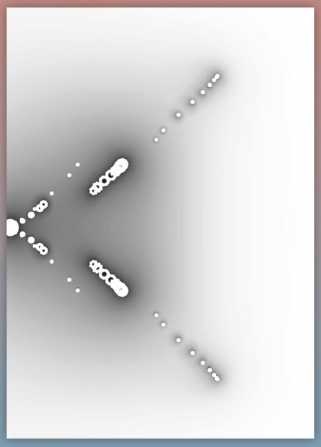
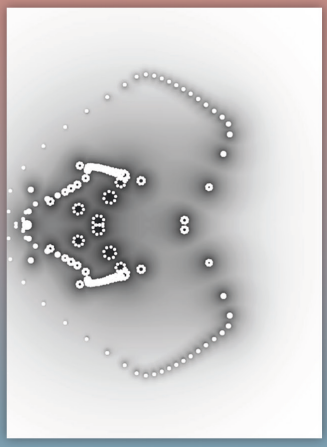
$$\mathbf{T}_{\mathrm{MG}}(\nu,\eta) = \left(\mathbf{I} - \mathbf{P}_{\mathrm{post}}^{-1}\mathbf{M}\right)^{\nu}\left(\mathbf{I} - \mathbf{T}_C^F\tilde{\mathbf{M}}^{-1}\mathbf{T}_F^C\mathbf{M}\right)\left(\mathbf{I} - \mathbf{P}_{\mathrm{pre}}^{-1}\mathbf{M}\right)^{\eta}$$

$$\mathbf{T}_{\mathrm{PFASST}} = \left(\mathbf{I} - \hat{\mathbf{P}}_{[t_0,T]}^{-1}\mathbf{M}_{[t_0,T]}\right)\left(\mathbf{I} - \mathbf{T}_C^F\tilde{\mathbf{P}}^{-1}\mathbf{T}_F^C\mathbf{M}_{[t_0,T]}\right)$$





$$\Lambda_\epsilon(\mathbf{A}) = \{\lambda \in \mathbb{C} \mid \exists \mathbf{x} \in \mathbb{C}^n \setminus \{0\}, \exists \mathbf{F} \in \mathbb{C}^{n\times n} : (\mathbf{A}+\mathbf{F})\mathbf{x} = \lambda\mathbf{x}, \|\mathbf{F}\| \leq \epsilon\}$$

$$\frac{S(K_s,K_p,L,P)}{S(K_s,K_p,L,2P)} = \frac{T(K_p,L,2P)}{T(K_p,L,P)} = \frac{2P-1+\frac{L}{2P}K_p(\nu+\alpha)}{P-1+\frac{L}{P}K_p(\nu+\alpha)}$$

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

Dieter Moser

IAS Series
Band / Volume 36
ISBN 978-3-95806-315-0

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
Forschungszentrum

Schriften des Forschungszentrums Jülich
Reihe IAS Band / Volume 36

Forschungszentrum Jülich GmbH
Institute for Advanced Simulation (IAS)
Jülich Supercomputing Centre (JSC)

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

Dieter Moser

Schriften des Forschungszentrums Jülich
Reihe IAS

Band / Volume 36

ISSN 1868-8489

ISBN 978-3-95806-315-0

Bibliografische Information der Deutschen Nationalbibliothek.
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte Bibliografische Daten
sind im Internet über http://dnb.d-nb.de abrufbar.

Dissertation an der Universität Kassel
Fachbereich 10 Mathematik und Naturwissenschaften

Gutachter:	Prof. Matthias Bolten Prof. Andreas Meister
Disputation:	23.11.2017
Herausgeber und Vertrieb:	Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag 52425 Jülich Tel.: +49 2461 61-5368 Fax: +49 2461 61-6103 zb-publikation@fz-juelich.de www.fz-juelich.de/zb
Umschlaggestaltung:	Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH
Druck:	Grafische Medien, Forschungszentrum Jülich GmbH
Copyright:	Forschungszentrum Jülich 2018

Schriften des Forschungszentrums Jülich
Reihe IAS, Band / Volume 36

D 34 (Diss., Kassel, Univ., 2017)

ISSN 1868-8489
ISBN 978-3-95806-315-0

Persistent Identifier: urn:nbn:de:0001-2018031401

The complete volume is freely available on the Internet on the Jülicher Open Access Server (JuSER)
at www.fz-juelich.de/zb/openaccess

Für meine Großmutter, die mir das Rechnen beibrachte

Contents

1. Introduction

> Cuius auctores quanto sunt iuniores,
> tanto perspicaciores.
>
> Priscianus Caesariensis - 526

From careful observations, scientists derive rules to describe phenomena in nature. These rules are implemented in form of algorithms in order to simulate these phenomena. Nowadays, simulations are a vital element of numerous research fields, which study not only natural phenomena, but also societal or macro-economical systems. For simulations of a certain size and complexity, an adequate amount of computing power is required. Such simulations are found in fields like molecular dynamics or material science, where sometimes billions of atoms need to be simulated to observe emergent structures. Other fields, which employ such simulations, include weather and climate sciences, where the degrees of freedom simply surpass the capacities of a personal computer. For these simulations a high performance computing approach is required. A more complete list of such fields and their current state of research is found in [1]. The task of the mathematician is to study the stability, accuracy, and cost of computation of the numerical methods used in these simulations. In the last decades, the rapid increase of processors per machine created a need for concurrent computation. This resulted in the reformulation of the existing algorithms and development of new algorithms, which in turn also need to be studied.

Most of the fields mentioned above model their problems in the form of partial differential equations. For this class of equations many effective parallel methods already exist. Independent of the method chosen, the model of nature has to be transformed into a model that may be processed by a computer. Since computers are only able to process and store a finite number of quantities with a limited precision, the newly transformed model has to represent nature with this finite number of quantities. One way to do so is to decompose the computational domain of the problem into a finite grid. Imagine, for example, a simulation of the wing of an airplane. The computational domain consists of the wing itself and the floating air around it. A system of partial differential equations describes how the pressure, the temperature, and the speed of the air interact with the mechanical forces of the wing at every point in this computational domain. When we decompose this domain, we are only interested in the physical quantities on a finite number of grid points. With, for example, the finite difference scheme, we derive a set of rules from the partial differential equations. These rules describe how the values on

the grid points are related to one another. Together with the grid, these rules constitute the new model, which is then processed by the computer.

To process this model in parallel, we decompose the computational domain and distribute the parts of the domain to different processors. To understand the limits of this parallelization strategy, we imagine a cube-shaped computational domain, which is divided into smaller cubes. The size of the cube determines the number of grid points it contains and thus the computational effort needed. During the computation, each smaller cube needs some of the data stored in other cubes. When, for example, a diffusive phenomena is observed, it should be sufficient to exchange information between neighboring cubes. Whereas, in the case of electromagnetic phenomena, the cubes, most probably, have to exchange information over long distances. Consequently, the respective processors communicate with each other globally. Assume, we have a greater number of processors available. This results in a greater number of smaller cubes. Each cube requires less computational effort, but overall they cause more communication. When the number of processors reaches a critical level, the communication represents the bulk of the total effort. This means that additional processors do not result in a shorter computational time. The parallelization strategy described above reaches the so called strong scaling limit. A way to overcome this limit, is to scale the number of grid points with the number of processors. This allows computations of a greater size or a higher accuracy. Computational domains of a greater size do not, however, necessarily provide additional insights. In our example with the wing, the speed and pressure too far away from the wing do not always provide useful information, even if we study effects like wake turbulences. Thus, a greater computational domain is not needed. Computational domains of a higher accuracy are achieved by using more grid points. This means we are able to state the quantities with a higher precision. However, this is not always beneficial. For example, when the uncertainties of the boundary conditions are too high, the precision loses its significance. Let us assume we are interested in a higher precision for a time dependent problem. To treat the temporal dimension, a time stepping method is needed. Since the current state of a physical system only affects its future state, the natural design of a time stepping method is sequential and forward in time. This sequential design results in a new bottleneck when the spatial resolution is increased in order to achieve a higher accuracy. To maintain the stability of the time stepping method, the temporal resolution has to match the spatial resolution. Even if a greater number of processors makes it possible to compute one time step with a higher accuracy in the same computation time, we will need more time steps and thus more computation time overall. This means that the weak scaling limit is reached. The only way to push these scaling limits further, is to apply the surplus of processors to the temporal dimension instead of the spatial dimension.

This approach is called parallel-in-time integration. In [2], such methods are classified into parallelization *across the step, across the method*, and *across the problem*. Direct time-parallel methods mostly belong to the class of parallelization across the method;

examples include certain parallel Runge-Kutta methods [3, 4]. Only modest parallel speedup is expected for these methods, since the number of processing units used for the parallelization are limited by e.g., the number of Runge-Kutta stage values. Other direct methods for parallel-in-time integration include RIDC [5], ParaExp [6], tensor-product space-time solvers [7], or methods using the Laplace transformation [8]. The class of parallelization across the problem includes methods that decompose the problem into subproblems, which are solvable in a parallel manner and couples these subproblems using an iterative method. The most prominent examples are waveform relaxation methods [9, 10], which are part of the broad area of domain decomposition methods.

The first idea of parallel-in-time integration belongs to the class parallelization across the step. More precisely, one of the first parallel-in-time integration methods is a multiple-shooting method introduced by Nievergelt in 1964 [11]. Further examples of multiple-shooting methods, which are able to perform parallel-in-time are found in [12, 13]. Among them, in 2001 by Lions et al., *Parareal* [14] renewed the interest in parallel-in-time methods and influenced other methods (see [15]) and even inspired the development of new methods. In [16], the Parareal approach is coupled to the spectral deferred correction (SDC) method, which is an iterative solver for the collocation problem. This approach is extended to the "parallel full approximation scheme in space and time" (PFASST) in [17]. For time-dependent PDEs, parallel-in-time integration using PFASST is a promising way to accelerate existing space-parallel approaches beyond their scaling limits [18, 19]. While many examples of potential uses exist, a solid and reliable mathematical foundation is still needed. This makes PFASST an excellent study object. In Section 1.1, we introduce Parareal as a foundation for the introduction of PFASST. More precisely, we will show how PFASST adopts and evolves the characteristics of Parareal by interweaving its iterations with those of the local SDC scheme. In addition to Parareal and SDC, features of the nonlinear multigrid theory also played a vital role in the development of PFASST. We will introduce these features along the lines of representative space-time multigrid methods in Section 1.2.

All the classes mentioned above are not strictly separated from each other. Often, methods of one class may be reformulated to conform with another class. A prominent example of this is Parareal itself: it was reformulated as a multiple-shooting method as well as a multigrid method in [20], which in turn paved the way for a comprehensive analysis of Parareal. In this work we will also reformulate PFASST, to achieve a more profound and rigorous mathematical understanding of PFASST. We will establish a close connection between PFASST and standard multigrid methods, providing a comprehensive approach to the mathematical analysis and algorithmic optimization. To this end, we will focus on linear autonomous ordinary differential equations. This will enable us to present SDC and MLSDC in the form of a stationary iterative scheme in Section 1.3. Together with the introduction of Parareal and space-time multigrid methods in Sections 1.1 and 1.2, the classical formulation of PFASST in algorithmic form presented in Section 1.3, represents the foundation of this work. From there on, we will leave fa-

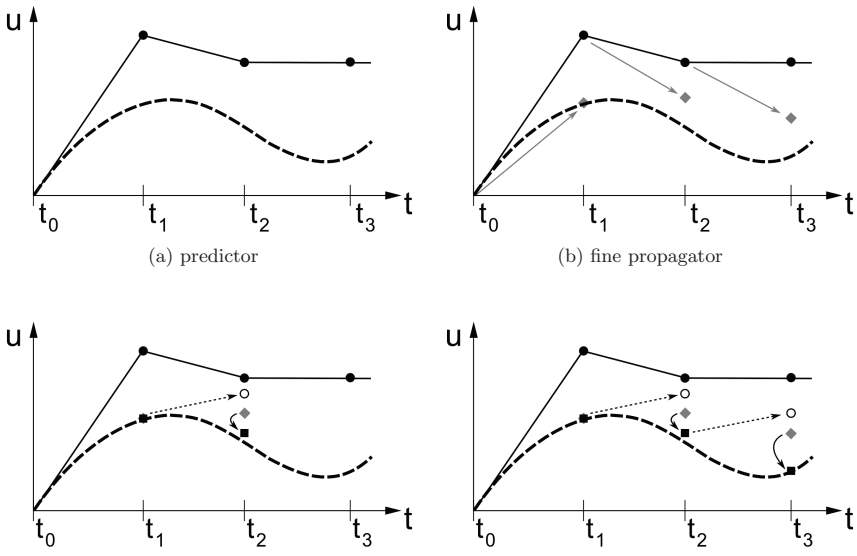
miliar territory and begin our own study of PFASST in Chapter 2. We will present the matrix formulation of PFASST, use this formulation to show equivalence to a multigrid method, and identify the smoother and coarse grid correction of PFASST. This yields the iteration matrix of PFASST, which is presented in Theorem 3. In the second part of Chapter 2, we decompose this iteration matrix with the help of techniques similar to local Fourier analysis. On the basis of this decomposition, we derive the matrix symbols in Theorem 6. Finally, with Definition 4 and Lemma 5, we show an efficient way to employ this decomposition of the iteration matrix. In Chapter 3, we will introduce our two model problems, the diffusion and advection problem. With the mathematical groundwork described in Chapter 2, we study the convergence of PFASST for the two model problems. In particular, we will investigate the role of the initial error vector and the dispersion relation number of the problem. Furthermore, we will investigate the interaction of the different parts of PFASST and we will present 3 setups, which invoke distinguishable modes of operation of PFASST. In Chapter 4, we will introduce a classical and a new strategy to parallelize PFASST. For both parallelization strategies, we will outline the theoretical limits of the speedup. Finally, with the use of the 3 representative setups, we will estimate the speedup from a wide range of numerical experiments for both model problems. In Chapter 5, we will present many promising ideas that were gathered through the investigation of PFASST and conclude with a summary of the work.

1.1. Parareal

The Parareal method was invented by Lions, Maday, and Turinici and first published in [14]. It quickly became the most studied method in the realm of parallel-in-time integration. Parareal is an effective and simple way to solve initial value problems parallel-in-time, denoted by

$$u_t(t) = f(t, u),\ u(0) = u_0, \tag{1.1}$$

where $u_0 \in \mathbb{R}^d$ and $t \in [0, T]$. To do so, Parareal employs established numerical methods. The first numerical method for these problems is the famous Euler method [21], which dates back to the year 1768. Since then, a solid mathematical foundation has been formed for numerical methods for ordinary differential equations, cf. [22, 23]. The advantage of Parareal is that it builds upon this foundation: it combines two, well-studied methods in an iterative manner, to form a novel, time-parallel method. More precisely, it employs a fine-but-expensive propagator $\mathcal{F}(t_1, t_0, u_0)$ and the coarse-but-cheap propagator $\mathcal{G}(t_1, t_0, u_0)$. A propagator is an operator, which yields either an exact or a numerical solution of the initial value problem at the time point t_1, given an initial value u_0 at the time point t_0. Parareal works in an iterative manner, thus we denote the time points and the approximative values at these time points in the k-th Parareal



(a) predictor (b) fine propagator

(c) correction (d) full

Figure 1.1.: Different phases of the Parareal algorithm. Preceding iteration values are marked with full circles, next iteration with black circles. Intermediate values of the fine and coarse propagator are marked with grey diamonds, and empty circles respectively. Each marker is an approximation of the exact solution, represented by the dashed line.

iteration, as $t_1, t_2, \ldots, t_L$ and $U_1^k, U_2^k, \ldots, U_L^k$. Additionally, we denote the set of values $U_1^0, \ldots, U_L^0$, as starting point of the Parareal iteration.

One way to generate them is to employ the coarse propagator

$$U_{n+1}^0 = \mathcal{G}(t_{n+1}, t_n, U_n^0),$$

beginning with $\mathcal{G}(t_1, t_0, u_0) = U_1^0$. We denote this part of the algorithm as predictor step. Finally, the whole Parareal iteration may be described with the following simple calculation rule:

$$U_{n+1}^{k+1} = \mathcal{G}(t_{n+1}, t_n, U_n^{k+1}) + \mathcal{F}(t_{n+1}, t_n, U_n^k) - \mathcal{G}(t_{n+1}, t_n, U_n^k) \tag{1.2}$$

It is best explained along the lines of Figure 1.1. In Figure 1.1a, the exact solution (dashed line) and the results of the predictor (straight lines and full circles) are depicted. This is the initial situation for the Parareal iteration. The first step involves the parallel computation of the fine propagator; the results are depicted in 1.1b in form of grey diamond markers. Next, these values are corrected by $\mathcal{G}(t_{n+1}, t_n, U_n^{k+1}) - \mathcal{G}(t_{n+1}, t_n, U_n^k)$, which has to be done in serial. In Figure 1.1c, this correction is sketched: The value U_1^{k+1} is used to generate an estimation $\mathcal{G}(t_2, t_1, U_1^{k+1})$ (empty circle), which is then used to compute the correction (distance between full and empty circle). Consequently, this value corrects the fine propagator value $\mathcal{F}(t_2, t_1, U_1^k)$ (grey diamond) and therefore produces U_2^{k+1} (black square). This is repeated for all time points. Note that this correction is absent at the time point t_1 because it holds $U_0^k = u_0$ for all iterations. Therefore, it holds $U_1^{k+1} = \mathcal{F}(t_1, t_0, U_0^k)$.

After examining the basic mechanisms of Parareal, we will now present the basic properties of Parareal in the following section.

