed if a decomposition of $Q$ is found. In the case of a $Q \in \mathbb{R}^{1 \times 1}$, the time stepping part reduces to, for example, an implicit Euler. In this case, no eigenvalue computations are necessary anymore, and the Fourier symbols are easily derived from the basic collocation blocks.

**Remark 3.** With the assumption of periodicity in time we loose the initial value, which means that if $u(t,x)$ is a solution of the problem, then $u(t,x) + c$ is also a solution for any $c \in \mathbb{R}$. Hence, the inverses of $B_{k,0}^{(P)}$ and $B_{k,0}^{(E)}$ do not exist and neither do the inverses of the iteration matrix blocks $B_{k,0}^{(T)} = B_{k,0}^{(S)} \cdot B_{k,0}^{(CGC)}$. Our remedy for this problem is to set $B_{k,0}^{(T)}$ to $0$. This blocks belong to constant modes, and we assume that there are no constant error modes, which have to be damped.

Based on this transformation of the iteration matrix, we are now able to investigate the behavior of PFASST for two standard model problems in the following section.

## 5 NUMERICAL EXPERIMENTS

In this section, we show how the convergence properties of PFASST may be examined along the lines of two examples, namely, the diffusion and advection problems. Within this paper though, a full analysis of the influence of all the parameters (such as $N$, $L$, $M$, $\Delta t$), the choice of the quadrature rule, or the PDE parameters is not possible. Therefore, the experiments presented here do not aim for a complete analysis; they should rather be viewed as a recipe to analyze PFASST for a certain class of problems, defined by the requirements we posed for the theoretical results aforementioned.
All computations are performed with $N = 128$ degrees of freedom in space on the fine level and 64 on the coarse level, and no temporal coarsening is used (which is the typical setup for PFASST). For matrices and vectors, the infinity norm is used. For the advection problem, we will use the SDC algorithm with the LU-based preconditioner $Q_\Delta$ as in the work of Weiser,\textsuperscript{35} whereas for the diffusion problem, $Q_\Lambda$ is the standard implicit Euler method. For all experiments, we use $M = 5$ Gauß–Radau nodes on each of the $L = 4$ subintervals of length $\Delta t = 0.1$. Hence, we have $T = 0.4$. The interpolation is constructed such that polynomials up to order 6 are interpolated exactly and the restriction is a scaled transposed interpolation operator, which interpolates polynomials up to the order of 2.

Our main goal will be the estimation of the error by using the block form of the iteration matrix of PFASST. For an arbitrary matrix $T$ consisting of blocks $B_k$, the computation of the norm reduces to

$$
\|T\|^2 = \sup_{x_i \neq 0} \frac{\sum_{k=1}^{m} \|B_k x_i\|^2}{\sum_{k=1}^{m} \|x_i\|^2} = \max_k \frac{\|B_k x_k\|^2}{\|x_k\|^2} = \max \|B_k\|^2
$$

(see the work of Trottenberg et al.\textsuperscript{18} for a proof). The same holds for the computation of the spectral radii. In addition, the effort of computing the eigenvalues of $N/2$ time collocation blocks of size $2LM \times 2LM$ is obviously less than for an $MLN \times MLN$ matrix. With the assumption in Section 4.3, it even reduces to the computation of $NL/2$ collocation blocks of size $2M \times 2M$.

For both cases (time collocation and collocation blocks), we consider the following strategies for the estimation of the error vector $e^\kappa$ of the $\kappa$ iteration:

1. Use the spectral radius $\rho(T)$ of the iteration matrix.
2. Use the norm of the iteration matrix $\|T\|$.
3. Use the norm of the $x$th potency of the iteration matrix $\|T^x\|$.
4. Apply $x$th times the transformed iteration matrix to the spatial modes of the error vector.