1.1.1. Convergence, stability and parallel efficiency

Since Parareal primarily consists of the propagators, the convergence behavior is dependent on the convergence properties of the propagators. To give an insight into the structure of convergence proofs, we study Parareal for the test equation

$$u_t(t) = -au(t),\ u(0) = u_0, \tag{1.3}$$

with the explicit Euler method as the coarse propagator $\mathcal{G}(t_1, t_2, u_1)$. For the fine propagator $\mathcal{F}(t_1, t_2, u_1)$, we use the exact solution of

$$u_t(t) = -au(t),\ u(t_1) = u_1, \tag{1.4}$$

evaluated at the time point t_2. Inspired by the results in [14], where an implicit Euler scheme is used as the coarse propagator, we prove the following proposition for our

particular setup.

Theorem 1 (Lions, Maday and Turinici 2001)**.** *The Parareal scheme is of order k, i.e. there exists c_k so that*

$$\left|U_n^k - u\left(t_n\right)\right| + \max_{t\in[t_n,t_{n+1}]} \left|\mathcal{F}(t_n, t, U_n^k) - u\left(t\right)\right| \leq c_k \Delta t^k.$$

Proof. See Section A in the appendix. □

Note that this proof may be considered as a blueprint for proofs of similar convergence results found in literature. To produce more general results, the proof needs to be expanded, which is quite an exercise. For example, it is possible to substitute the term $(1 - a\Delta t)$ with the term $\frac{1}{1+a\Delta t}$. This represents the usage of an implicit instead of an explicit Euler scheme as the coarse propagator. From there on it is easy to try other time-stepping schemes. As long as the stability function R of the coarse propagator is known, it may be employed to represent the coarse propagator in the form of $R\left(-a\Delta t\right)$. This exercise gives valuable insights in the working mechanisms of Parareal. For example, the proof demonstrates the recurring nature of Parareal, as well as the roles of the fine and coarse propagator. The proof also provides ideas of how to produce more general results. In [24], Gander et. al. presented such a generalized convergence result for the Parareal algorithm, which holds for autonomous non-linear ODEs. In the respective proof, the recurrence of errors and the difference between fine and coarse propagator are employed on a more abstract level than in the proof of Theorem 1. We state the result without proof.

Theorem 2 (Gander et. al. 2008)**.** *Let $\mathcal{F}\left(t_{n+1}, t_n, U_n^k\right)$ denote the exact solution at t_{n+1} and $\mathcal{G}\left(t_{n+1}, t_n, U_n^k\right)$ be a one step method with local truncation error bounded by $C_1\Delta t^{p+1}$. If*

$$\left|\mathcal{G}\left(t + \Delta t, t, x\right) - \mathcal{G}\left(t + \Delta t, t, y\right)\right| \leq \left(1 + C_2\Delta t\right)\left|x - y\right|,$$

then the following error estimate holds:

$$\max_{1\leq n\leq L} \left|u(t_n) - U_n^k\right| \leq \frac{C_1\Delta t^{k(p+1)}}{k!}\left(1 + C_2\Delta t\right)^{N-1-k} \prod_{j=1}^{k}(N-j) \max_{1\leq n\leq L} \left|u(t_n) - U_n^0\right| \tag{1.5}$$

$$\leq \frac{\alpha^k}{k!} \prod_{j=1}^{k}(N-j) \max_{1\leq n\leq L} \left|u(t_n) - U_n^0\right| \tag{1.6}$$

$$\leq \frac{(C_1T)^k}{k!} e^{C_2(T-(k+1)\Delta t)} \Delta t^{pk} \max_{1\leq n\leq L} \left|u(t_n) - U_n^0\right| \tag{1.7}$$

This convergence result reflects properties which were observed in earlier works. For example, the increase in order with increasing iteration number is found in form of Δt^{pk} in (1.7). This concurs with the simpler result in Th. 1 and additionally shows how the order of the coarse propagator affects the order of Parareal. Another example is the exactness property: when the number of iterations reaches the number of time points L, then the Parareal algorithm yields the solution of the fine propagator. Under the requirements of Th. 2, Parareal yields the exact solution after a sufficient number of steps, which is easily seen in (1.5). There, the term $\prod_{j=1}^{k}(N-j)$ becomes 0. The last interesting term is $k!$: it produces a faster than linear contraction, since it grows faster than the k-th power of C_1T, while the remaining terms are not growing with more iterations. This shows the superlinear convergence of Parareal. Beyond that, the estimated bounds are tested on two model problems, the diffusion and the advection problem. The convergence behavior of both problems may be explained along the lines of inequality in (1.6). Depending on the propagators it is possible to bring the value α down to 0.06 for the diffusion problem, whereas for the advection problem the lowest value stated is 1.22. For diffusion, this leads to a bound which is monotonically decreasing with more iterations For advection, the same curve first rises, reaches the highest point, and then declines until the exactness property takes effect. Clearly, those are just theoretical estimations. Nonetheless, the numerical experiments show that Parareal has difficulties handling the advection problem. This is further investigated in [25] by studying the characteristics of the advection problem.

The diffusion and advection problems are representatives of the classes of parabolic and hyperbolic partial differential equations. Together, both classes cover many applications in different fields of science. Generally, parallel-in-time algorithms have difficulties in yielding speedups for hyperbolic problems. To ensure comparability to PFASST in Chapter 3, we will also employ the diffusion and advection problem as model problems for our numerical experiments.

However, the convergence result is just one part of a complete analysis. The other part is the stability of the time-stepping method. The stability of Parareal was studied for various requirements and problems, cf. [26, 27]. In this work, we will briefly sketch the basic results for the linear case following [27]. Starting with the stability functions R_i and R_e of the implicit and explicit Euler methods respectively, given by

$$U_{n+1} = U_n - a\Delta t U_n = (1 - a\Delta t)^{n+1} u_0 = R_e(-a\Delta t)^{n+1}$$
$$U_{n+1} = U_n - a\Delta t U_{n+1} = (1 + a\Delta t)^{-n-1} u_0 = R_i(-a\Delta t)^{n+1}.$$

In the case of $|R(z)| \leq 1$, the method is denoted as stable. This definition is easily

adapted to linear ODEs of the form

$$\boldsymbol{u}_t(t) = \mathbf{A}\boldsymbol{u}(t),\ \boldsymbol{u}(0) = \boldsymbol{u}_0, \tag{1.8}$$

by using an eigenvalue decomposition

$$\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1},$$

where $\mathbf{D}$ is a diagonal matrix containing the eigenvalues $\{\lambda_1, \ldots, \lambda_N\}$. With this transformation, the exact solution is given by

$$u(t) = e^{t\mathbf{A}}u_0 = \mathbf{V}e^{t\mathbf{D}}\mathbf{V}^{-1}u_0.$$

In the same manner, the stability functions are generated, resulting in

$$R_{\mathrm{i}}(\Delta t) = \mathbf{V}\left(\mathbf{I} - \Delta t\mathbf{D}\right)^{-1}\mathbf{V}^{-1} \quad \text{and} \quad R_{\mathrm{e}}(\Delta t) = \mathbf{V}\left(\mathbf{I} + \Delta t\mathbf{D}\right)\mathbf{V}^{-1}.$$

As long as $|R(\Delta t\lambda_i)| \leq 1$ for all eigenvalues λ_i, the method is stable for the system of ODEs. For this Problem, Staff et al. [27] were able to derive the stability function H of Parareal in the form of

$$U_n^k = \mathbf{V}H\left(n, k, r, R\right)\mathbf{V}^{-1}u_0,$$

with the stability functions r and R of the fine and coarse propagator respectively. By exploiting the Pascal tree structure of the recurrence of U_n^k, it yields

$$H(n, k, r, R) = \sum_{i=0}^{k} \binom{n}{i}(r - R)^i R^{n-1}.$$

The stability of the method is then achieved if

$$\sup_{1\leq n\leq L}\ \sup_{1\leq k\leq L} |H(n, k, r(\Delta t\lambda_i), R(\Delta t\lambda_i))| \leq 1 \quad \forall\lambda_i, i = 1, \ldots, N.$$

For real eigenvalues, this condition is met for all possible L and all number of iterations k as long as

$$\frac{r\Delta t\lambda_i - 1}{2} \leq R(\Delta t\lambda_i) \leq \frac{r\Delta t\lambda_i + 1}{2} \quad \forall\lambda_i, i = 1, \ldots, N.$$

When the fine propagator is almost exact, it holds $r(\Delta t) \ll 1$. Therefore, for real eigenvalues and an exact fine propagator, Parareal is stable if $|R(\Delta t)| \leq \frac{1}{2}$. More abstract stability results are found in [28].

We complete the brief theoretical outline of Parareal with the observation of the

parallel efficiency. For this purpose, we assume that the same time-stepping method is used for the fine and coarse propagator, but with different time steps. For the fine propagator we use the time step δt and for the coarse propagator we use a multiple $s\delta t$, which we denote as Δt. Further, let T_C be the time needed for one application of the coarse propagator. Consequently, the time needed for the application of the fine propagator is $T_F = sT_C$. The whole computing time adds up to $KLT_C + (K-1)sLT_C$, where K is the number of Parareal iterations and L is the number coarse steps needed to reach the time end point t_L. Since the fine and coarse propagator may be arranged in a pipelining manner, the wall time is $(K+1)LT_C + KsT_C = ((K+1)(L+s) - s)T_C$. In order to compute the speedup and therefore the parallel efficiency, a comparison algorithm is needed. The canonical choice is to use only the fine propagator resulting in sT_CL. However, it should be kept in mind that Parareal and the fine propagator are two different methods with different convergence properties. Thus, different numbers of iterations have to be performed to reach the same level of accuracy. Therefore, the comparison between a serial time-stepping method and Parareal is only reasonable when a sufficient number of Parareal iterations is performed. Then, the parallel efficiency reads

$$\frac{sT_CL}{((K+1)(L+s) - s)T_CL} = \frac{1}{(K+1)\left(\frac{L}{s}+1\right) - 1} \leq \frac{1}{K}. \tag{1.9}$$

As the estimation shows in the case $1 \ll s$, the parallel efficiency is ideal when only 1 Parareal iterations are needed. This is also a theoretical limit for the parallel efficiency of standard Parareal. Therefore, for a reasonable parallel performance, a small number of iterations is needed.

In the last years, Parareal was modified and applied to various problems. For some problems Parareal showed reasonable convergence behavior and for some problems it was just the most convenient way to parallelize legacy code. The modifications lead to versions of Parareal, which for example, add adaptivity or lift the theoretical limit of the parallel efficiency.

1.1.2. Applications and modifications

Shortly after the introduction of Parareal, many works showed the application of Parareal for various problems. In [29], Baffico et al. applied Parareal for molecular dynamics problems. They conclude that for the field of multiscale modeling, the introduction of Parareal is very useful. They also note that the symplecticity has to be investigated for such problems and that the previous time step solution should be used, which we count as the first idea for a modification of Parareal. Also in 2002, in [30], Maday and Turinici reinterpreted Parareal as a preconditioning procedure on an algebraic setting and used this reinterpretation for optimal control for PDEs. See also [31–33]. In the following two years, Parareal was applied to fluid-structure applications in [34], to quantum control

problems in [35], and for unsteady Navier-Stokes equations for incompressible flow with Reynolds numbers up to 1000 in [36]. The application to the more general Navier-Stokes equations and its relatives was studied by many different authors, cf. [37–41]. Other interesting problems from various fields were studied in [42–45].

Every field has particular requirements for the time-stepping methods it is using, for example the field of molecular dynamics asks for symplecticity, meaning that the time-stepping method is volume-preserving in the phase space defined by the underlying hamiltonian problem. Take for example a pendulum which is described by hamiltonian equations, then the phase space is spanned by the momentum and the height of the pendulum. In this phase space, the volume-preserving property relates to the preservation of the kinetic and potential energy. In this case, a symplectic time-stepping method would preserve the kinetic and potential energy. Thus, it would indicate a stability over many time steps. Parareal itself is not symplectic, even when the fine and coarse propagator are, because the linear combination of symplectic time-stepping methods does not necessarily result in a symplectic time method. In [46], this is overcome by the introduction of an interpolation operator, which substitutes the correction steps of Parareal. This comes with raised computational costs. Another favorable property for many applications is adaptivity, where the time step size is adjusted during the computation of the problem. In [47], the formulation of Parareal as a multiple-shooting method is used to develop an adaptive version. Additional efforts to develop an improved Parareal error control mechanism were made in [48]. Lastly, for the case when a problem consists of a stiff and non-stiff part, it is convenient to use an IMEX scheme, which treats the stiff part implicitly and the non-stiff part explicitly. Consequently, a version of Parareal, equipped with IMEX-RK schemes as propagators, were studied in [49].

However, to alleviate the theoretical limit of the parallel efficiency, one has to take previous iterations into account. Following this idea, it is useful to interpret Parareal as an iterative scheme and it becomes obvious to use iterative schemes as fine and coarse propagator. Such iterative schemes, as Parareal itself, need an starting value additional to an initial value. Fortunately, Parareal produces the starting values for the propagators in each iteration, since the result of the previous iteration may be employed. This idea has many realizations. One of the first realizations is found in [50]. In this work, spectral deferred corrections (SDC) are used as fine propagators. Spectral deferred corrections are also the basic building block of PFASST. This is no coincidence, given that the algorithms Parareal/SDC described in [50] and the successing algorithm Hybrid Parareal SDC described in [16] are precursor of PFASST.

Without going into the details of SDC, which we will do in Section 1.3.2, we denote it as an iterative method for the discretization of ODE in the form

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(\tau, u(\tau))\mathrm{d}\,\tau,$$

which is known as the Picard formulation. A fine propagator has for example the form

$$\mathcal{F}\left(t_{n+1}, t_n, U_n^k\right) = \bar{U}_n^k + \mathcal{I}\left(t_{n+1}, t_n, U_n^k\right),$$

where $\mathcal{I}\left(t_{n+1}, t_n, U_n^k\right)$ is a quadrature operator, which approximates the integration term of the Picard formulation. Note that this quadrature operator may as well be iterative and that depending on the quadrature used, the role of U_n^k changes. In the usual case of more than one quadrature node, it represents the approximate values on all quadrature nodes in the interval $[t_n, t_{n+1}]$, instead of just the value at the time point t_n. Therefore, we denote $\bar{U}_n^k$ as the value at the time point t_n. With a similar definition for the coarse propagator

$$\mathcal{G}\left(t_{n+1}, t_n, U_n^k\right) = \bar{U}_n^k + \mathcal{Q}\left(t_{n+1}, t_n, U_n^k\right),$$

where $\mathcal{Q}\left(t_{n+1}, t_n, U_n^k\right)$ is also an approximation of the integration term, it yields

$$\bar{U}_{n+1}^{k+1} = \bar{U}_n^{k+1} + \mathcal{Q}\left(t_{n+1}, t_n, U_n^{k+1}\right) - \mathcal{Q}\left(t_{n+1}, t_n, U_n^k\right) + \mathcal{I}\left(t_{n+1}, t_n, U_n^k\right).$$

At first glance, this form does not seem to differ much from the standard Parareal iteration, but the novel idea is hidden behind the notation. The main difference here is that we employ iterative operators $\mathcal{Q}$ and $\mathcal{I}$ and, therefore, have to conserve results of the last iteration. This comes with a significant advantage. Similar to Parareal, SDC increases in order with each iteration until it reaches a spectral order, depending on the choice of quadrature nodes. Instead of having the cost of a high order time-stepping method in each Parareal step, SDC steps are of low order. Thus, these lower costs are spread over several Parareal iterations. Then, the spectral order of SDC should be reached after several Parareal iterations. This is reflected in the new upper limit of the parallel efficiency, which reads

$$\frac{M}{(N+K)\frac{1}{s}+K},$$

where K is the number of Parareal iterations, s the relation between the costs of a fine and coarse propagator, N the number of processors, and finally M the number of fine SDC iterations needed for the serial method, to which the modified Parareal is compared. This new theoretical limit is significantly increased compared to the theoretical limit of the classical Parareal method.

At this point, it is vital to set the right expectations for the analysis of PFASST in this work. Generally, we will not take the time-stepping perspective, as we have done here for Parareal. Thus, we will not be able to present statements such as in Theorems 1 and 2. The statements about convergence for PFASST will be more accurate, but also

only applicable to very particular cases. For the stability in the sense of time-stepping methods, it is necessary to solve the test equation. Since PFASST uses coarsening in space, it is not applicable to the test equation and, thus, we are not able to study stability in the classical sense. The parallel performance of PFASST is accessible in the same way as the parallel performance of Parareal. In Chapter 4, we will investigate the parallel performance in detail.

In this work, the focus lies on multigrid methods, given that PFASST has origins in this field. Consequently, we illuminate these origins by introducing multigrid and presenting several multigrid methods, which were devised as parallel-in-time methods.

1.2. Space-time multigrid

Multigrid methods are a class of efficient algorithms for the approximative solution of systems of linear equations, given in the form

$$\mathbf{A}\boldsymbol{u} = \boldsymbol{f}. \tag{1.10}$$

One of the earliest works dates back to the year 1962, where Fedorenko devised in [51] a prototypical multigrid method to solve an elliptic difference equation. The two main subclasses of multigrid are algebraic multigrid, where only the system of linear equations is given, and geometric multigrid, where additional knowledge about the underlying PDE is incorporated into the method. For both subclasses it is important to have a suitable hierarchy of discretizations and suitable transfer operations between them. For algebraic multigrid, these discretizations are the result of heuristic methods applied to the system matrix. For geometric multigrid, these discretizations stem from the PDE. A reasonably-well designed multigrid method rewards one with computational costs of linear order in the number of unknowns and with a convergence rate independent of the grid size. These properties are quite common for elliptic PDEs, but we do not expect such properties to occur for the time dependent PDEs, which we will encounter later on in Chapter 3.

1.2.1. Multigrid basics

Here, we give a very brief description of multigrid; for a comprehensive introduction to multigrid, we refer to [52]. The main components of multigrid are relaxation, interpolation, restriction, and coarse grid correction. To understand the roles of these components, we need to introduce the terms of high and low oscillation error modes. This differentiation stems from the diagonalization of the system matrices of the common model problems. This diagonalization results in eigenvectors, which are associated with sinusoidal functions with different frequencies. Consequently, the error modes that appear are decomposed into the eigenvectors and thus into high and low oscillating error modes. This definition is closely intertwined with the respective problem. Therefore, this

definition is not valid if the resulting eigenvectors have a different form. Nonetheless, the core idea of dividing the possible errors into two groups translates well for other problems. Usually, the definition for these two groups is adjusted according to the problem. To remain consistent with the terminology of multigrid, we denote these two groups as high and low oscillation modes independent of the definition given by the problem. Note that other designations for these two groups are "smooth" for low oscillation error modes and "non-smooth" for high oscillation error modes. In addition, we denote the different discretizations in the hierarchy as "levels". The key mechanism on which the superior convergence behavior of multigrid is built, is the change in frequency of the error modes when transferred to another level. On each level, we have two groups of error modes, when, e.g., a low oscillation error mode is transferred to a coarser level it may join both groups on the coarser level Depending on the coarser grid, it is possible that almost half of the low oscillation modes are reinterpreted as high oscillation modes on the coarser level. Vice versa, high oscillation modes might be reinterpreted as low oscillation modes. It is important to keep this in mind, when the roles of the components of multigrid are discussed.

The first component is relaxation; it is an iterative operation that efficiently reduces the high oscillation error modes on the respective level, but is usually inefficient in reducing the low oscillation error modes. Common examples for relaxation are iterative solvers of the form

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \mathbf{P}^{-1}\left(\boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k\right), \tag{1.11}$$

with the preconditioning matrix $\mathbf{P}$, the right hand side $\boldsymbol{U}_0$ and the system matrix $\mathbf{A}$. The preconditioning matrix defines the nature of the iterative solver. In the case of the damped iterative Jacobi method, $\mathbf{P}$ equals the diagonal of $\mathbf{A}$ scaled with a damping factor ω and in the case of the iterative Gauß-Seidel method $\mathbf{P}$ consists of the lower triangular part of $\mathbf{A}$. In both cases, it easy to compute the inverse of $\mathbf{P}$, which is one requirement for a reasonably-well designed preconditioning matrix. The other requirement is that it in some way inherits the characteristics of the system matrix $\mathbf{A}$. For more details on iterative solvers we recommend [53]. Finally, the relaxation reduces the errors of one group and leaves the second group nearly unchanged.

The next step involves the transfer of the residual $\boldsymbol{r}^k = \boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k$ to the next coarser level, which is performed by the restriction operator $\mathbf{T}_F^C$. The simplest choice for a restriction operator is the straight forward injection of the values of the fine grid onto the coarse grid. From an algebraic point of view, it is favorable to use the transpose of the interpolation operator as restriction operator. This algebraic point of view is presented in detail in Chapter 5 of [54]. For linear interpolation this results in the full weighted restriction. This restriction averages the neighboring values to compute the

value on the coarse grid. Now, on the coarser level we treat the residual equation

$$\tilde{\mathbf{A}}\tilde{\boldsymbol{e}} = \mathbf{T}_F^C \boldsymbol{r}^k$$

with the relaxation to reduce the respective high oscillation error modes of the coarse level. Subsequently, the residual is transferred to the next coarser level. This procedure is repeated until the coarsest level is reached, on which it becomes feasible to solve the problem directly or to use a sufficiently large number of iterations to get an almost exact solution. The solution from the coarsest level is interpolated to the next finer level and used to correct the result of this level. This correction is often accompanied by an additional relaxation. And again this procedure is repeated until the finest level is reached. On the finest level, this correction has the form $\boldsymbol{u}^k + \mathbf{T}_C^T \tilde{\boldsymbol{e}}^k$. We denote this process as coarse grid correction. Because of its visual form, this combination of transfer and relaxation operators is called V-cycle. On each level, the algorithm can either pass the result to the next finer level or get a correction from a coarser level. Therefore, other forms like a W-cycle are possible and reasonable, depending on the problem. When a suitable initial guess is needed, it is advisable to use the full multigrid cycle [55]. This cycle starts on the coarsest level and interpolates the solution to the finest level, while using additional V-cycles on each level to correct the coarse solution before reaching the next finer level.

Note at this point that in the linear case, each coarse level receives the residual of the preceding level in order to compute an error, which will subsequently correct the approximate solution of the preceding level. The basis for the coarse grid correction is the equation

$$\mathbf{A}\boldsymbol{e}^k = \mathbf{A}\left(\boldsymbol{u} - \boldsymbol{u}^k\right) = \boldsymbol{f} - \mathbf{A}\boldsymbol{u}^k = \boldsymbol{r}^k.$$

In the case of non-linear problems, this residual equation does not hold any more and we have to fall back to strategies, such as the full approximation scheme, which was first introduced in [56] and is a key element of PFASST. The full approximation scheme is the non-linear version of the coarse grid correction. Instead of treating the residual equation with relaxation, the non-linear equation with a corrected right-hand side

$$\mathbf{A}\left(\boldsymbol{u}^{k+1}\right) = \boldsymbol{f} + \boldsymbol{\tau}^k$$

is treated on the coarser level, where

$$\tau^k = \tilde{\mathbf{A}}\left(\mathbf{T}_F^C \boldsymbol{u}^k\right) - \mathbf{T}_F^C \mathbf{A}\left(\boldsymbol{u}^k\right)$$

is denoted as $\boldsymbol{\tau}$-correction. With the updated value $\tilde{\boldsymbol{u}}^{k+1}$ from the coarser level, we

compute the $\boldsymbol{\delta}$-correction

$$\boldsymbol{\delta}^k = \mathbf{T}_C^F \left(\tilde{\boldsymbol{u}}^{k+1} - \mathbf{T}_F^C \boldsymbol{u}^k \right).$$

With this correction we update the value on the finer level

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \boldsymbol{\delta}^k.$$

In the linear case, these three steps are equivalent to the coarse grid correction. Another variation of multigrid for non-linear problems is the Newton multigrid method; see [57] for a detailed comparison of the Newton multigrid and the full approximation scheme.

Equipped with these basic multigrid ideas, we return to the field of parallel-in-time methods. More precisely, we briefly review space-time multigrid methods, which are capable to perform parallel-in-time integration or offer an interesting mathematical foundation.

1.2.2. Parabolic multigrid

When we apply these multigrid ideas to discretizations in space and time, we denote it as space-time multigrid. First attempts were made by Hackbusch in 1984 with parabolic multigrid [58]. For a parabolic problem

$$u_t + L(t)u = f(t),$$

where $L(t)$ is a time dependent elliptic operator, an implicit Euler discretization in time direction is used to define the parabolic operator

$$P(u,t) = \frac{u(t) - u(t - \Delta t_l)}{\Delta t_l} + L(t)u(t) = f(t),$$

for which Hackbusch used a multigrid method in each time step. It is stated that with coarsening in the spatial dimension only, the standard multigrid convergence speed is achieved. In this work a first intuitive way of understanding what happens with coarsening in the temporal dimension is given. Hackbusch argues that since his smoother only works on non-smooth modes in space, a non-smooth mode in time does not vanish from relaxation. These are then handled on the coarser level by the exact solve and the coarse grid correction. Therefore, temporal coarsening could in fact generate high frequency error modes on the coarser level which are then interpolated back and not affected by the relaxation. He suggests the use of coarsening in time as a preconditioning strategy in the case of error modes, which are initially smooth in the temporal direction.

The second intuitive argument given describes what happens to error modes when the relaxation is done parallel-in-time. This is done by using the values from the last

iteration for the computation of the next time step, which is usually sequential. For this case, he assumes that the error is an eigenvector of the elliptic operator L_h with the associated eigenvalue λ. Consequently, it yields the error reduction for the parabolic multigrid

$$\alpha = \frac{1-\rho}{1+\lambda\Delta t_l}.$$

It is pointed out for the Poisson problem that α is of the order $\mathcal{O}(\Delta x_l^2/\Delta t_l)$, meaning that the convergence behavior improves with the stiffness of the problem.

1.2.3. Waveform relaxation

The paper on waveform relaxation [59], written by Lubich and Ostermann in 1987, provides a mathematically more rigorous approach to this topic. There, the parabolic equation is transformed by the Laplace transformation in time

$$F(s) = \int_0^\infty e^{-st} f(t)\,\mathrm{d}t$$

into an elliptic equation

$$s\hat{u} + L_h\hat{u} = \hat{f}.$$

Subsequently, a splitting method like Gauß-Seidel is applied formally. This means that the system matrix

$$\mathbf{A}(s) = sI + L_h$$

is solved by an iterative method according to (1.11) with a preconditioning matrix $\mathbf{P}(s)$, which consists of the lower triangular part of $\mathbf{A}(s)$. When transformed back to the real time, this yields

$$\frac{du^{k+1}}{dt} + (D-C)\,u^{k+1} - Bu^{(k)} = f, \quad u^{k+1}(0) = u_0,\, t \in [0, \Delta T]\,,$$

with functions u^k that are generated at each iteration and the decomposition of $L_h = D - C - B$ into the diagonal, the strictly upper, and lower triangular parts. This transformation is also applied to the usual multigrid two-grid operator, which yields a set of sequential initial value problems. For this scenario an abstract two-level operator $\mathcal{M}$ is introduced and analysed. This leads to the convergence result, given as an estimate of the spectral radius

$$\rho(\mathcal{M}) \leq \frac{1}{2}\sqrt{\mu_0(2\nu-1)}, \quad \text{for } \nu = \nu_1 + \nu_2 \geq 1 \text{ and } \mu_0(\nu) = \frac{\nu^\nu}{(\nu+1)^{\nu+1}},$$

where ν_1 and ν_2 are the number of pre and post-relaxation steps. The proof relies on

$$\rho(\mathcal{M}) = \max_{\mathrm{Re}(s)\geq 0} \rho\left(M(s)\right),$$

which is true, as long as $\mathcal{M} \in L^p\left(\mathbb{R}_+, \mathbb{C}^n\right)$. Except for the additional parameter s, the two-grid operator $M(s)$ in the Laplace space is analysed with the same means as standard multigrid, e.g., the decomposition of the error into modes.

Hackbusch's parabolic multigrid is covered by this theory, when the integration operator $\mathcal{M}$ is exchanged by the respective discrete operator $\mathcal{M}_{\Delta t}$, representing a time-stepping method like the implicit Euler or more involved methods like Runge-Kutta. For algebraically stable Runge-Kutte methods it is then shown that

$$\rho\left(\mathcal{M}_{\Delta t}\right) \leq \rho\left(\mathcal{M}\right)$$

holds. Therefore, for adequate time-stepping methods, multigrid convergence is achieved. Note that this work does not include the parallel-in-time relaxation nor the coarsening in time as proposed by Hackbusch.

1.2.4. Space-time multigrid by Horten and Vandewalle

In [60], Horten and Vandewalle proposed another kind of multigrid method, which is based on space-time stencils and checkered Gauß-Seidel smoothing. Using local Fourier analysis, Horton and Vandewalle estimated the error reduction for coarsening in the spatial, the temporal, and both directions simultaneously. The error reduction also depends on the dispersion relation number, e.g., $\Delta t/\Delta x^2$ for the diffusion problem. For this problem, it is then shown that coarsening in time works well when backward differentiation of first or second order is used for a high dispersion relation number. On the other hand, it is shown that coarsening in space works well for low dispersion relation numbers. The break even point of both strategies is found at a point of high error reduction. When the appropriate coarsening strategy is chosen on each level according to the dispersion relation number, the worst error reduction is found at this break even point. The parallel efficiency for this method is determined by the number of colors used for relaxation, which is rather limited.

1.2.5. Multigrid reduction in time

The "multigrid reduction in time" method [61], in short MGRIT, is applicable to a wide range of problems and exhibits a high potential for parallelization. It is devised for initial value problems and only requires a time-stepping method $\Phi(t_i, t_{i+1}, u_i)$. This time-stepping method is used for the fine time steps in the form of the linear operator Φ_δ and for the coarse time steps in the form of the operator Φ_Δ. In the linear case, this

leads to a system

$$\mathbf{A}\boldsymbol{u} = \begin{bmatrix} I & & & \\ -\Phi_\delta & I & & \\ & \ddots & \ddots & \\ & & -\Phi_\delta & I \end{bmatrix} \boldsymbol{u} = \boldsymbol{f}.$$

This system is then solved in a multigrid fashion that could be interpreted as a multilevel enhancement of Parareal. For that purpose, the time domain is discretized into N_t time points. Every m-th point is now marked as a C-point; every point in-between the C-points are marked as F-points. Associated with these points, F and C relaxation is defined as the application of the respective time-stepping methods. More precisely, F relaxation starts from each preceding C-point and performs the time-stepping method Φ_δ on all F-points until the next C-point is reached. In contrast, C relaxation takes as many preceding F-points as needed to compute the next C-point with Φ_Δ. Both relaxations may be computed on N_t/m computing units, where N_t is the number of fine steps. Restriction is done by injection on the C-points, interpolation is an injection back followed by a F-relaxation. MGRIT is a robust and flexible method, devised for HPC systems with many cores to work as a black box parallelization tool for linear and non-linear problems in the temporal dimension. In the last years, MGRIT was actively developed and efforts were made to provide theoretical results. The most recent work on this matter is [62]. There, we find the two-grid relaxation propagation

$$E_\Delta = P\left(I - B_\Delta^{-1} A_\Delta\right)\left(I - A_\Delta\right) R = P E_\Delta^{FCF} R,$$

which consists of the error propagation of the FCF relaxation enlaced by special interpolation and restriction operators P and R, and of the operators

$$A_\Delta = \begin{bmatrix} I & & & \\ -\Phi_\delta^m & I & & \\ & \ddots & \ddots & \\ & & -\Phi_\delta^m & I \end{bmatrix} \quad \text{and} \quad B_\Delta = \begin{bmatrix} I & & & \\ -\Phi_\Delta^m & I & & \\ & \ddots & \ddots & \\ & & -\Phi_\Delta^m & I \end{bmatrix}.$$

Under the assumption that the time-stepping operators Φ_δ and Φ_Δ are diagonalized by the same unitary transformation with eigenvalues λ_ω and μ_ω, the authors show the following estimation:

$$\left\| E_\Delta^{FCF} \boldsymbol{e} \right\|_2 \leq \max_\omega \left\{ |\lambda_\omega^m - \mu_\omega| \frac{1 - |\mu_\omega|^{N_t/m-1}}{1 - |\mu_\omega|} |\lambda_\omega|^m \right\} \|\boldsymbol{e}\|_2 .$$

By means of this estimation, we recognize the convergence improving tendencies, such as the proximity between the coarse and fine time-stepping method ($|\lambda_\omega^m - \mu_\omega|$), the improved accuracy of the fine time-stepping method, and the number of time steps ($|\lambda_\omega|^m$).

1.2.6. Space-time multigrid by Neumüller and Gander

The method in [63] is more in alignment with the works of Hackbusch, Horten and Vandewalle. In addition, this method has more capability of parallelism in space and time and an equally well established theoretical foundation. The authors employ discontinuous Galerkin formulations in space and time for the diffusion and Stokes equation, which leads to the linear system

$$\begin{bmatrix} A_{\tau,h} & & & \\ B_{\tau,h} & A_{\tau,h} & & \\ & \ddots & \ddots & \\ & & B_{\tau,h} & A_{\tau,h} \end{bmatrix} u = f,$$

where

$$\begin{aligned} A_{\tau,h} &= K_\tau \otimes M_h + M_\tau \otimes K_h \quad \text{and} \\ B_{\tau,h} &= -N_\tau \otimes M_h \end{aligned}$$

are compositions of the stiffness and mass matrix in time (K_τ, M_τ) and space (K_h, M_h), the operator N_τ determines the initial value for the next time step. Based on this, the design of the method is straightforward. The relaxation is given in the form of stationary iterative solver, see (1.11), with $P = \omega \cdot \text{diag}\left(\tilde{A}_{\tau,h}, \ldots, \tilde{A}_{\tau,h}\right)$, where $\tilde{A}_{\tau,h}$ is a simple-to-compute approximation of $A_{\tau,h}$. We denote this as weighted block Jacobi relaxation. The usual linear interpolation and restriction operators are employed, both in space and time combined together depending on the coarsening strategy. Like in [60], the coarsening strategy depends on the dispersion relation number of the problem, e.g., $\Delta t/\Delta x^2$ for the heat equation. A detailed local Fourier analysis yields the appropriate coarsening strategy and the optimal weighting parameters for relaxation. The numerical experiments are performed on state-of-the-art HPC systems with almost optimal strong and weak scaling results.

In summary, there are many possibilities to design successful parallel-in-time multigrid methods and it is feasible to apply the theoretical framework of multigrid for the analysis of such methods. For the space-time multigrid by Neumüller and Gander the Fourier analysis is a key element, since it determines the coarsening strategy that is needed for the excellent convergence behavior of the method. For PFASST, such a clear categorization into the class of multigrid methods has not been provided so far. Therefore, we will

now introduce PFASST in the classical formulation, but in a notation suited to study PFASST from a multigrid perspective.