The first strategy is based on the inequality for consistent matrix norms $\| \cdot \|$ and each $\kappa \in \mathbb{N}$

$$
\rho(A) \leq \|A^\kappa\|^\frac{1}{\kappa}
$$

(see the work of Kelley\textsuperscript{40}). Strategies 2 and 3 rely on the inequality

$$
\|e^\kappa\| = \|T^x e^0\| \leq \|T^x\| \|e^0\| \leq \|T\|^x \|e^0\|.
$$

(52)

Note that the iteration matrix is separated from the initial error vector $e^0$, and therefore, an a priori estimation of the relative error reduction is possible for this strategies. In contrast, the error vector $e^0$ (i.e., the analytical solution) has to be known for strategy 4, making it an a posteriori strategy. If time collocation blocks are used, the computation following strategy 4 yields the correct error for each iteration. Using collocation blocks, this approach just provides another estimate. We note that strategy 4 differs from merely running the full method and measuring the error: After transformation, it is possible to compute the error propagation

$$
\|F_{\kappa} e\| = \|F_{\kappa} T e\| = \|\text{diag}(B_{\kappa}) F e\| = \|B_{\kappa}^* e\|_{LM}
$$

much easier, where $e^0 = m_j \otimes 1_L \otimes 1_M$ and $m_j$ is the $j$th space mode. Thus, instead of running the full method for this particular initial value, the results of the previous section allows us to simply monitor the temporal evolution of the $j$th space mode and its harmonic counterpart. If we in addition assume periodicity in time, this is further reduced to the evolution of the harmonics over $M$ collocation nodes only. We finally note that all errors reported in the following sections are the errors measured with respect to the exact solution of the linear collocation system, even when the decomposition of the error vector is shown for the analytical error.

### 5.1 Diffusion problem

The elliptic Poisson problem is often used in the multigrid literature to demonstrate the basic ideas of multigrid.\textsuperscript{18} Hence, the time-dependent parabolic version of it (i.e., the classical heat equation) is a canonical candidate for the analysis of a multigrid-like time integration method such as PFASST.

The problem in one spatial dimension is given by

$$
\begin{align*}
u u_t &= \nu \Delta u, & x \in [0, 1] \text{ and } t \in [0, T] \\
 (x, 0) &= u_0(x), & u(0, t) = u(1, t) \quad t \in [0, T]
\end{align*}
$$

(53)
for a time \( T > 0 \) and the diffusion coefficient \( v > 0 \). Using second-order finite differences on an isometric grid, we get a simple discretization in the spatial dimension with

\[
X = [x_1, \ldots, x_N], \text{ with } x_j = \frac{j - 1}{N} \quad \text{and} \quad \Delta x = \frac{1}{N},
\]

which leads to a system of linear ODEs

\[
U_i(t) = AU(t), \quad t \in [0, T] \quad \text{and} \quad U(0) = [u(x_1, 0), \ldots, u(x_N, 0)],
\]

with \( A = \frac{v}{(\Delta t)^2} \begin{pmatrix} 2 & -1 & 0 & \cdots & -1 \\ -1 & 2 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ -1 & 0 & \cdots & -1 & 2 \end{pmatrix} \in \mathbb{R}^{N\times N}. \)

The spacing in the temporal and spatial dimensions clearly have an effect on the numerical properties, and we denote

\[
\mu = \frac{v \Delta t}{\Delta x^2}
\]

as the dispersion relation number of the heat equation. Because the matrix \( A \) is circulant, the spectral decomposition in eigenvalues and eigenvectors is easily computed. For the eigenvalues \( \lambda_k \) and normal eigenvectors \( \psi_k, k \in \{0, \ldots, N - 1\} \), we have

\[
\lambda_k = \frac{4v}{\Delta x^2} \sin^2 \left( \frac{k\pi}{N} \right) \quad \text{and} \quad \psi_k = \frac{1}{\sqrt{N}} \exp \left( \frac{i2\pi k}{N} \cdot 0 \right), \ldots, \exp \left( \frac{i2\pi k}{N} \cdot (N - 1) \right)^T.
\]

5.1.1 The error vector

For initial values given by the function

\[
u_0(x) = \sin(2\pi x k), \quad k \in \{1, \ldots, N - 1\}, \quad x \in [0, 1],
\]

we know that the solution to our PDE with periodic boundary conditions is given by

\[
u(t, x) = \exp \left( -v(2\pi k)^2 t \right) \sin(2\pi x k).
\]

Usually, the PFASST algorithms starts with a vector where the initial value is spread on each node; that is, we have the initial error vector

\[
e^0 = \begin{pmatrix} 1 - \exp(-v(2\pi k)^2(t_0 + \tau_1)) \\ \vdots \\ 1 - \exp(-v(2\pi k)^2 T) \end{pmatrix} \otimes \begin{pmatrix} \sin(2\pi kx_1) \\ \vdots \\ \sin(2\pi kx_N) \end{pmatrix}.
\]