1.3. The parallel full approximation scheme in space and time

After a brief overview of Parareal, the method which inspired PFASST, and multigrid, the theory which inspired this work, we will now focus on PFASST itself. The fundamental requirement of all results in this work is the assumption that the underlying problem stems form linear autonomous ordinary differential equations. This makes it possible to reformulate PFASST as a complex stationary iterative solver and to use results from linear algebra for the analysis.

However, in contrast to the various notations, which were matched to the literature behind each method, we strive for a harmonized notation for the remainder of this work. We dedicate the following section to this purpose. Note that the following sections coincide with a paper, which was already submitted in [64].

1.3.1. Preliminaries and notation

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

$$U(t) = U_0 + \int_{t_0}^{t} \mathbf{A}\, U(\tau)\mathrm{d}\tau, \quad t \in [t_0, T]\,, \tag{1.12}$$

where $\mathbf{A}$ is a discretized spatial operator, e.g., 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 $[t_{l-1}, t_l]$, with $l \in \{1, \ldots, L\}$, contains a set of M nodes $\{\tau_1, \ldots, \tau_M\}$. We choose

$$\begin{gathered} 0 = t_0 < \ldots < t_L = T, \quad t_l \leq \tau_1 < \ldots < \tau_M = t_{l+1}, \\ \Delta t = t_{l+1} - t_l, \quad \Delta\tau_m = \tau_{m+1} - \tau_m. \end{gathered} \tag{1.13}$$

Each set of nodes $\{\tau_1, \ldots, \tau_M\}$ is used as quadrature nodes for the numerical integration with rules such as 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 $[t_l, t_{l+1}]$, we denote it e.g., with $\boldsymbol{U}_{[t_l, t_{l+1}]}$ (if it is not clear from the context).

Due to the nested structure and the distinct treatment of spatial and temporal dimen-

sions, an appropriate notation is needed. Continuous functions are always represented by lower case letters, discretized and semi-discretized functions are represented by upper case letters. 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 $x_1,...,x_N$, we use the notation

$$U(t)=(u(t,x_1),u(t,x_2),\dots,u(t,x_N))^T\in\mathbb{R}^N,\quad t\in[t_0,T]$$

for semi-discretization in space. A full space-time discretization is denoted as

$$\begin{aligned}\boldsymbol{U}_{[t_{l-1},t_l]}&=(U(\tau_1),U(\tau_2),\dots,U(\tau_M))^T\in\mathbb{R}^{M\cdot N},\quad \tau_i\in[t_{l-1},t_l],\ l\in\{1,\dots,L\},\\ \boldsymbol{U}&=\left(\boldsymbol{U}_{[t_0,t_1]},\dots,\boldsymbol{U}_{[t_{L-1},T]},\right)^T\in\mathbb{R}^{M\cdot N\cdot L}.\end{aligned}$$

On each subinterval, a *collocation problem* arises when quadrature is used as a numerical counterpart to the integration in (1.12). The basis for most quadrature formulations is the interpolation, which is easily expressed using the Lagrange polynomial basis $\{\ell_i\}_{i=1}^M$ with

$$\ell_i(s):=\prod_{k=1,k\neq i}^{M}\frac{s-\tau_k}{\tau_i-\tau_k}. \tag{1.14}$$

If we weight each Lagrange polynomial with the evaluation of the function $f(t)$ at the point τ_i and sum them up, we get the interpolation polynomial of the function $f(t)$, which is exact on the nodes $\{\tau_1,\dots,\tau_M\}$. Note that the formulation in (1.14) is not the preferable way to evaluate the Lagrange polynomials, since it becomes numerically unstable for a large number of nodes. A remedy for this problem is the barycentric interpolation, cf. [65].

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 $a\le\tau_1<\tau_2<\ldots<\tau_M=b$ be the set of quadrature nodes on a subinterval $[a,b]$ with $\Delta t=b-a$ and $\mathbf{Q}$ the quadrature matrix with entries*

$$q_{i,j}=\frac{1}{\Delta t}\int_a^{\tau_j}\ell_i(\tau)\,\mathrm{d}\tau,\quad i,j=1,...,M.$$

We discretize (1.12) *at the quadrature nodes, using the matrix $\mathbf{Q}$ as an approximation of the integral and obtain this set of linear equations:*

$$U(\tau_i)=U(t_0)+\Delta t\sum_{j=1}^{M}q_{i,j}\mathbf{A}\,U\left(\tau_i\right),\quad i=1,...,M.$$

Using the Kronecker product and the vector of ones $\mathbf{1}_M \in \mathbb{R}^M$ *we write this system of linear equations as*

$$\boldsymbol{U} = \boldsymbol{U}_0 + \Delta t \, (\mathbf{Q} \otimes \mathbf{A}) \, \boldsymbol{U}, \quad \textit{with } \boldsymbol{U}_0 = \mathbf{1}_M \otimes U(t_0),$$

or, more compactly,

$$\mathbf{M}\boldsymbol{U} = (\mathbf{I} - \Delta t \mathbf{Q} \otimes \mathbf{A}) \, \boldsymbol{U} = \boldsymbol{U}_0. \tag{1.15}$$

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 (1.13). Due to the higher order, we will employ the Gauß-Radau quadrature rule in this work.

Finally, the PFASST algorithm works 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 solves 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., $\tilde{\mathbf{A}}$ is denoted as the coarse level version of $\mathbf{A}$.

1.3.2. Spectral deferred corrections

Instead of directly solving the collocation problem on a subinterval, the spectral deferred corrections method (SDC) utilizes a low-order method to generate an iterative solution that converges to the collocation solution $\boldsymbol{U}$. SDC was first introduced by Dutt et al. [66] as an improvement of deferred correction methods [67]. In the last decade, SDC has been accelerated with GMRES or other Krylov subspace methods [68], enhanced to a high-order splitting method [69–71], and found its way into the domain of time-parallel computing [50, 72], in particular within PFASST [16, 17].

Regarding the setting of this work, 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 et al. [73], this interpretation was used to optimize the convergence speed of SDC.

Like in (1.11), a general preconditioned stationary iterative method, here once again denoted in our notation as

$$\boldsymbol{U}^{k+1} = \boldsymbol{U}^k + \mathbf{P}^{-1}(\boldsymbol{c} - \mathbf{M}\boldsymbol{U}^k), \tag{1.16}$$

is fully described by the preconditioner $\mathbf{P}$, the system matrix $\mathbf{M}$, and the right-hand side

$\boldsymbol{c}$ of the linear equation under consideration. $\mathbf{P}$ has to be easy to invert, while being an accurate alternative for the system matrix $\mathbf{M}$. The SDC method follows this approach by replacing the full quadrature matrix $\mathbf{Q}$ with a lower triangular matrix $\mathbf{Q}_\Delta$. A simple way to generate a lower triangular matrix is to use the rectangle rule for quadrature instead of the Gauß-Radau rule. In [73], a LU decomposition of $\mathbf{Q}$ provides a $\mathbf{Q}_\Delta$ which results in better convergence properties than the use of the simple rectangle rule, while requiring the same computational effort.

The particular choice

$$\mathbf{P}_{\text{SDC}} = \mathbf{I} - \Delta t \mathbf{Q}_\Delta \otimes \mathbf{A} \quad \text{and} \quad \boldsymbol{c} = \left(U(t_0), U(t_0), \ldots, U(t_0)\right)^T \in \mathbb{R}^{NM}, \tag{1.17}$$

then allows us to write SDC as an iterative method for the system matrix $\mathbf{M}$ as defined in Def. 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 other than the first, the right-hand side consists of a numerical approximation of $U(t_l)$ spread on each node. In order to start the iteration, an initial iteration vector $\boldsymbol{U}^0$ 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 [73, 74]. The iteration matrix of SDC is simply given by

$$\begin{aligned} \mathbf{T}_{\text{SDC}} &= \mathbf{I} - \mathbf{P}_{\text{SDC}}^{-1}\mathbf{M} \\ &= \mathbf{I} - \left(\mathbf{I} - \Delta t \mathbf{Q}_\Delta \otimes \mathbf{A}\right)^{-1} \left(\mathbf{I} - \Delta t \mathbf{Q} \otimes \mathbf{A}\right). \end{aligned} \tag{1.18}$$

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

Note further that the same procedure may be written using matrices $\mathbf{S}$, which is dense, and $\mathbf{S}_\Delta$, which is a lower triangular matrix with a similar role as $\mathbf{Q}_\Delta$. These matrices have the effect of a node-to-node quadrature rule in each row. It is possible to reformulate (1.15) and the associated SDC sweep associated with (1.17). Therefore, we define the required constituents as

$$\mathbf{M}_S = \mathbf{I} - \Delta t \mathbf{S} \otimes \mathbf{A} - \mathbf{E} \otimes \mathbf{I}_N \tag{1.19}$$

$$\mathbf{P}_S = \mathbf{I}_M \otimes \mathbf{I}_N - \Delta t \mathbf{S}_\Delta \otimes \mathbf{A} - \mathbf{E} \otimes \mathbf{I}_N \tag{1.20}$$

$$\boldsymbol{c}_S = \left(U_0, 0, \ldots, 0\right)^T \in \mathbb{R}^{NM}. \tag{1.21}$$

The multiplication with the matrix

$$\left[\mathbf{L}\right]_{i,j} = \begin{cases} 1 & \text{if } i \leq j, \\ 0 & \text{else}, \end{cases} \tag{1.22}$$

which forms the accumulated sum over the components from the left, yields the equivalence to the $\mathbf{Q}$ matrix formulation of the SDC sweep. Yet another equivalence appears by multiplying the preconditioning matrix $\mathbf{P}_S$ from the left to both sides of the stationary iteration equation and comparing it component-wise to the node-to-node formulation

$$\begin{aligned} U_{m+1}^{k+1} = U_m^{k+1} &+ \Delta\tau_m \left[f(U_{m+1}^{k+1}, \tau_{m+1}) - f(U_{m+1}^{k}, \tau_{m+1}) \right] \\ &+ \Delta t \left(\mathbf{S}_m^{m+1} \mathbf{F}^k \right)_m , \end{aligned} \tag{1.23}$$

which is the common formulation of SDC found in the literature, for example in [75]. In this formulation it is also more convenient to define an IMEX or multi-rate version of SDC, cf. [76]. More important than the formulation, are the stability and convergence properties of SDC, for which we refer to [66, 73, 76, 77]. Note that it is difficult to transfer these results to PFASST, since additional correction steps from coarser levels are performed. In addition, many results have strong requirements, such as the Lipschitz continuity and differentiability of the right hand side $f(u)$. Most efforts to transfer these results would keep PFASST in the realm of time-stepping methods. Whereas in this work, efforts are made to move PFASST to the realm of multigrid methods. Consequently, a multi-level version of SDC is introduced in the following chapter.

1.3.3. Multi-level spectral deferred corrections

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 in [78]. 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, e.g., 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 $\mathbf{T}_C^F$ and $\mathbf{T}_F^C$, and a coarse-grid correction in order to transfer information between the different levels. As a consequence, MLSDC can be written as a FAS-multigrid-like iteration. Like SDC, it solves the collocation problem in an iterative manner, using the same initial iteration vector. For our purposes, we derive a two-level version of one MLSDC step $\boldsymbol{U}^k \to \boldsymbol{U}^{k+1}$ from [78] as:

1. Perform n_F fine SDC sweep using the values $\boldsymbol{U}^k$ according to (1.16). This yields provisional values $\boldsymbol{U}^*$.
2. Sweep from fine to coarse:
   a) Restrict the fine values $\boldsymbol{U}^*$ to the coarse values $\tilde{\boldsymbol{U}}^k$.
   b) Compute the FAS correction $\boldsymbol{\tau}^k = \tilde{\mathbf{M}}\tilde{\boldsymbol{U}}^k - \mathbf{T}_C^F \mathbf{M} \boldsymbol{U}^*$

c) Perform n_C coarse SDC sweeps on the collocation problem

$$\left(\mathbf{I} - \Delta t \tilde{\mathbf{Q}} \otimes \tilde{\mathbf{A}}\right) \tilde{\boldsymbol{U}} = \tilde{\boldsymbol{U}}_{\mathbf{0}} + \boldsymbol{\tau}^k,$$

with $\tilde{\boldsymbol{U}}^k$ as starting value. This yields new values $\tilde{\boldsymbol{U}}^{k+1}$.

3. Sweep from coarse to fine :
Compute the interpolated coarse correction $\boldsymbol{\delta}^k = \mathbf{T}_C^F \left(\tilde{\boldsymbol{U}}^{k+1} - \mathbf{T}_F^C \boldsymbol{U}^*\right)$ and add it to $\boldsymbol{U}^*$ to obtain $\boldsymbol{U}^{k+1}$.

Note that we use the FAS correction strategy here to match the description of [78]. Since in the linear case using this correction strategy is equivalent to the standard coarse-grid correction [52], this is just a question of notation. 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, like for SDC, we can cast this algorithm as a preconditioned stationary iterative method.

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

$$\begin{aligned} \boldsymbol{U}^{k+\frac{1}{2}} &= \boldsymbol{U}^k + \mathbf{T}_C^F \tilde{\mathbf{P}}_{\text{SDC}}^{-1} \mathbf{T}_F^C \left(\boldsymbol{U}^0 - \mathbf{M}\boldsymbol{U}^k\right) \\ \boldsymbol{U}^{k+1} &= \boldsymbol{U}^{k+\frac{1}{2}} + \mathbf{P}_{\text{SDC}}^{-1} \left(\boldsymbol{U}^0 - \mathbf{M}\boldsymbol{U}^{k+\frac{1}{2}}\right) \end{aligned} \tag{1.24}$$

It is possible to write (1.24) *in form of* (1.16)*, using a new preconditioner $\boldsymbol{P}_{\text{MLSDC}}$, where*

$$\boldsymbol{P}_{\text{MLSDC}}^{-1} = \mathbf{T}_C^F \tilde{\mathbf{P}}_{\text{SDC}}^{-1} \mathbf{T}_F^C + \mathbf{P}_{\text{SDC}}^{-1} - \mathbf{P}_{\text{SDC}}^{-1} \mathbf{M} \mathbf{T}_C^F \tilde{\mathbf{P}}_{\text{SDC}}^{-1} \mathbf{T}_F^C. \tag{1.25}$$

Following (1.18) *yields the MLSDC iteration matrix*

$$\begin{aligned} \mathbf{T}_{\text{MLSDC}} &= \mathbf{I} - \boldsymbol{P}_{\text{MLSDC}}^{-1} \mathbf{M} \\ &= \left(\mathbf{I} - \mathbf{P}_{\text{SDC}}^{-1} \mathbf{M}\right) \left(\mathbf{I} - \mathbf{T}_C^F \tilde{\mathbf{P}}_{\text{SDC}}^{-1} \mathbf{T}_F^C \mathbf{M}\right). \end{aligned} \tag{1.26}$$

Proof. Let $\boldsymbol{U}^k$ 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

$$\boldsymbol{\tau}^k = \tilde{\mathbf{M}}\mathbf{T}_F^C\boldsymbol{U}^k - \boldsymbol{T}_F^C\mathbf{M}\boldsymbol{U}^k \tag{1.27}$$

and use it to modify $\tilde{\boldsymbol{c}}$ for the next iteration on the coarse level. We start the iteration on the coarse level with

$$\tilde{\boldsymbol{U}}^{k+1} = \tilde{\boldsymbol{U}}^k + \tilde{\mathbf{P}}_{\text{SDC}}^{-1}\left(\tilde{\boldsymbol{c}} + \boldsymbol{\tau}^k - \tilde{\mathbf{M}}\tilde{\boldsymbol{U}}^k\right) = \mathbf{T}_F^C\boldsymbol{U}^k + \tilde{\mathbf{P}}_{\text{SDC}}^{-1}\mathbf{T}_F^C\left(\boldsymbol{c} - \mathbf{M}\boldsymbol{U}^k\right),$$

with the restricted value $\tilde{\boldsymbol{U}}^k = \mathbf{T}_F^C\boldsymbol{U}^k$. Then, we compute the coarse correction

$$\boldsymbol{\delta}^k = \mathbf{T}_C^F\left(\tilde{\boldsymbol{U}}^{k+1} - \mathbf{T}_F^C\boldsymbol{U}^k\right)$$

and obtain the half-step

$$\boldsymbol{U}^{k+\frac{1}{2}} = \boldsymbol{U}^k + \boldsymbol{\delta}^k = \boldsymbol{U}^k + \mathbf{T}_C^F\tilde{\mathbf{P}}_{\text{SDC}}^{-1}\mathbf{T}_F^C\left(\boldsymbol{U}^0 - \mathbf{M}\boldsymbol{U}^k\right)$$

after some algebraic manipulations. Using this half-step for the next iteration on the fine level gives (1.24). Simple algebraic manipulations, after inserting the half-step into the second step, yield the preconditioner (1.25), which immediately leads to the iteration matrix (1.26). □

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 that are 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.

1.3.4. The PFASST algorithm

First we explain PFASST on the basis of the schematic representation in Figure 1.2. This is followed by a simplified two level algorithmic representation as it is found in the literature [17]. 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 MLSDC iterations and sends intermediate results on each level to the next processor. The blue and red blocks represent the SDC sweeps on the coarse and fine level. These blocks are connected through FAS corrections to the subjacent blocks (red to blue). The arrows represent the communication between the processors. Before starting with the actual PFASST iterations, a prediction phase,

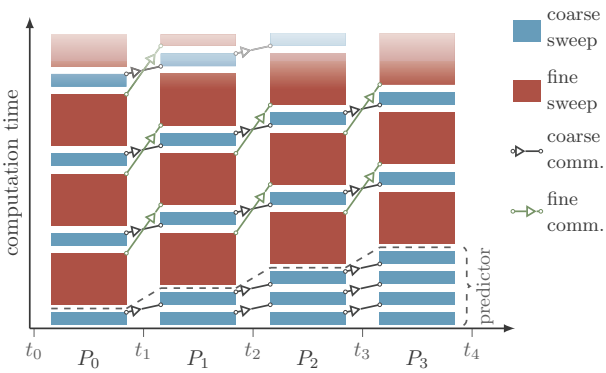


Figure 1.2.: Schematic representation of the PFASST algorithm with two levels and four processes $P_0, ..., P_3$ handling four parallel time steps. Created using `pfasst-tikz` [79].

represented by the first blue blocks near the x-axis, computes suitable initial values for the iterations to come.

Based on the schematic representation and the full algorithm description in [17], we state a two-level version without the prediction phase. Let $U^k_{[t_{l-1},t_l],m}$ be the value on the l-th subinterval at the k-th iteration and the m-th node. We have

$$\boldsymbol{F}^k_{[t_{l-1},t_l]} = [\mathbf{A}\, U^k_{[t_{l-1},t_l],1}, \ldots, \mathbf{A}\, U^k_{[t_{l-1},t_l],M_l}] \text{ and } \boldsymbol{U}^k_{[t_{l-1},t_l]} = \left[U^k_{[t_{l-1},t_l],1}, \ldots, U^k_{[t_{l-1},t_l],M_l}\right],$$

where M_l is the number of nodes on the l-th interval. An upper bar, e.g. $\bar{U}^{k+1}_{l-1}$, 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 $U^k_{[t_{l-1},t_l],1}, \ldots, U^k_{[t_{l-1},t_l],M_l}$. If the last quadrature node coincides with the right interval border just $U^k_{[t_{l-1},t_l],M_l}$ 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 as $U^0_{[t_{l-1},t_l],m}$. Prepared with this notations, we are ready to formulate the PFASST algorithm:

1. Go down to the coarse level:
   a) Restrict the fine values $\boldsymbol{U}^k_{[t_{l-1},t_l]}$ to the coarse values $\tilde{\boldsymbol{U}}^k_{[t_{l-1},t_l]}$ and compute $\tilde{\boldsymbol{F}}^k_{[t_{l-1},t_l]}$.
   b) Compute FAS correction $\boldsymbol{\tau}^k$, using $\tilde{\boldsymbol{F}}^k_{[t_{l-1},t_l]}$ and $\boldsymbol{F}^k_{[t_{l-1},t_l]}$.
   c) If $l > 0$, then receive the new initial value $\tilde{\bar{U}}^k_l$ from processor $\mathbf{P}_{l-1}$ and

compute $\tilde{\boldsymbol{F}}^k_{[t_{l-1},t_l],0}$, or else use the initial value of the ODE.

d) Perform n_C SDC sweeps with values $\tilde{\boldsymbol{U}}^k_{[t_{l-1},t_l]}$, $\tilde{\boldsymbol{F}}^k_{[t_{l-1},t_l]}$ and the FAS correction $\boldsymbol{\tau}^k$. This yields new values $\tilde{\boldsymbol{U}}^{k+\frac{1}{2}}_{[t_{l-1},t_l]}$ and $\tilde{\boldsymbol{F}}^{k+\frac{1}{2}}_{[t_{l-1},t_l]}$.

e) Send $\tilde{U}^{k+\frac{1}{2}}_{[t_{l-1},t_l],M_l}$ to processor $\mathbf{P}_{l+1}$ if $l < N-1$. This will be received as the new initial condition $\tilde{\bar{U}}^k_l$ for the solver on the coarse level.

2. Return to the fine level:

a) Interpolate the coarse correction $\boldsymbol{\delta}^k = \tilde{\boldsymbol{U}}^{k+\frac{1}{2}}_{[t_{l-1},t_l]} - \tilde{\boldsymbol{U}}^k_{[t_{l-1},t_l]}$ and add to $\boldsymbol{U}^k_{[t_{l-1},t_l]}$, yielding $\boldsymbol{U}^{k+\frac{1}{2}}_{[t_{l-1},t_l]}$. Recompute $\boldsymbol{F}^{k+\frac{1}{2}}_{[t_{l-1},t_l]}$.

b) If $l > 0$, then receive the new initial value $\bar{U}^k_{l-1}$ from processor $\mathbf{P}_{l-1}$, or else take the initial value of the ODE.

c) Interpolate coarse correction vector $\delta^k = \tilde{\bar{U}}^{k+\frac{1}{2}}_{l-1} - \tilde{\bar{U}}^k_{l-1}$ and add it to $\bar{U}^k_l$, yielding $\bar{U}^{k+\frac{1}{2}}_l$. Recompute $F^{k+\frac{1}{2}}_{[t_{l-1},t_l],1}$.

3. Perform n_F fine SDC sweeps using the values $\boldsymbol{U}^{k+\frac{1}{2}}_{[t_{l-1},t_l]}$ and $\boldsymbol{F}^{k+\frac{1}{2}}_{[t_{l-1},t_l]}$. This yields values $\boldsymbol{U}^{k+1}_{[t_{l-1},t_l]}$ and $\boldsymbol{F}^{k+1}_{[t_{l-1},t_l]}$.

4. Send $U^{k+1}_{[t_{l-1},t_l],M_l}$ to processor $\mathbf{P}_{l+1}$ if $l < N-1$. This will be used as initial value $\bar{U}^{k+1}_{l+1}$ 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. In the next chapter, we change the perspective to overcome this limitation.

2. A Multigrid Perspective

> We don't want to focus on the trees (or their leaves) at the expense of the forest.
>
> Douglas Hofstadter

As the main result of this chapter, we will provide an iteration matrix in Theorem 3 for PFASST similar to the iteration matrix of MLSDC in Lemma 1. Therefore, instead of building the algorithm in a "vertical" way (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 $[t_0, T]$. The greatest difficulty in this undertaking, is to understand in which form the communication of PFASST is represented. When this difficulty is overcome, the already conjectured multigrid nature of PFASST is finally revealed. With this representation, we elaborate a particular Fourier transformation of the iteration matrix and thus a foundation for the analysis of PFASST. Note that the following sections mostly coincide with the own work submitted in [64].

2.1. Iteration matrix of PFASST

The first step towards the iteration matrix of PFASST is the system matrix of the underlying problem.

2.1.1. The composite collocation problem

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* (1.13) *into L subintervals $[t_l, t_{l+1}]$. On each subinterval a collocation problem in the form of* (1.15)*, denoted by*

$\mathbf{M}_{[t_l,t_{l+1}]}$, *is posed. The collocation matrix on the whole time domain is then defined as*

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

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

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

$$\boldsymbol{c}_{[t_l,t_{l+1}]} = \begin{cases} \boldsymbol{U}_0, & \text{for } l = 0 \\ \mathbf{0}, & \text{for } l > 0 \end{cases} \in \mathbb{R}^{NM},$$

forms the right-hand side $\boldsymbol{c}_{[t_0,T]}$ *for the* ***composite collocation problem***

$$\mathbf{M}_{[t_0,T]} \begin{pmatrix} \boldsymbol{U}_{[t_0,t_1]} \\ \boldsymbol{U}_{[t_1,t_2]} \\ \vdots \\ \boldsymbol{U}_{[t_{L-1},T]} \end{pmatrix} = \begin{pmatrix} \boldsymbol{U}_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \boldsymbol{c}_{[t_0,T]}. \tag{2.1}$$

With this definition, the block structure of our problem becomes evident. On the diagonal of the new collocation matrix, we find blocks of the size NM, each of which are associated with one subinterval $[t_l, t_{l+1}]$. The operators on the subdiagonal deal with the communication between two adjacent subintervals. Note that the structure of the operator $\mathbf{N}$ depends on the choice of the quadrature nodes, since the last quadrature node coincides with the right boundary of the respective subinterval. When designing iterative solvers for the composite collocation problem, it is our goal to exploit this block structure. Therefore, the following two sections are dedicated to the block versions of an approximate Jacobi and an approximate Gauß-Seidel iteration, both emerging 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 2.1.4. Such block-solvers call to mind the block-solvers of the space-time multigrid methods in Section 1.2. Indeed, especially the relaxation operators in [63] appear very similar to the operators introduced here.

2.1.2. 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 a preconditioning matrix. In principle, this strategy may be applied 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 we perform one SDC sweep on each subinterval via

$$\boldsymbol{U}^{k+1}_{[t_l,t_{l+1}]} = \boldsymbol{U}^{k}_{[t_l,t_{l+1}]} + \mathbf{P}^{-1}_{[t_l,t_{l+1}]} \left(\boldsymbol{c}^{k+1}_{[t_l,t_{l+1}]} - \mathbf{M}_{[t_l,t_{l+1}]} \boldsymbol{U}^{k}_{[t_l,t_{l+1}]} \right), \tag{2.2}$$

where $\mathbf{P}_{[t_l,t_{l+1}]}$ denotes the SDC preconditioner (1.17), and $\boldsymbol{c}^{k}_{[t_l,t_{l+1}]}$ 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 $\mathbf{N}$. Therefore, the right-hand side of the collocation problem can be written as

$$\begin{aligned} \boldsymbol{c}^{k}_{[t_0,t_1]} &= [U_0, \ldots, U_0], \quad \text{for} \quad l = 0 \text{ and } k > 0 \\ \boldsymbol{c}^{k}_{[t_l,t_{l+1}]} &= \left[\bar{U}^k_l, \ldots, \bar{U}^k_l \right] = \mathbf{H} \boldsymbol{U}^{k}_{[t_{l-1},t_l]} \quad \text{for} \quad l > 0 \text{ and } k > 1. \end{aligned} \tag{2.3}$$

For some initial iteration vector $\boldsymbol{U}^{0}_{[t_l,t_{l+1}]}$, stemming, e.g., 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 $\mathbf{M}_{[t_0,T]}$ be the matrix of a composite collocation problem, see Definition 2. Using SDC on each subinterval and passing the results via* (2.3) *corresponds to*

$$\boldsymbol{U}^{k+1}_{[t_0,T]} = \boldsymbol{U}^{k}_{[t_0,T]} + \mathbf{P}^{-1}_{[t_0,T]} \left(\boldsymbol{c}_{[t_0,T]} - \mathbf{M}_{[t_0,T]} \boldsymbol{U}^{k}_{[t_0,T]} \right), \tag{2.4}$$

with

$$\boldsymbol{U}^{k}_{[t_0,T]} = \begin{pmatrix} \boldsymbol{U}^{k}_{[t_0,t_1]} \\ \boldsymbol{U}^{k}_{[t_1,t_2]} \\ \vdots \\ \boldsymbol{U}^{k}_{[t_{L-1},T]} \end{pmatrix} \in \mathbb{R}^{NML}, \quad \boldsymbol{c}_{[t_0,T]} = \begin{pmatrix} \boldsymbol{U}_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^{NML}$$

and

$$\mathbf{P}_{[t_0,T]} = \begin{pmatrix} \mathbf{P}_{[t_0,t_1]} & & & \\ -\mathbf{H} & \mathbf{P}_{[t_1,t_2]} & & \\ & \ddots & \ddots & \\ & & -\mathbf{H} & \mathbf{P}_{[t_{L-1},T]} \end{pmatrix} \in \mathbb{R}^{NML \times NML}.$$

Proof. We multiply equation (2.2) with $\mathbf{P}_{[t_l,t_{l+1}]}$ from the left and equation (2.4) with $\mathbf{P}_{[t_0,T]}$ from the left. Comparing the resulting terms line by line reveals the equivalence. □

This Gauß-Seidel-like iteration can be found in Fig. 1.2: Here, after each blue block, which represent SDC sweeps on the coarse level, the values $\bar{U}_l^k$ 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. The communication between subintervals is performed using operator $\mathbf{N}$ on the block sub-diagonal of $\mathbf{P}_{[t_0,T]}$, by passing the value $\bar{U}_l^k$ in form of a new right-hand side to the subsequent solver. This communication is represented in Fig. 1.2 as the arrows connecting the blue blocks.

2.1.3. The approximative block Jacobi solver

The communication, which is 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 blockage of communication is to use an approximate block Jacobi solver, omitting the subdiagonal blocks responsible for the communication.

Assume we perform a step similar to (2.2), but we use the right-hand side

$$\begin{aligned} \boldsymbol{c}^k_{[t_0,t_1]} &= [U_0, \ldots, U_0], \quad \text{for} \quad l = 0 \text{ and } k > 0 \\ \boldsymbol{c}^k_{[t_l,t_{l+1}]} &= \left[\bar{U}_l^{k-1}, \ldots, \bar{U}_l^{k-1}\right] = \mathbf{H}\boldsymbol{U}^{k-1}_{[t_{l-1},t_l]} \quad \text{for} \quad l > 0 \text{ and } k > 1 \end{aligned} \tag{2.5}$$

instead. This means that instead of using the result of the current iteration, the result of the previous iteration 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 $\bar{U}_l^0 = U_0$ for $l > 0$.

Lemma 3. *Let $\mathbf{M}_{[t_0,T]}$ be the matrix of a composite collocation problem, see Definition 2. Then, using SDC on each subinterval and passing the results via* (2.5) *corresponds to*

$$\boldsymbol{U}^{k+1}_{[t_0,T]} = \boldsymbol{U}^k_{[t_0,T]} + \hat{\mathbf{P}}^{-1}_{[t_0,T]} \left(\boldsymbol{c}_{[t_0,T]} - \mathbf{M}_{[t_0,T]}\boldsymbol{U}^k_{[t_0,T]}\right) \tag{2.6}$$

with

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

and $\boldsymbol{U}^k_{[t_0,T]}$, $\boldsymbol{c}_{[t_0,T]}$ *defined as in Lemma 2.*

Proof. Similar to the proof in Lemma 2, a block wise comparison yields the equivalence. Especially the influence of the subdiagonal of $\mathbf{M}_{[t_0,T]}$ on the communication is revealed by a block wise view on (2.6):

$$\begin{aligned} \boldsymbol{U}^{k+1}_{[t_0,t_1]} &= \boldsymbol{U}^k_{[t_0,t_1]} + \mathbf{P}^{-1}_{[t_0,t_1]} \left(\boldsymbol{U}_0 - \mathbf{M}_{[t_0,t_1]} \boldsymbol{U}^k_{[t_0,t_1]} \right), && \text{for } l = 0 \\ \boldsymbol{U}^{k+1}_{[t_l,t_{l+1}]} &= \boldsymbol{U}^k_{[t_l,t_{l+1}]} + \mathbf{P}^{-1}_{[t_l,t_{l+1}]} \left(\mathbf{H}\boldsymbol{U}^k_{[t_{l-1},t_l]} - \mathbf{M}_{[t_l,t_{l+1}]} \boldsymbol{U}^k_{[t_l,t_{l+1}]} \right), && \text{for } l > 1. \end{aligned}$$

The values $\mathbf{N}\boldsymbol{U}^k_{[t_{l-1},t_l]}$ are equivalent to $\mathbf{1}_M \otimes \bar{U}_l^{k-1}$. □

It is evident that due to the block diagonal structure of $\hat{\mathbf{P}}_{[t_0,T]}$, one block Jacobi iteration may be performed concurrently on L computing units. This approach corresponds with the sweeps on the fine (red) blocks in Fig. 1.2; these sweeps can be performed in parallel, since they do not depend on the previous subinterval of the same iteration. Therefore, the iteration on the fine level can be identified with an approximate block Jacobi iteration for the composite collocation problem (2.1).

2.1.4. Assembling PFASST

Already in Section 1.3.3, multigrid elements were incorporated into SDC to construct MLSDC. The same principles apply when we interlace both iterative block solvers from above 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 carried out in parallel. As the following theorem shows, it is now possible to write PFASST in the form of (1.24) and we are able to state an iteration matrix, which has the form of a two-level multigrid iteration matrix.