With the iteration matrix, we compute the succeeding error vector for PFASST as

\[
Te^k = e^{k+1}.
\]

Like the iteration matrix, the error vector \( e^k \) of the kth iteration itself can be transformed and decomposed into parts belonging to a certain mode and associated with the time-collocation (TC) block of the iteration matrix. We can write

\[
F^{-1}TF^{-1}e^k = F^{-1}e^{k+1}.
\]

The transformed error is thus \( \hat{e}^k = F^{-1}e^k \), following precisely the transformation procedure described in Section 4.1. The initial value function \( u_0(x) \) decomposes into two modes, which are represented by spatial Fourier space functions

\[
\sin(2\pi k x_j) = \frac{1}{2i} \exp \left( \frac{i2\pi}{N} k x_j \right) - \frac{1}{2i} \exp \left( -\frac{i2\pi}{N} k x_j \right)
\]

\[
= \sqrt{\frac{N}{2i}} \psi_k j - \sqrt{\frac{N}{2i}} \psi_{N-k} j
\]

and belong to two different harmonics. This reduces our analysis to the blocks belonging to these certain harmonics, which are \( B^{(T)}_k, B^{(T)}_{\frac{N}{2} - k} \) of sizes \( 2LM \) for \( k = 0, \ldots, N/2 - 1 \) in the case that time collocation blocks are considered and \( B^{(T)}_{k_1}, B^{(T)}_{\frac{N}{2} - k_2} \) of sizes \( 2M \) for \( k = 0, \ldots, N/2 - 1 \) and \( j = 0, \ldots, L - 1 \) if collocation blocks are considered.

Remark 4. In the following, we compute the error with respect to the solution of the collocation problem (i.e., the solution of the linear system that PFASST solves). To this end, we use the transformation described earlier to also modify the initial error
BOL TEN ET AL.

The error estimates from strategies 1 to 3, compared to the actual error plotted against the number of iterations of PFASST, for the diffusion problem with an initial value function of \( \sin(2\pi 8x) \). With this so-called transformed collocation error, we are also able to utilize the block structure, because the collocation error decomposes in the same modes as the error to the analytical solution.

5.1.2 | Error prediction

We choose \( \nu \) so that \( \mu = 10 \). In Figure 2, observing the solid line of the actual error measured during the iterations, we first see a short-term convergence behavior until roughly \( 10^{-4} \), which is then followed by a much slower long-term convergence phase. We see that the use of the norm of the 4th potency of the iteration matrix \( T \) (strategy 3) is well suited to capture the long-term and thus asymptotic convergence behavior. This is of course also true for strategy 1, using the spectral radius of the iteration matrix. Similar plots for various initial value functions \( \sin(2\pi kx) \), for \( k \in \{1, \ldots, N/2-1\} \), were inspected and showed the same behavior for the long-term convergence. In particular, there is no significant difference between time collocation and collocation blocks. However, the norm of the iteration matrix is greater than 1 for most cases, as a survey over different \( \mu \in (0.01, 100) \) and \( L \in \{2, \ldots, 50\} \) showed. This renders strategy 2 useless for most of the cases we considered so far.

The short-term convergence on the other hand is not captured by the first three strategies. In contrast, strategy 4 does this very well, as we see in Figure 3. We also see that the short-term convergence is faster for initial values with a small wave number \( k \) and that the long-term convergence speed is almost independent from the initial value. Our interpretation is that PFASST is more efficient in reducing the low-frequency error modes in space. After the first convergence phase, the error consists of a mixture of modes, which is reduced by PFASST likewise, independently from the initial value frequency.

Here, we actually see a difference between the different types of blocks: For the error prediction of the first iterations, using strategy 4 with collocation blocks is not as accurate as using time collocation blocks. In contrast, no differences in the quality of the error prediction are notable in the long-term convergence phase, again.