Theorem 3. *Let* $\mathbf{T}_F^C$ *and* $\mathbf{T}_C^F$ *be block-wise defined transfer operators, which treat the subintervals independently from each other, let*

$$\left\{ \mathbf{P}_{[t_0,t_1]}, \ldots, \mathbf{P}_{[t_{L-1},T]} \right\} \quad \textit{and} \quad \left\{ \tilde{\mathbf{P}}_{[t_0,t_1]}, \ldots, \tilde{\mathbf{P}}_{[t_{L-1},T]} \right\}$$

be sets of the preconditioner for the fine and coarse level, respectively, describing SDC sweeps on $[t_l, t_{l+1}]$ *for* $l \in \{0, \dots, L-1\}$ *and* $t_L = T$*. Let* $\mathbf{M}_{[t_0,T]}$ *be the composite collocation matrix of Definition 2 and* $\mathbf{H}$, $\tilde{\mathbf{H}}$ *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*

$$\begin{aligned} \boldsymbol{U}^{k+\frac{1}{2}}_{[t_0,T]} &= \boldsymbol{U}^{k}_{[t_0,T]} + \mathbf{T}^F_C \tilde{\mathbf{P}}^{-1}_{[t_0,T]} \mathbf{T}^C_F \left(\boldsymbol{c}_{[t_0,T]} - \mathbf{M}_{[t_0,T]} \boldsymbol{U}^{k}_{[t_0,T]} \right) \\ \boldsymbol{U}^{k+1}_{[t_0,T]} &= \boldsymbol{U}^{k}_{[t_0,T]} + \hat{\mathbf{P}}^{-1}_{[t_0,T]} \left(\boldsymbol{c}_{[t_0,T]} - \mathbf{M}_{[t_0,T]} \boldsymbol{U}^{k+\frac{1}{2}}_{[t_0,T]} \right), \end{aligned} \tag{2.7}$$

with $\tilde{\mathbf{P}}_{[t_0,T]}$ *as in Lemma 2 and* $\hat{\mathbf{P}}_{[t_0,T]}$ *as in Lemma 3. In addition, define* $\mathbf{H}$ *and* $\tilde{\mathbf{H}}$ *such that* $\tilde{\mathbf{H}}\mathbf{T}^C_F = \mathbf{T}^C_F \mathbf{H}$ *and* $\boldsymbol{c}_{[t_0,T]} = [\boldsymbol{U}^0, 0, \dots, 0]$ *as well as* $\tilde{\boldsymbol{c}}_{[t_0,T]} = \mathbf{T}^C_F \boldsymbol{c}_{[t_0,T]}$*. Finally, the PFASST iteration matrix is given by*

$$\mathbf{T}_{\text{PFASST}} = \left(\mathbf{I} - \hat{\mathbf{P}}^{-1}_{[t_0,T]} \mathbf{M}_{[t_0,T]} \right) \left(\mathbf{I} - \mathbf{T}^F_C \tilde{\mathbf{P}}^{-1} \mathbf{T}^C_F \mathbf{M}_{[t_0,T]} \right). \tag{2.8}$$

Proof. We compare, systematically, each step of PFASST with the computations on the subinterval found in equation (2.7), which expands into

$$\boldsymbol{\tau}^k_{[t_0,T]} = \tilde{\mathbf{M}}_{[t_0,T]} \mathbf{T}^C_F \boldsymbol{U}^k_{[t_0,T]} - \mathbf{T}^C_F \mathbf{M}_{[t_0,T]} \boldsymbol{U}^k_{[t_0,T]} \tag{2.9}$$

$$\tilde{\boldsymbol{U}}^{k+1}_{[t_0,T]} = \tilde{\boldsymbol{U}}^{k}_{[t_0,T]} + \tilde{\mathbf{P}}^{-1} \left(\tilde{\boldsymbol{c}}_{[t_0,T]} + \boldsymbol{\tau}^k_{[t_0,T]} - \tilde{\mathbf{M}}_{[t_0,T]} \tilde{\boldsymbol{U}}^{k}_{[t_0,T]} \right) \tag{2.10}$$

$$\boldsymbol{U}^{k+\frac{1}{2}}_{[t_0,T]} = \boldsymbol{U}^{k}_{[t_0,T]} + \mathbf{T}^F_C \left(\tilde{\boldsymbol{U}}^{k+1}_{[t_0,T]} - \mathbf{T}^C_F \boldsymbol{U}^k_{[t_0,T]} \right) \tag{2.11}$$

$$\boldsymbol{U}^{k+1}_{[t_0,T]} = \boldsymbol{U}^{k+\frac{1}{2}}_{[t_0,T]} + \hat{\mathbf{P}}^{-1} \left(\boldsymbol{c}_{[t_0,T]} - \mathbf{M}_{[t_0,T]} \boldsymbol{U}^{k+\frac{1}{2}}_{[t_0,T]} \right). \tag{2.12}$$

From top to bottom, we have the computation of the FAS τ-correction of the right-hand side, the SDC sweep on the coarse level, the FAS δ-correction of the updated value, and the SDC sweep on the fine level. PFASST's communication between the subintervals has already been derived in Lemmas 2 and 3. The evaluations of the right-hand side, in the form of $\boldsymbol{F}$ and $\tilde{\boldsymbol{F}}$, are included in the matrix vector multiplication with $\mathbf{M}_{[t_0,T]}$ and $\tilde{\mathbf{M}}_{[t_0,T]}$, respectively.

The computation of the FAS correction $\boldsymbol{\tau}^k_{[t_0,T]}$, as in (2.9), differs from the formula (1.27), which we derived for MLSDC, as the latter is only applied on one subinterval. This leads to the FAS correction vector of (2.9) having additional terms

$$\mathbf{L} = \tilde{\mathbf{H}} \mathbf{T}^C_F - \mathbf{T}^C_F \mathbf{H} \tag{2.13}$$

with

$$\tau_{[t_0,T]} = \left(\tau_{[t_0,t_1]},\ \tau_{[t_1,t_2]} + \mathbf{L}\boldsymbol{U}^k_{[t_0,t_1]},\ ...,\ \tau_{[t_{N-1},T]} + \mathbf{L}\boldsymbol{U}^k_{[t_{N-1},T]}\right)^T \tag{2.14}$$

However, by requirement we have $\mathbf{L} = \mathbf{0}$ 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* $\mathbf{T}_F^C = \mathbf{I}_M \otimes \bar{\mathbf{T}}_F^C$*, where* $\bar{\mathbf{T}}_F^C$ *is the restriction operator in space. In this case, it yields*

$$\begin{aligned}\mathbf{L} &= \tilde{\mathbf{H}} \cdot \left(\mathbf{I}_M \otimes \bar{\mathbf{T}}_F^C\right) - \left(\mathbf{I}_M \otimes \bar{\mathbf{T}}_F^C\right) \cdot \mathbf{H} \\ &= \mathbf{N} \otimes \left(\bar{\mathbf{T}}_F^C - \bar{\mathbf{T}}_F^C\right) = \mathbf{0}\end{aligned}$$

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

$$\mathbf{L} = \left(\tilde{\mathbf{N}}\hat{\mathbf{T}}_F^C - \hat{\mathbf{T}}_F^C\mathbf{N}\right) \otimes \bar{\mathbf{T}}_F^C$$

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

$$\begin{aligned}\mathbf{0} &= \hat{\mathbf{T}}_F^C\mathbf{N} - \tilde{\mathbf{N}}\hat{\mathbf{T}}_F^C \\ &= \begin{pmatrix} 0 & \cdots & 0 & \sum_{j=1}^M t_{1,j} \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & \sum_{j=1}^M t_{\tilde{M},j} \end{pmatrix} - \begin{pmatrix} t_{\tilde{M},1} & \cdots & t_{\tilde{M},M-1} & t_{\tilde{M},M} \\ \vdots & & \vdots & \vdots \\ t_{\tilde{M},1} & \cdots & t_{\tilde{M},M-1} & t_{\tilde{M},M} \end{pmatrix}.\end{aligned}$$

Hence, we require that

$$t_{\tilde{M},j} = 0 \quad \forall\, j \in \{1,\dots,M-1\} \qquad \textit{and} \qquad t_{\tilde{M},M} = \sum_{j=1}^M t_{i,j} \quad \forall\, i \in \left\{1,\dots,\tilde{M}\right\}.$$

If the restriction $\hat{\mathbf{T}}_F^C$ *of a constant vector yields a constant vector with the same values*

but a smaller dimension, we infer that

$$\sum_{j=1}^{M} t_{i,j} = 1 \quad \forall\, i \in \left\{1, \tilde{M}\right\}$$

and hence $t_{\tilde{M},M} = 1$. *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 e.g., for the linear restriction or simple injection, as long as* $\tilde{\tau}_{\tilde{M}} = \tau_M$.

The hierarchy of discretizations on which PFASST is working and the exchange of information between those levels using coarse-grid correction clearly indicates a strong similarity to classical multigrid methods. This relation is in particular emphasized by the iteration matrix. Standard multigrid methods are typically described and analyzed by their iteration matrix $\mathbf{T}_{\text{MG}}$, which reads

$$\mathbf{T}_{\text{MG}}(\nu, \eta) = \left(\mathbf{I} - \mathbf{P}_{\text{post}}^{-1}\mathbf{M}\right)^{\nu} \left(\mathbf{I} - \mathbf{T}_C^F \tilde{\mathbf{M}}^{-1} \mathbf{T}_F^C \mathbf{M}\right) \left(\mathbf{I} - \mathbf{P}_{\text{pre}}^{-1}\mathbf{M}\right)^{\eta}, \tag{2.15}$$

for ν post- and η pre-smoothing steps. The expression in the middle is the coarse grid correction and $\tilde{\mathbf{M}}$ is a coarse version of $\mathbf{M}$. 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 form of $\tilde{\mathbf{P}}^{-1}$. PFASST does exactly this. Under the conditions of Theorem 3, the comparison of (2.15) and (2.8) 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, the smoothing and approximation property 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 which are usually used for multigrid schemes.

2.2. Fourier transformation of the iteration matrix

The most common tool for the analysis and the design of multigrid algorithms is the local Fourier analysis (LFA), which was initially introduced by Brandt in [56], cf. [52]. 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. in [80] 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 problem 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 stay consistent 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

$$\varphi_\theta(x) = \exp\left(i\theta x/h\right), \quad x \in [0,1], \quad \theta \in [0, 2\pi), \tag{2.16}$$

for distinct frequencies θ. 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 first avoid the use of explicit Fourier symbols. Rather, we perform a block diagonalization of the matrices of PFASST. The goal is the block-wise 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 degree.

Due to the periodicity in space, parts of the iteration matrix consists of circulant matrices. A circulant matrix is a special kind of Toeplitz matrix, where each row vector is shifted one element to the right, relative to the preceding row vector. More precisely, we denote

$$\mathbf{C} = \begin{pmatrix} c_0 & c_1 & \cdots & c_{N-1} \\ c_{N-1} & c_0 & c_1 & \\ & \ddots & \ddots & \\ c_1 & \cdots & c_{N-1} & c_0 \end{pmatrix}. \tag{2.17}$$

This matrix has the eigenvalues λ_k and eigenvectors ψ_k for $k = 0, \ldots, N-1$ with

$$\begin{aligned} \lambda_k &= \sum_{j=0}^{N-1} c_j \exp\left(i\frac{2\pi}{N}k \cdot j\right) \quad \text{and} \\ \psi_k &= \frac{1}{\sqrt{N}}\left[\exp\left(i\frac{2\pi}{N}k \cdot 0\right), \exp\left(i\frac{2\pi}{N}k \cdot 1\right), \ldots, \exp\left(i\frac{2\pi}{N}k \cdot (N-1)\right)\right]^T. \end{aligned} \tag{2.18}$$

This also means that with the transformation matrix $\mathbf{\Psi}$, which is orthogonal and consists

of the eigenvectors, it holds

$$\left(\mathbf{\Psi}^T \mathbf{C} \mathbf{\Psi}\right)_{j,j} = \lambda_j. \tag{2.19}$$

For two diagonalizable matrices $\mathbf{A}, \mathbf{B}$ with the same eigenvector space it holds

$$\begin{aligned} \mathbf{\Psi}^T \left(\mathbf{A} + \mathbf{B}\right) \mathbf{\Psi} &= \mathbf{\Psi}^T \mathbf{A} \mathbf{\Psi} + \mathbf{\Psi}^T \mathbf{B} \mathbf{\Psi} = \mathbf{D}^{(A)} + \mathbf{D}^{(B)}, \\ \mathbf{\Psi}^T \mathbf{A} \mathbf{B} \mathbf{\Psi} &= \mathbf{\Psi}^T \mathbf{A} \mathbf{\Psi} \mathbf{\Psi}^T \mathbf{B} \mathbf{\Psi} = \mathbf{D}^{(A)} \mathbf{D}^{(B)}, \\ \mathbf{\Psi}^T \mathbf{A}^{-1} \mathbf{\Psi} &= \left(\mathbf{D}^{(A)}\right)^{-1} \end{aligned} \tag{2.20}$$

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

2.2.1. The three layers

The PFASST algorithm operates on three layers. 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 $\mathbf{M}_{[t_0,T]}$ under the assumption that we have the same problem (i.e. the same discretization of the same operator) on each subinterval

$$\mathbf{M}_{[t_0,T]} = \mathbf{I}_L \otimes \mathbf{I}_M \otimes \mathbf{I}_N - \Delta t \cdot \mathbf{I}_L \otimes \mathbf{Q} \otimes \mathbf{A} - \mathbf{E} \otimes \mathbf{N} \otimes \mathbf{I}_N, \tag{2.21}$$

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 $\mathbf{E} \in \mathbb{R}^{L \times L}$ 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 $\mathbf{\Psi}$. With the help of the permutation matrix $\mathcal{P}$, we define a transformation matrix $\mathcal{F}$, which effects and reorders the layers, as

$$\mathcal{F} = \mathcal{P} \cdot \left(\mathbf{I}_L \otimes \mathbf{I}_M \otimes \mathbf{\Psi}\right), \quad \mathcal{F}^{-1} = \left(\mathbf{I}_L \otimes \mathbf{I}_M \otimes \mathbf{\Psi}^T\right) \cdot \mathcal{P}^{-1},$$

and therefore

$$\mathcal{F}^{-1} \mathbf{M}_{[t_0,T]} \mathcal{F} = \mathbf{I}_N \otimes \left(\mathbf{I}_L \otimes \mathbf{I}_M - \mathbf{E} \otimes \mathbf{N}\right) - \Delta t \cdot \mathbf{D}^{(A)} \otimes \mathbf{I}_L \otimes \mathbf{Q}.$$

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

$$\mathcal{F}^{-1}\mathbf{M}_{[t_0,T]}\mathcal{F} = \text{diag}\left(\mathbf{B}_1^{(M_{[t_0,T]})}, ..., \mathbf{B}_N^{(M_{[t_0,T]})}\right)$$
$$\text{with} \quad \mathbf{B}_j^{(M_{[t_0,T]})} = \mathbf{I}_L \otimes \mathbf{I}_M - \mathbf{E} \otimes \mathbf{N} - \Delta t \cdot \lambda_j \mathbf{I}_L \otimes \mathbf{Q}.$$

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 transformation strategy above leads to a block structure for all matrices, which emerge in the formulation of PFASST; particularly for the iteration matrix. Here however, the interpolation and restriction matrices need special attention.

2.2.2. 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. We define a special class of interpolation and restriction pairs between these two grids.

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

$$X = [x_1, \ldots, x_N] \;\; \textit{and} \;\; \tilde{X} = [\tilde{x}_1, \ldots, \tilde{x}_{N/2}] \;\textit{,with}\; x_{2j-1} = \tilde{x}_j \;\textit{for all}\; j \in \left\{1, \ldots, \frac{N}{2}\right\}.$$

Let $\mathcal{W}(.,.) : \mathbb{R}^{N/2\times N/2} \times \mathbb{R}^{N/2\times N/2} \mapsto \mathbb{R}^{N\times N/2}$ 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

$$\Pi = \left\{\mathbf{T}_C^F : \exists \mathbf{C} \in \mathbb{R}^{N/2\times N/2} \textit{ circulant and } \mathbf{C}\cdot\mathbf{1} = \mathbf{1}, \mathbf{T}_C^F = \mathcal{W}(\mathbf{I}_{N/2}, \mathbf{C})\right\} \tag{2.22}$$

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

$$\Pi^T = \left\{\mathbf{T}_F^C : c\left(\mathbf{T}_F^C\right)^T \in \Pi,\; c \in \mathbb{R}\right\}. \tag{2.23}$$

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

Lemma 4. *Let $\mathbf{T}_C^F$ be a CI-interpolation and $\mathbf{T}_F^C$ the associated CI-restriction operator, $\mathbf{\Psi}$ the transformation matrix for N grid points and $\mathbf{\Psi}_C$ the transformation matrix for*

$N/2$ grid points. Then it holds

$$\mathbf{\Psi}^T \mathbf{T}_C^F \mathbf{\Psi}_C = \begin{pmatrix} d_0 & & & & \\ & \ddots & & & \\ & & d_{N/2-1} & & \\ \hat{d}_0 & & & & \\ & \ddots & & & \\ & & & \hat{d}_{N/2-1} & \end{pmatrix} \tag{2.24}$$

and

$$\mathbf{\Psi}_C^T \mathbf{T}_F^C \mathbf{\Psi} = c \begin{pmatrix} d_0 & & & \hat{d}_0 & & \\ & \ddots & & & \ddots & \\ & & d_{N/2-1} & & & \hat{d}_{N/2-1} \end{pmatrix}. \tag{2.25}$$

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

$$d_k = \frac{1 + \lambda_k^{(C)} \exp(-i\frac{2\pi}{N}k)}{\sqrt{2}} \quad \textit{and} \quad \hat{d}_k = \frac{1 - \lambda_k^{(C)} \exp(-i\frac{2\pi}{N}k)}{\sqrt{2}}. \tag{2.26}$$

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

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

Using the eigenvector-eigenvalue relation (2.18) of the two circulant matrices $\mathbf{C}$ and $\mathbf{I}$, see Section 2.2, for the computation of the k-th column, we have

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

Multiplying (2.24) by the eigenvector Ψ_k from the left yields

$$\left[\mathbf{T}_C^F \cdot \mathbf{\Psi}_C\right]_{-,k} \overset{!}{=} \frac{d_k}{\sqrt{N}} \begin{pmatrix} \exp(i2\pi/Nk \cdot 0) \\ \vdots \\ \exp(i2\pi/Nk \cdot (N-1)) \end{pmatrix} + \frac{\hat{d}_k}{\sqrt{N}} \begin{pmatrix} \exp(i2\pi/N(N/2+k) \cdot 0) \\ \vdots \\ \exp(i2\pi/N(N/2+k) \cdot (N-1)) \end{pmatrix}.$$

Then solve the resulting equation row-for-row to find that it has an unique solution (2.26). □