5.2 | Advection problem

The second prototype problem is the one-dimensional (1D) advection equation, given by

\[
\begin{align*}
  u_t &= cu_x, \quad x \in [0, 1] \text{ and } t \in [0, T] \\
  u(x, 0) &= u_0(x), \quad u(0, t) = u(1, t) \quad t \in [0, T],
\end{align*}
\]

with advection coefficient \( c > 0 \). The discretization is done in the same manner as Equation 55, but we use an upwind difference stencil of the order 3, instead of a central difference stencil. This yields again a circulant matrix \( F_D \), with eigenvalues and eigenvectors according to Equation 29. For the numerical experiments, we use advection speed \( c = 4.88 \cdot 10^{-3} \), resulting in a CFL number of \( 62.5 \cdot 10^{-3} \). Note that as for the diffusion case, this value is chosen such that the solution as well as PFASST show interesting behavior (and converge reasonably well). In principle, the analysis works for other choices equally well. The discretization in space and time is similar to the discretization in the previous section.
5.2.1 The error vector

For an initial value function $u_0$, the solution reads

$$u(t, x) = u_0(x - ct).$$

We use again the initial values given by Equation 57. The initial values are spread on each node, and this yields the initial error vector

$$e^0 = \left(e^0_1, \ldots, e^0_L\right)^T$$

with

$$e^0_n = \left(e^0_{n,1}, \ldots, e^0_{n,M}\right)^T,$$

$$e^0_{j,m} = \left(u_0(x_1) - u_0(x_1 - c(t_{j-1} + \tau_m)), \ldots, u_0(x_N) - u_0(x_N - c(t_{j-1} + \tau_m))\right).$$

When the class of initial value (57) is used, the initial values can be decomposed again into two modes, belonging to different harmonics. The analysis is thus again reduced to certain harmonic blocks $B^{(T)}_k \cdot B^{(T)}_{\frac{\pi}{\Delta x} k} \cdot B^{(T)}_{j,\frac{\pi}{\Delta x} k} \cdot B^{(T)}_{j,\frac{\pi}{\Delta x} k}$, respectively. Note, that this computation of the error vector works even if only a numerical solution to the problem is given.

5.2.2 Error prediction

For the advection problem, the four strategies yield significantly different results than for the diffusion problem. In Figure 4, we now observe three phases of convergence: two rapid phases at the beginning and at the end and one almost stagnating phase in

![Figure 3](image-url)
the middle. We observed these phases for all initial wave numbers $\kappa$, with the peculiarity of a decreasing and almost vanishing first phase for increasing $\kappa$. Regarding the different strategies, we see that only the spectral radii (strategy 1) is able to capture the first phase, whereas the norm of the powers of the iteration matrix (strategy 3) captures the last phase, which strategy 1 captures as well.

As we can see in Figure 5, strategy 4 is again successful in exactly predicting the error, when time collocation blocks are used. On the other hand, for the advection equation, the use of collocation blocks only serve as an assessment for initial values with high $k$, and then only for the first phase. Obviously, the assumption of periodicity in time is not valid for advection-dominated problems. Thus, time collocation blocks should be considered in this case.

Note that for the two numerical experiments shown here, we chose setups, where one observes a development of the error, which is more complex than just an exponential decay. However, these observations vary heavily with the setup, depending on the choice of the parameters. For example, the convergence speed is strongly influenced by the dispersion relation number. We emphasize that for the advection problem, a setup was chosen that resulted in faster convergence speed than the setup for the diffusion problem.

### 6 Conclusion and Outlook

In this paper, we decomposed the PFASST algorithm into its atomic parts. Using analogies to classical iterative methods such as Gauß–Seidel and Jacobi, we described PFASST for two levels and linear problems as a combination of a highly parallel approximative block Jacobi solver on the fine level and a serial approximative block Gauß–Seidel solver on the coarse level. With this, we could show that for linear problems, PFASST is a multigrid algorithm for the composite collocation problem in space and time. We stated the underlying composite collocation problem in matrix formulation, spanning the full domain in space and time, and decomposed it into three layers: spatial decomposition, time stepping, and quadrature nodes. With suitable transformations, we could show the similarity of PFASST’s iteration matrix to a block-diagonal matrix, containing either $N/2$ time collocation blocks of size $2ML$ or $NL/2$ collocation blocks of size $2M$. In the first case, the analysis is rigorous, whereas in the second case, periodicity in time is assumed.

We identified four different strategies to test the convergence properties of PFASST using the block diagonalization of the iteration matrix. Along the lines of two prototype problems, we investigated the quality of the predictions given by these strategies compared to the numerical results from PFASST. We explored the effect of PFASST on different modes of the solution, depending on the initial values.

With a suitable measure for the convergence speed of PFASST at hand, the next key step is to estimate the parallel performance of this algorithm in comparison to serial runs of SDC. To this end, block diagonalizations of PFASST and SDC can be compared, following the strategies presented in this work. This would augment the current speedup considerations of PFASST as stated in the work of Emmett et al.\(^{16}\) by providing estimates for the actual iteration counts. In addition, we have identified the following topics as relevant for further studies.
Detailed parameter and component studies. So far, we have only investigated simple 1D problems, demonstrating how the LFA of the iteration matrix can be used to predict the convergence behavior of PFASST for different situations. These examples can serve as a blueprint for a much deeper and more detailed analysis of PFASST’s convergence properties for various problems. Also, the matrix formulation of PFASST allows us to exchange parts more easily. We can test other smoothers other than SDC, change the quadrature rules used on the subintervals, vary interpolation and restriction on space (and even time), and apply iterative solvers like standard multigrid in space for inverting the spatial operators.

Nonlinear functions. In this work, we focused on linear problems in order to apply the LFA. However, for consistency with the existing literature, the notation used here is derived from FAS and therefore is also applicable to nonlinear right-hand sides. Then, however, the matrices representing the right-hand side of the system of ordinary differential equations become nonlinear operators \( f(U(\tau)) \) and a straightforward analysis of the properties using LFA is not possible and other options will have to be explored.

Algorithmic extension. In contrast to Parareal, the PFASST algorithm is designed to use more than two levels. Due to the simplification of the notation and the rigor of the argumentation chain, this fact was not exploited. For the same reasons, the interpolation and restriction matrices affected only the spatial dimension, although it is possible to construct coarse levels with less quadrature nodes than on the fine level. The effects on the formalism in Section 4 would be minor. This will change dramatically, if a coarse level is constructed where two or more subintervals from the fine level are merged to one (coarsening in time). This would be a step in the direction of a full space-time multigrid approach, but some work is needed to adjust the formalism in Section 4 for a similar convergence analysis. Also, a suitable coarsening strategy will have to be defined, which allows to perform either temporal or spatial coarsening or both. Another step towards a more multigrid-like behavior would be to use the exact solution on the coarsest level instead of one or more SDC sweeps, but first brief experiments showed no
significant difference between the use of the exact solution or the use of SDC sweeps. Then, while each spatial problem is assumed to be solved exactly in this work, a more natural approach is to use, for example, a fixed number of V-cycles of a spatial multigrid solver.\textsuperscript{41,42} Analyzing the impact of these “inexact solves” is also left for future work.

**Rigorous convergence analysis.** The usual attempt in multigrid theory for a rigorous convergence analysis contains the proof of the smoothing and approximation properties. Both endeavors are difficult on their own, but, in our case, are further impeded by the matrices $Q, Q_\Delta$. These matrices are dense and yield no obvious structural properties, which could be exploited. First steps towards a more rigorous analysis would be to resolve this problem. However, the multigrid perspective presented in this work will help to analyze the asymptotic convergence properties of PFASST (i.e., the behavior of this algorithm for the stiff and non-stiff limits). This corresponds, for example, to $\Delta t \to 0$ and $\Delta x \to 0$, respectively, and the results obtained here may pave the way for a first rigorous convergence proof of PFASST for these cases. In addition, while Figures 2 and 4 demonstrate that already for small $L$ the asymptotic convergence speed of PFASST to the solution of the composite collocation problem is captured quite well, the behavior of PFASST for increasing numbers of time steps $L$ on a fixed interval in time is also a highly interesting topic and a natural extension. We leave these asymptotic convergence considerations for future work.