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 $\mathbf{C}$ be, so that*

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

In this case, $\mathbf{C}$ is generated by an symmetric stencil with an odd number of entries. Then, it holds

$$d_k = d_{N/2-k} \quad \textit{and} \quad \hat{d}_k = \hat{d}_{N/2-k}.$$

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

$$d_0 = 0 \quad \textit{and} \quad \hat{d}_0 = \sqrt{2}.$$

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

obtain

$$
\begin{aligned}
\boldsymbol{\Psi}^T\mathbf{T}_C^F\tilde{\mathbf{A}}^{-1}\mathbf{T}_F^C\boldsymbol{\Psi} &= \boldsymbol{\Psi}^T\mathbf{T}_C^F\boldsymbol{\Psi}_C\boldsymbol{\Psi}_C^T\tilde{\mathbf{A}}^{-1}\boldsymbol{\Psi}_C\boldsymbol{\Psi}_C^T\mathbf{T}_F^C\boldsymbol{\Psi} \\
&= \frac{1}{2}\begin{pmatrix} d_0 & & & \\ & \ddots & & \\ & & d_{\frac{N}{2}-1} & \\ \hat{d}_0 & & & \\ & \ddots & & \\ & & \hat{d}_{\frac{N}{2}-1} & \end{pmatrix}\begin{pmatrix} \tilde{\lambda}_0^{-1} & & \\ & \ddots & \\ & & \tilde{\lambda}_{\frac{N}{2}-1}^{-1} \end{pmatrix}\begin{pmatrix} f_0 & & & \hat{f}_0 & & \\ & \ddots & & & \ddots & \\ & & f_{\frac{N}{2}-1} & & & \hat{f}_{\frac{N}{2}-1} \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} d_1\tilde{\lambda}_0^{-1}f_0 & & & & d_0\tilde{\lambda}_0^{-1}\hat{f}_0 & & \\ & \ddots & & & & \ddots & \\ & & d_{\frac{N}{2}-1}\tilde{\lambda}_{\frac{N}{2}-1}^{-1}f_{\frac{N}{2}-1} & & & & d_{\frac{N}{2}-1}\tilde{\lambda}_{\frac{N}{2}-1}^{-1}\hat{f}_{\frac{N}{2}-1} \\ \hat{d}_0\tilde{\lambda}_0^{-1}f_0 & & & & \hat{d}_0\tilde{\lambda}_0^{-1}\hat{f}_0 & & \\ & \ddots & & & & \ddots & \\ & & \hat{d}_{\frac{N}{2}-1}\tilde{\lambda}_{\frac{N}{2}-1}^{-1}f_{\frac{N}{2}-1} & & & & \hat{d}_{\frac{N}{2}-1}\tilde{\lambda}_{\frac{N}{2}-1}^{-1}\hat{f}_{\frac{N}{2}-1} \end{pmatrix}.
\end{aligned}
$$

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

$$\mathcal{P}^{-1}\boldsymbol{\Psi}^T\mathbf{T}_C^F\tilde{\mathbf{A}}^{-1}\mathbf{T}_F^C\boldsymbol{\Psi}\mathcal{P} = \operatorname{diag}\left(\mathbf{B}_0,\ldots,\mathbf{B}_{\frac{N}{2}-1}\right), \tag{2.27}$$

$$\text{where}\quad \mathbf{B}_l = \begin{pmatrix} d_l\tilde{\lambda}_l^{-1}f_l & d_l\tilde{\lambda}_l^{-1}\hat{f}_l \\ \hat{d}_l\tilde{\lambda}_l^{-1}f_l & \hat{d}_l\tilde{\lambda}_l^{-1}\hat{f}_l \end{pmatrix} \in \mathbb{R}^{2\times 2}. \tag{2.28}$$

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 harmonic pairs. In Section 2.3, we will formally introduce these harmonic pairs.

2.2.3. Transforming the full iteration matrix

The iteration matrix of PFASST can now be transformed into a block matrix with $N/2$ blocks of the 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 4. *Let us have a iteration matrix in the form of* (2.8) *with*

$$\mathbf{T} = \left(\mathbf{I} - \mathbf{P}^{-1}\mathbf{M}\right)\left(\mathbf{I} - \mathbf{T}_C^F \tilde{\mathbf{P}}^{-1}\mathbf{T}_F^C\mathbf{M}\right),$$

where $\mathbf{M}$ *is the collocation matrix,* $\mathbf{T}_F^C, \mathbf{T}_C^F$ *are two circulant interweaved transfer operators and* $\mathbf{P}, \tilde{\mathbf{P}}$ *are two preconditioners with a matrix in the spatial layer, which is diagonalizable and has the same eigenvector space as the spatial system matrix* $\mathbf{A}$*. Then, there exists a transformation* $\mathcal{F}$*, so that*

$$\mathcal{F}^{-1}\mathbf{T}\mathcal{F} = \operatorname{diag}\left(\mathcal{B}_0^{(S)}\mathcal{B}_0^{(CGC)}, \ldots, \mathcal{B}_{\frac{N}{2}-1}^{(S)}\mathcal{B}_{\frac{N}{2}-1}^{(CGC)}\right) \in \mathbb{R}^{LMN\times LMN} \quad \text{, with} \tag{2.29}$$

$$\mathcal{B}_k^{(S)} = \begin{pmatrix} \mathbf{I} - \left(\mathbf{B}_k^{(P)}\right)^{-1}\mathbf{B}_k^{(M)} & \\ & \mathbf{I} - \left(\mathbf{B}_{\frac{N}{2}+k}^{(P)}\right)^{-1}\mathbf{B}_{\frac{N}{2}+k}^{(M)} \end{pmatrix} \in \mathbb{R}^{2LM\times 2LM} \tag{2.30}$$

$$\mathcal{B}_k^{(CGC)} = \begin{pmatrix} \mathbf{I} - f_k d_k \left(\mathbf{B}_k^{(\tilde{P})}\right)^{-1}\mathbf{B}_k^{(M)} & -\hat{f}_k d_k \left(\mathbf{B}_k^{(\tilde{P})}\right)^{-1}\mathbf{B}_{\frac{N}{2}+k}^{(M)} \\ -\hat{d}_k f_k \left(\mathbf{B}_k^{(\tilde{P})}\right)^{-1}\mathbf{B}_k^{(M)} & \mathbf{I} - \hat{f}_k\hat{d}_k \left(\mathbf{B}_k^{(\tilde{P})}\right)^{-1}\mathbf{B}_{\frac{N}{2}+k}^{(M)} \end{pmatrix} \in \mathbb{R}^{2LM\times 2LM}, \tag{2.31}$$

with matrices $\mathbf{B}_k^{(P)}, \mathbf{B}_k^{(M)} \in \mathbb{R}^{LM\times LM}$ *for* $k = 0 \ldots N-1$ *and* $\mathbf{B}_k^{(\tilde{P})} \in \mathbb{R}^{LM\times LM}$ *for* $k = 0 \ldots \frac{N}{2} - 1$*, solely depending on the eigenvalues of* $\mathbf{A}$ *and* $\tilde{\mathbf{A}}$*. Where*

$$\begin{aligned}
\boldsymbol{\Psi}^T\mathbf{P}\boldsymbol{\Psi} &= \operatorname{diag}\left(\mathbf{B}_0^{(P)}, \ldots, \mathbf{B}_{N-1}^{(P)}\right), \text{with } \mathbf{B}_j^{(P)} = \mathbf{I}_L \otimes \mathbf{I}_M - \Delta t \cdot \lambda_j^{(A)}\mathbf{I}_L \otimes \mathbf{Q}_{\boldsymbol{\Delta}}, \\
\boldsymbol{\Psi}^T\mathbf{M}\boldsymbol{\Psi} &= \operatorname{diag}\left(\mathbf{B}_0^{(M)}, \ldots, \mathbf{B}_{N-1}^{(M)}\right), \text{with } \mathbf{B}_j^{(M)} = \mathbf{I}_L \otimes \mathbf{I}_M - \mathbf{E} \otimes \mathbf{N} - \Delta t \cdot \lambda_j^{(A)}\mathbf{I}_L \otimes \mathbf{Q}, \\
\boldsymbol{\Psi}^T\tilde{\mathbf{P}}\boldsymbol{\Psi} &= \operatorname{diag}\left(\mathbf{B}_0^{(\tilde{P})}, \ldots, \mathbf{B}_{\frac{N}{2}-1}^{(\tilde{P})}\right), \text{with } \mathbf{B}_j^{(\tilde{P})} = \mathbf{I}_L \otimes \mathbf{I}_M - \mathbf{E} \otimes \mathbf{N} - \Delta t \cdot \lambda_j^{(\tilde{A})}\mathbf{I}_L \otimes \mathbf{Q}_{\boldsymbol{\Delta}}.
\end{aligned} \tag{2.32}$$

We call $\mathbf{B}_j^{(M)}, \mathbf{B}_j^{(P)}$ *and* $\mathbf{B}_j^{(\tilde{P})}$ *basic blocks. The matrix* $\mathbf{Q}_\Delta \in \mathbb{R}^{M\times M}$ *is a lower triangular matrix approximating* $\mathbf{Q}$*, see Section 1.3.2.*

Proof. The proof is rather straightforward. The matrices $\mathbf{P}, \mathbf{M}$ and $\tilde{\mathbf{P}}$ have 3 layers, separated by Kronecker products like in (2.21). Applying the transformation in the spatial dimension leads to the basic blocks (2.32). Similar to (2.28), we choose the adequate permutation matrices on the layers of subintervals and quadrature nodes, to get the blocks of harmonic pairs. Also, each block of the post smoother is associated with a mode, hence we stack harmonic pairs together to $\mathcal{B}_k^{(S)}$ in order to match them

with the blocks of the coarse grid correction $\mathcal{B}_k^{(CGC)}$, performed by the same permutation matrix. □

This theorem makes it possible, at least semi-algebraically, to analyze the convergence properties of PFASST by computing the spectral radius of each block $\left(\mathcal{B}_k^{(S)}\mathcal{B}_k^{(CGC)}\right)$. Until this point, the choice of the particular problem and operators yields a rigorous transformation. Hence, the blocks and the full iteration matrix of PFASST have exactly the same eigenvalues. This translates to computing $N/2$ eigenvalue decompositions of matrices of the size $2ML \times 2ML$. As we can see in (2.32), the basic blocks consist of $\mathbf{I}_L$ and $\mathbf{E}$ on the first layer. However, it is not directly possible to apply the transformation strategy presented above 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 rigorousness of the transformation in order to find a decomposition of the time collocation blocks into L much smaller blocks of size $2M \times 2M$.

2.2.4. Assuming periodicity in time

To enable the further decomposition of the basic blocks, we substitute

$$\mathbf{E} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 1 & 0 & & \\ \vdots & & \ddots & \\ 0 & & 1 & 0 \end{pmatrix} \quad \text{with} \quad \hat{\mathbf{E}} = \begin{pmatrix} 0 & 0 & \cdots & 1 \\ 1 & 0 & & \\ \vdots & & \ddots & \\ 0 & & 1 & 0 \end{pmatrix} \tag{2.33}$$

in the matrix formulation of PFASST. This introduces time periodicity to the problem and makes the matrix circulant. Hence, it becomes easy to transform

$$\left[\mathbf{\Psi}^{-1}\hat{\mathbf{E}}\mathbf{\Psi}\right]_{j,j} = \exp\left(-i2\pi\frac{j}{L}\right), \tag{2.34}$$

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

Theorem 5. *Let us have the identical requirements as in Theorem 4, with the exception of the use of* $\hat{\mathbf{E}}$ *instead of* $\mathbf{E}$*. Then there exists a transformation* $\hat{\mathcal{F}}$*, such that*

$$\hat{\mathcal{F}}^{-1}\mathbf{T}\hat{\mathcal{F}} = \operatorname{diag}\left(\mathcal{B}_{0,0}^{(S)} \cdot \mathcal{B}_{0,0}^{(CGC)}, \mathcal{B}_{0,1}^{(S)} \cdot \mathcal{B}_{0,1}^{(CGC)}, \ldots, \mathcal{B}_{\frac{N}{2}-1,L-1}^{(S)} \cdot \mathcal{B}_{\frac{N}{2}-1,L-1}^{(CGC)}\right), \tag{2.35}$$

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

$$\mathcal{B}_{k,j}^{(S)} = \begin{pmatrix} \mathbf{I} - \left(\mathbf{B}_{k,j}^{(P)}\right)^{-1} \mathbf{B}_{k,j}^{(M)} & \\ & \mathbf{I} - \left(\mathbf{B}_{\frac{N}{2}+k,j}^{(P)}\right)^{-1} \mathbf{B}_{\frac{N}{2}+k,j}^{(M)} \end{pmatrix} \in \mathbb{R}^{2M\times 2M} \text{ and} \quad (2.36)$$

$$\mathcal{B}_{k,j}^{(CGC)} = \begin{pmatrix} \mathbf{I} - f_k d_k \left(\mathbf{B}_{k,j}^{(\tilde{P})}\right)^{-1} \mathbf{B}_{k,j}^{(M)} & -\hat{f}_k d_k \left(\mathbf{B}_{k,j}^{(\tilde{P})}\right)^{-1} \mathbf{B}_{N/2+k,j}^{(M)} \\ -\hat{d}_k f_k \left(\mathbf{B}_{k,j}^{(\tilde{P})}\right)^{-1} \mathbf{B}_{k,j}^{(M)} & \mathbf{I} - \hat{f}_k \hat{d}_k \left(\mathbf{B}_{k,j}^{(\tilde{P})}\right)^{-1} \mathbf{B}_{N/2+k,j}^{(M)} \end{pmatrix} \in \mathbb{R}^{2M\times 2M}, \quad (2.37)$$

with matrices $\mathbf{B}_{k,j}^{(P)}, \mathbf{B}_{k,j}^{(M)} \in \mathbb{R}^{M\times M}$ *for* $k = 0 \dots N-1, j = 0 \dots L-1$ *and* $\mathbf{B}_{k,j}^{(\tilde{P})} \in \mathbb{R}^{M\times M}$ *for* $k = 0 \dots \frac{N}{2}-1, j = 0 \dots L-1$*, solely depending on the eigenvalues of* $\mathbf{A}$ *and* $\tilde{\mathbf{A}}$*, with*

$$\begin{aligned} \mathbf{\Psi}^T \mathbf{P} \mathbf{\Psi} &= \operatorname{diag}\left(\mathbf{B}_{0,0}^{(P)}, \dots, \mathbf{B}_{N-1,L-1}^{(P)}\right), \text{with } \mathbf{B}_{k,j}^{(P)} = \mathbf{I} - \lambda_k^{(A)} \Delta t \mathbf{Q}_\Delta, \\ \mathbf{\Psi}^T \mathbf{M} \mathbf{\Psi} &= \operatorname{diag}\left(\mathbf{B}_{0,0}^{(M)}, \dots, \mathbf{B}_{N-1,L-1}^{(M)}\right), \text{with } \mathbf{B}_{k,j}^{(M)} = \mathbf{I} - \lambda_k^{(A)} \Delta t \mathbf{Q} - \exp\left(-i2\pi \frac{j}{L}\right) \mathbf{N}, \\ \mathbf{\Psi}^T \tilde{\mathbf{P}} \mathbf{\Psi} &= \operatorname{diag}\left(\mathbf{B}_{0,0}^{(\tilde{P})}, \dots, \mathbf{B}_{\frac{N}{2}-1,L-1}^{(\tilde{P})}\right), \text{with } \mathbf{B}_{k,j}^{(\tilde{P})} = \mathbf{I} - \lambda_k^{(\tilde{A})} \Delta t \mathbf{Q}_\Delta - \exp\left(-i2\pi \frac{j}{L}\right) \mathbf{N}. \end{aligned} \quad (2.38)$$

We denote these blocks as "collocation blocks", in contrast to the time-collocation blocks of Theorem 4.

This leaves us with $NL/2$ blocks of the size $2M \times 2M$. We identify the matrices $\mathbf{Q}$ and $\mathbf{Q}_\Delta$ as the atomic part of the whole matrix formulation. Further decompositions may only be performed if a decomposition of $\mathbf{Q}$ is found. In the case of a $\mathbf{Q} \in \mathbb{R}^{1\times 1}$, the time stepping part reduces to, e.g., 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* $c \in \mathbb{R}$*. Therefore, neither the inverse of* $\mathbf{B}_{k,0}^{(P)}$ *and* $\mathbf{B}_{k,0}^{(\tilde{P})}$ *nor the inverse of the iteration matrix blocks* $\mathcal{B}_{k,0}^{(T)} = \mathcal{B}_{k,0}^{(S)} \cdot \mathcal{B}_{k,0}^{(CGC)}$ *exist. Our remedy for this problem is to set* $\mathcal{B}_{k,0}^{(T)}$ *to* $\mathbf{0}$*. These blocks belong to constant modes and we assume that there are no constant error modes which have to be damped.*

2.3. Using the Fourier transformed iteration matrix

Through the Fourier transformation, we have a relation between the various block matrices and the frequencies $\theta \in [0, 2\pi]$. As described in Section 1.2.1, a well-designed multigrid reduces the high frequencies on the fine level and the low frequencies on the coarse level. One of our goals is to find out if PFASST shares this property. Therefore, we denote

$$\Theta_{\text{high}} = (\pi/2, 3\pi/2) \quad \text{and} \quad \Theta_{\text{low}} = [0, 2\pi] \setminus (\pi/2, 3\pi/2)\,,$$

as the high and low frequencies. To describe the relation between matrix blocks and the frequencies in more detail, we need to elaborate on the idea of harmonics. The concept of harmonics is found in all fields where waves play an important role, such as music theory or physics. Assume a basis wave with a certain frequency, then every wave with an integer multiple k of this frequency is denoted as the k-th harmonic of the basis wave. In the case of two superposed equidistant grids, where the coarser grid has half the number of points, the harmonic and basis wave represented on the fine grid are indistinguishable on the coarse grid. Thus, we need the composition of the basic block matrices to construct the time collocation block for the coarse grid correction. More precisely, we recognize that the block matrices $\mathbf{B}_k^{(M)}$ and $\mathbf{B}_{k+N/2}^{(M)}$ are employed. Since we assume a circulant system matrix for the spatial problem, the index k is directly related to the frequency $\theta_k = 2\pi k/N$. This relation is evident from (2.18). Consequently, the index $k + N/2$ is directly related to the frequency

$$\theta_{k+N/2} = \frac{2\pi\left(k + \frac{N}{2}\right)}{N} = \frac{2\pi k}{N} + \pi = \theta_k + \pi.$$

Thus, we denote the harmonic frequency as

$$\theta' = \begin{cases} \theta + \pi & ,\theta \leq \pi \\ \theta - \pi & ,\text{else} \end{cases} \tag{2.39}$$

and in order to have a mapping between harmonic and base frequencies we further denote

$$\Theta_{\text{left}} = [0, \pi) \quad \text{and} \quad \Theta_{\text{right}} = [\pi, 2\pi)\,,$$

as the left and right frequencies, and (θ, θ') as a harmonic pair. Note that it is important to distinguish left and right frequencies from the high and low frequencies, since both have different roles in the interaction of the different constituents of PFASST. In Figure 2.1, these different frequencies are presented. With these definitions, it is natural to make the transition from a block structure, with an finite number of blocks, to matrix symbols, which depend on the continuous frequency value θ.