**REFERENCES**

1. Burrage K. Parallel methods for ODEs. Adv Comput Math. 1997;7:1–3. Available from: https://doi.org/10.1023/A:1018997130884
2. Iserles A, Norsett S. On the theory of parallel Runge-Kutta methods.IMA J Numer Anal. 1990;10(4):463–488.
3. Butcher J. Order and stability of parallel methods for stiff problems. Adv Comput Math. 1997;7(1): 79–96.
4. Christlieb AJ, Macdonald CB, Ong BW. Parallel high-order integrators. SIAM J Sci Comput. 2010;32(2):818–835. Available from: https://doi.org/10.1137/09075740X
5. Güttel S. A parallel overlapping time-domain decomposition method for ODE's. In: Bank R, Holst M, Widlund O, Xu J, editors. Domain Decomposition Methods in Science and Engineering XX. Lecture Notes in Computational Science and Engineering. vol 91. Berlin, Heidelberg: Springer; 2013. p. 459–466.
6. Maday Y, Ronquist EM. Parallelization in time through tensor-product space-time solvers. Comptes Rendus Math. 2008;346(1–2):113–118.
7. Sheen D, Sloan IH, Thomée V. A parallel method for time discretization of parabolic equations based on Laplace transformation and quadrature.IMA J Numer Anal. 2003;23(2):269–299.
8. Gander MJ. A waveform relaxation algorithm with overlapping splitting for reaction diffusion equations. Numer Linear Algebra Appl. 1999;6(2):125–145.
9. Vandewalle S, Roose D. The parallel waveform relaxation multigrid method. Paper presented at: Proceedings of the Third SIAM Conference on Parallel Processing for Scientific Computing; 1989 Dec 1–4; Los Angeles, CA. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics; 1989. p. 152–156.
10. Nievergelt J. Parallel methods for integrating ordinary differential equations. Commun ACM. 1964;7(12):731–733. Available from: https://doi.org/10.1145/355588.365137
11. Chartier P, Philippe B. A parallel shooting technique for solving dissipative ODE's. Computing. 1993;51(3–4):209–236. Available from: https://doi.org/10.1007/BF02238534
12. Bellen A, Zennaro M. Parallel algorithms for initial-value problems for difference and differential equations. J Comput Appl Math. 1989;25(3):341–350. Available from: https://doi.org/10.1016/0377-0427(89)90037-X
13. Lions JL, Maday Y, Turinici GA. A “parareal” in time discretization of PDE’s. Comptes Rendus de l’Académie des Sci - Ser I - Math. 2001;332:661–668. Available from: https://doi.org/10.1016/S0764-4442(00)01793-6
14. Gander M, Jiang YL, Li RJ. Parareal Schwarz waveform relaxation methods. In: Bank R, Holst M, Widlund O, Xu J, editors. Domain Decomposition Methods in Science and Engineering XX, Lecture Notes in Computational Science and Engineering, vol. 91. Berlin Heidelberg: Springer, 2013. p. 451–458. Available from: https://doi.org/10.1007/978-3-642-35275-1_53
15. Minion ML. A hybrid parareal spectral deferred corrections method. Commun Appl Math Comput Sci. 2010;5(2):265–301. Available from: https://doi.org/10.2140/camcos.2010.5.265
16. Emmett M, Minion ML. Toward an efficient parallel in time method for partial differential equations. Commun Appl Math Comput Sci. 2012;7:105–132. Available from: https://doi.org/10.2140/camcos.2012.7.105
17. Brandt A. Multi-level adaptive solutions to boundary-value problems. Math Comput. 1977;31(138):333–390.
18. Trottenberg U, Osterlee CW, Schuller A. Multigrid. Cambridge, MA, USA: Academic Press; 2000.
19. Hackbusch W. Parabolic multigrid methods. Paper presented at: Proceedings of the Sixth International Symposium on Computing Methods in Applied Sciences and Engineering, VI; 1984; Versailles, France. p. 189–197. Available from: http://dl.acm.org/citation.cfm?id=645818.669246
20. Lubich C, Ostermann A. Multi-grid dynamic iteration for parabolic equations. BIT Numer Math. 1987;27(2):216–234. Available from: https://doi.org/10.1007/BF01934186
21. Vandewalle S, Van de Velde E. Space-time concurrent multigrid waveform relaxation. Ann Numer Math. 1994;1:347–363.
22. Falgout R, Friedhoff S, Kolev TV, MacLachlan S, Schroder JB. Parallel time integration with multigrid. SIAM J Sci Comput. 2014;36(6):C635–C661.
23. Horton G, Vandewalle S. A space-time multigrid method for parabolic partial differential equations. SIAM J Sci Comput. 1995;16(4):848–864. Available from: https://doi.org/10.1137/0916050
24. Gander MJ, Neumüller M. Analysis of a new space-time parallel multigrid algorithm for parabolic problems. SIAM J Sci Comput. 2016;38(4):A2173–A2208.
25. Gander MJ, Vandewalle S. Analysis of the parareal time-parallel time-integration method. SIAM J Sci Comput. 2007;29(2):556–578. Available from: https://doi.org/10.1137/05064607X
26. Berrut JP, Trefethen LN. Barycentric Lagrange interpolation. SIAM Rev. 2004;46(3):501–517.
27. Dutt A, Greengard L, Rokhlin V. Spectral deferred correction methods for ordinary differential equations. BIT Numer Math. 2000;40(2):241–266. Available from: https://doi.org/10.1023/A:1022338906936
28. Frank R, Ueberhuber CW. Iterated defect correction for the efficient solution of stiff systems of ordinary differential equations. BIT Numer Math. 1977;17(2):146–159. Available from: https://doi.org/10.1007/BF01932286
29. Huang J, Jia J, Minion M. Accelerating the convergence of spectral deferred correction methods. J Comput Phys. 2006;214(2):633–656. Available from: https://doi.org/10.1016/j.jcp.2005.10.004
30. Layton AT, Minion ML. Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics. J Comput Phys. 2004;194(2):697–715. Available from: https://doi.org/10.1016/j.jcp.2003.09.010
31. Minion ML. Semi-implicit projection methods for incompressible flow based on spectral deferred corrections. Workshop on Innovative Time Integrators for PDEs prefix. Appl Numer Math. 2004;48(3-4):369–387. Available from: https://doi.org/10.1016/j.apnum.2003.11.005
32. Bourlioux A, Layton AT, Minion ML. High-order multi-implicit spectral deferred correction methods for problems of reactive flow. J Comput Phys. 2003;189(2):651–675. Available from: https://doi.org/10.1016/S0021-9991(03)00251-1
33. Guibert D, Tromeur-Dervout D. Parallel deferred correction method for CFD problems. In: Kwon J, Ecer A, Satofuka N, Periaux J, Fox P, editors. Parallel Computational Fluid Dynamics 2006. Amsterdam: Elsevier Science BV, 2007. p. 131–138. Available from: https://doi.org/10.1016/B978-044453035-6/50019-5
34. Minion ML, Williams SA. Parareal and spectral deferred corrections. Paper presented at: AIP Conference Proceedings, vol. 1048; 2008. p. 388. Available from: http://link.aip.org/link/10.1063/1.2990941
35. Weiser M. Faster SDC convergence on non-equidistant grids with DIRK sweeps 2013. ZIB Report 13–30. Available from: http://opus4.kobv.de/opus4-zib/files/1866/ZR-13-30.pdf
36. Winkel M, Speck R, Ruprecht D. A high-order Boris integrator. J Comput Phys. 2015;295:456–474.
37. Speck R, Ruprecht D, Emmett M, Minion M, Bolten M, Krause R. A multi-level spectral deferred correction method. BIT Numer Math. 2015;55(3):843–867.
38. Koehler F. Pfasst tikz; 2015. Available from: https://github.com/Parallel-in-Time/pfasst-tikz
39. Friedhoff S, MacLachlan S. A generalized predictive analysis tool for multigrid methods. Numer Linear Algebra Appl. 2015;22(4):618–647. Available from: https://doi.org/10.1002/nla.1777.
40. Kelley CT. Iterative methods for linear and nonlinear equations. Raleigh: North Carolina State University; 1995.
41. Minion ML, Speck R, Bolten M, Emmett M, Ruprecht D. Interweaving PFASST and parallel multigrid. SIAM J Sci Comput. 2015;37(5):S244–S263.
42. Speck R, Ruprecht D, Emmett M, Bolten M, Krause R. A space-time parallel solver for the three-dimensional heat equation. Parallel Computing: Accelerating Computational Science and Engineering (CSE), Advances in Parallel Computing. 2014;25:263–272. IOS Press. Available from: https://doi.org/10.3233/978-1-61499-381-0-1-263

How to cite this article: Bolten M, Moser D, Speck R. A multigrid perspective on the parallel full approximation scheme in space and time. Numer Linear Algebra Appl. 2017;24:e2110. https://doi.org/10.1002/nla.2110