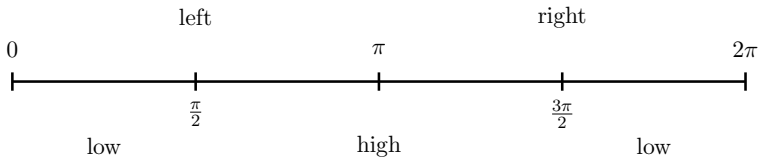


Figure 2.1.: Overview over how the frequency spectrum is subdivided.

2.3.1. Conversion to matrix symbols

A reinterpretation of the time collocation blocks into time collocation matrix symbols liberates us from the discretization of the spatial dimension. Assuming periodicity in time only liberates us from a part of the temporal discretization, since the number of subintervals and the frequency in the temporal dimension are merged by the reformulation of the time collocation blocks into collocation blocks. More precisely, we may only apply the Fourier transformation in the temporal dimension on the layer, which contains the subintervals. The layer, which contains the quadrature nodes, remains untouched from the Fourier transformation and, thus, needs to be discretized.

An example for a matrix symbol is the generalization of (2.33) and (2.34), which is done by substituting $2\pi j/L$ with a continuous variable $\theta \in [0, 2\pi)$ in (2.34). This yields the 1×1 matrix symbol $\exp(-i\theta)$ for the operator $\hat{\mathbf{E}}$ with an arbitrary number of subintervals L. In principle, this function also denotes the matrix symbol of the operator $\mathbf{E}$, since, usually, the matrix symbol of an operator is obtained from the diagonalization of its circulant counterpart. The dimension L of the operator is now reflected in how the interval $\theta \in [0, 2\pi)$ is sampled. Hence, the evaluation of the matrix symbol at these sampled points yields the eigenvalues of the operator. This is especially helpful when an upper bound is sought for the euclidean norm of the operator. In this case, a sharp upper bound is given by

$$\left\|\hat{\mathbf{E}}\right\| \leq \sup_{\theta \in [0,2\pi)} \|\exp(-i\theta)\|_2 = 1.$$

In the same sense, we are able state the matrix symbols for a reinterpretation of Theorem 4 in the following

Theorem 6. *Let us apply the conditions of Theorem 4. Further let $\lambda^A(\theta)$, $\lambda^C(\theta)$ and $\lambda^{C'}(\theta)$ be the matrix symbols of the spatial system matrix $\mathbf{A}$, the interpolation-defining matrix $\mathbf{C}$ and the restriction-defining matrix $\mathbf{C}'$. Thus, it yields the matrix symbols for*

the interpolation and restriction operators

$$d(\theta) = \frac{1 + \lambda^{C}(2\theta)\, e^{-i\theta}}{\sqrt{2}}, \qquad \hat{d}(\theta) = \frac{1 - \lambda^{C}(2\theta)\, e^{-i\theta}}{\sqrt{2}},$$
$$f(\theta) = \frac{1 + \lambda^{C'}(2\theta)\, e^{-i\theta}}{\sqrt{2}}, \qquad \hat{f}(\theta) = \frac{1 - \lambda^{C'}(2\theta)\, e^{-i\theta}}{\sqrt{2}},$$

and in the same way for the preconditioning matrices and the system matrix

$$\begin{aligned} \mathbf{B}^{(P)}(\theta) &= \mathbf{I}_L \otimes \mathbf{I}_M - \Delta t \cdot \lambda^{(A)}(\theta)\mathbf{I}_L \otimes \mathbf{Q}_{\mathbf{\Delta}}, \\ \mathbf{B}^{(M)}(\theta) &= \mathbf{I}_L \otimes \mathbf{I}_M - \mathbf{E} \otimes \mathbf{N} - \Delta t \cdot \lambda^{(A)}(\theta)\mathbf{I}_L \otimes \mathbf{Q}, \\ \mathbf{B}^{(\tilde{P})}(\theta) &= \mathbf{I}_L \otimes \mathbf{I}_M - \mathbf{E} \otimes \mathbf{N} - \Delta t \cdot \lambda^{(A)}(2\theta)\mathbf{I}_L \otimes \mathbf{Q}_{\mathbf{\Delta}}. \end{aligned} \tag{2.40}$$

Consequently, these symbols result in

$$\mathcal{B}^{(T)}(\theta) = \mathcal{B}^{(S)}(\theta)\mathcal{B}^{(CGC)}(\theta) \quad \textit{with} \tag{2.41}$$

$$\mathcal{B}^{(S)}(\theta) = \begin{pmatrix} \mathbf{I} - \left(\mathbf{B}^{(P)}(\theta)\right)^{-1}\mathbf{B}^{(M)}(\theta) & \\ & \mathbf{I} - \left(\mathbf{B}^{(P)}(\theta')\right)^{-1}\mathbf{B}^{(M)}(\theta') \end{pmatrix} \tag{2.42}$$

$$\mathcal{B}^{(CGC)}(\theta) = \begin{pmatrix} \mathbf{I} - \left(fd\left(\mathbf{B}^{(\tilde{P})}\right)^{-1}\mathbf{B}^{(M)}\right)(\theta) & -\left(\hat{f}d\left(\mathbf{B}^{(\tilde{P})}\right)^{-1}\right)(\theta)\,\mathbf{B}^{(M)}(\theta') \\ -\left(\hat{d}f\left(\mathbf{B}^{(\tilde{P})}\right)^{-1}\mathbf{B}^{(M)}\right)(\theta) & \mathbf{I} - \left(\hat{f}\hat{d}\left(\mathbf{B}^{(\tilde{P})}\right)^{-1}\right)(\theta)\mathbf{B}^{(M)}(\theta') \end{pmatrix}, \tag{2.43}$$

the matrix symbols of PFASST. These matrix symbols produce the time collocation blocks, when evaluated at the points $\theta_k = 2\pi k/N$ for $k = 0, \ldots, N-1$.

Proof. The first part of the theorem is only the definition of the emerging matrix symbols. Only the statement that the evaluation at the points $\theta_k = 2\pi k/N$ yields the time collocation blocks needs proof, which is just a series of straightforward computations. □

We also introduce the matrix symbols related to the collocation blocks in the following theorem, which has a similar proof as Theorem 6.

Theorem 7. *Let us apply the conditions of Theorem 5. Further let $\lambda^{A}(\theta)$, $\lambda^{C}(\theta)$ and $\lambda^{C'}(\theta)$ be the matrix symbols of the spatial system matrix $\mathbf{A}$, the interpolation defining matrix $\mathbf{C}$ and the restriction defining matrix $\mathbf{C}'$. Thus, it yields the matrix symbols for*

the interpolation and restriction operators

$$d(\theta) = \frac{1 + \lambda^C(2\theta)\, e^{-i\theta}}{\sqrt{2}}, \qquad \hat{d}(\theta) = \frac{1 - \lambda^C(2\theta)\, e^{-i\theta}}{\sqrt{2}}$$
$$f(\theta) = \frac{1 + \lambda^{C'}(2\theta)\, e^{-i\theta}}{\sqrt{2}}, \qquad \hat{f}(\theta) = \frac{1 - \lambda^{C'}(2\theta)\, e^{-i\theta}}{\sqrt{2}},$$

and in the same way for the preconditioning matrices and the system matrix

$$\begin{aligned} \mathbf{B}^{(P)}(\theta_x) &= \mathbf{I}_M - \Delta t \cdot \lambda^{(A)}(\theta_x)\mathbf{Q}_{\mathbf{\Delta}}, \\ \mathbf{B}^{(M)}(\theta_x, \theta_t) &= \mathbf{I}_M - \exp(-i\theta_t)\mathbf{N} - \Delta t \cdot \lambda^{(A)}(\theta_x)\mathbf{Q}, \\ \mathbf{B}^{(\tilde{P})}(\theta_x, \theta_t) &= \mathbf{I}_M - \exp(-i\theta_t)\mathbf{N} - \Delta t \cdot \lambda^{(\tilde{A})}(2\theta_x)\mathbf{Q}_{\mathbf{\Delta}}. \end{aligned} \tag{2.44}$$

Consequently, these symbols result in $\mathcal{B}^{(T)}(\theta) = \mathcal{B}^{(S)}(\theta)\mathcal{B}^{(CGC)}(\theta)$*, with* $\mathbf{\Theta} = (\theta_x, \theta_t)$*,* $\mathbf{\Theta}' = (\theta_x', \theta_t)$ *and*

$$\mathcal{B}^{(S)}(\mathbf{\Theta}) = \begin{pmatrix} \mathbf{I} - \left(\mathbf{B}^{(P)}(\theta_x)\right)^{-1} \mathbf{B}^{(M)}(\mathbf{\Theta}) & \\ & \mathbf{I} - \left(\mathbf{B}^{(P)}(\theta_x')\right)^{-1} \mathbf{B}^{(M)}(\mathbf{\Theta}') \end{pmatrix} \tag{2.45}$$

$$\mathcal{B}_k^{(CGC)}(\mathbf{\Theta}) = \begin{pmatrix} \mathbf{I} - \left(f d \left(\mathbf{B}^{(\tilde{P})}\right)^{-1} \mathbf{B}^{(M)}\right)(\mathbf{\Theta}) & -\left(\hat{f} d \left(\mathbf{B}^{(\tilde{P})}\right)^{-1}\right)(\mathbf{\Theta})\, \mathbf{B}^{(M)}(\mathbf{\Theta}') \\ -\left(f \hat{d} \left(\mathbf{B}^{(\tilde{P})}\right)^{-1} \mathbf{B}^{(M)}\right)(\mathbf{\Theta}) & \mathbf{I} - \left(\hat{f} \hat{d} \left(\mathbf{B}^{(\tilde{P})}\right)^{-1}\right)(\mathbf{\Theta}) \mathbf{B}^{(M)}(\mathbf{\Theta}') \end{pmatrix}, \tag{2.46}$$

the matrix symbols of PFASST with the assumption of periodicity in time. These matrix symbols produce the collocation blocks, when evaluated at the points $\theta_k = 2\pi k/N$ *for* $k = 0, \ldots, N-1$ *and* $\theta_j = 2\pi k/L$ *for* $j = 0, \ldots, L-1$*.*

We stated the matrix symbols in this work for reasons of completeness, since this is the form which appears in standard multigrid theory. By using matrix symbols, we are able to state convergence estimates independently from the number of grid points in space. A supremum over θ of norms or spectral radii yields an upper bound, which is independent of the number of grid points N used. Another potential application for matrix symbols is the simplified way of observing the effect of the operators on vectors with a certain structure. We investigate this simplified way in the following section.

2.3.2. Structure of the error vector

For space-time multigrid methods it is reasonable to write the vector as a Kronecker product between a spatial and temporal vector, as we find it in the following definition.

Definition 4. *Let the spatial problem with N degrees of freedom be diagonalizable, then let the eigenvectors be denoted as $\boldsymbol{v}_k$ for $k \in \{1, \dots, N\}$. Furthermore, let $j = M \cdot l + m$ be the index for the m-th quadrature node of the l-th subinterval, with $l \in \{0, \dots, L-1\}$ and $m \in \{0, \dots, M-1\}$, then s_k denotes the weight at the k-th time point and $\boldsymbol{w}$ denotes the vector containing this weights. Altogether we define*

$$\mathcal{E} = \{\boldsymbol{v}_k \otimes \boldsymbol{s} | k \in \{1, \dots, N\} \text{ and } \boldsymbol{s} \in \mathcal{S}\}, \text{ where } \mathcal{S} \text{ is a basis of } \mathbb{C}^{LM}, \tag{2.47}$$

as a set of basis vectors of the error space. Depending on the basis vector $\boldsymbol{s}_j \in \mathcal{S}$, we denote

$$\boldsymbol{e}_{k,j} = \boldsymbol{v}_k \otimes \boldsymbol{s}_j \in \mathcal{E}$$

*as the k, j-th **error basis vector**. Additionally, we denote the canonical basis of $\mathbb{C}^{LM}$ as $\mathcal{C}$ and its basis vectors as $\boldsymbol{b}_j$.*

Remark 4. *Note that $\mathcal{E}$ is a basis for $\mathbb{K}^{NML}$. Therefore, every possible error may be represented as a linear combination of **error basis vectors**. Furthermore, the construction of the vector $\boldsymbol{v}_j$ depends on the type and dimension of the spatial problem. For one dimensional problems with periodic boundaries on an equidistant mesh, we obtain the set $\{\boldsymbol{v}_j\}_{j \in \{1,\dots,N\}}$ by evaluating the function $\exp(i\theta k)$ for $k \in \{0, \dots, N-1\}$ for different $\theta \in [0, 2\pi)$. Once again, finite dimensionality is regained by using a sampled subset of discrete points from the interval $\theta = 2\pi \frac{k}{N} \in [0, 2\pi)$.*

The structure of an error basis vector acts jointly with the transformations. This leads to the time collocation blocks and therefore also to the corresponding matrix symbols. Exactly this structure is the foundation of the following lemma.

Lemma 5. *Let $\boldsymbol{e}_{k,j}$ be an error vector as defined in 4, $\mathcal{B}(\theta)$ one of the matrix symbols defined in* (2.41), (2.42) *or* (2.43) *and let N be the degree of freedom in space, then it holds*

$$\|\mathbf{T} \cdot \boldsymbol{e}_{k,j}\|_2 = \begin{cases} \left\| \mathcal{B}\left(\frac{\pi k}{N}\right) \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \boldsymbol{w}_j \right\|_2 & , \text{ for } k < \frac{N}{2}, \\ \left\| \mathcal{B}\left(\frac{\pi k}{N} - \frac{\pi}{2}\right) \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \boldsymbol{w}_j \right\|_2 & , \text{ for } k \geq \frac{N}{2}. \end{cases} \tag{2.48}$$

Proof. We use the same transformation $\mathcal{F}$, which was used to construct the time collocation blocks. For an arbitrary basis $\mathcal{S}$ in time, it holds $\mathcal{F}^T \cdot \boldsymbol{e}_{k,j} = \boldsymbol{v}_k \otimes \boldsymbol{s}_j$. Together

with the permutation operator

$$\mathcal{P}\boldsymbol{b}_k \otimes \boldsymbol{s} = \begin{cases} \boldsymbol{b}_{2k-1} \otimes \boldsymbol{s} & \text{, for } k < N/2 \\ \boldsymbol{b}_{2(k-N/2)} \otimes \boldsymbol{s} & \text{, for } k \geq N/2, \end{cases},$$

it holds

$$\begin{aligned} \mathcal{P}^{-1}\mathcal{F}^{-1}\mathbf{T}\mathcal{F}\mathcal{P}\mathcal{P}^{-1}\mathcal{F}^{-1}\boldsymbol{e}_{k,j} &= \mathcal{P}^{-1}\mathcal{F}^{-1}\mathbf{T}\mathcal{F}\mathcal{P}\mathcal{P}^{-1}\boldsymbol{b}_k \otimes \boldsymbol{s}_j \\ &= \begin{cases} \operatorname{diag}\left(\mathcal{B}_k^{(T)}\right) \cdot \boldsymbol{b}_{2k-1} \otimes \boldsymbol{s}_j & \text{, for } k < N/2, \\ \operatorname{diag}\left(\mathcal{B}_k^{(T)}\right) \cdot \boldsymbol{b}_{2(k-N/2)} \otimes \boldsymbol{s}_j & \text{, for } k \geq N/2, \end{cases} \\ &= \begin{cases} \mathcal{B}_k^{(T)} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \boldsymbol{s}_j & \text{, for } k \leq N/2, \\ \mathcal{B}_{k-N/2}^{(T)} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \boldsymbol{s}_j & \text{, for } k \leq N/2. \end{cases} \end{aligned}$$

The operator $\mathcal{P}^{-1}\mathcal{F}^{-1}$ is orthogonal, consequently it follows

$$\left\| \mathcal{P}^{-1}\mathcal{F}^{-1}\boldsymbol{e}_{k,j} \right\|_2 = \left\| \boldsymbol{e}_{k,j} \right\|_2 .$$

□

This lemma will assist the numerical experiments conducted in the next chapter. It also gives insight into the mechanism of mode mixing. In our case we denote mode mixing, as the property that e.g., error modes $\boldsymbol{e}_{k,j}$ associated with a frequency $\theta_k \in \Theta_{\text{left}}$ result in a linear combination of two error modes after one PFASST step. This effect is associated with the idea of harmonic pairs.

3. Convergence study

> Das Experiment, dem nicht eine Theorie, d.h. eine Idee vorausgeht, verhält sich zur Naturforschung wie das Rasseln einer Kinderklapper zur Musik.
>
> Justus Freiherr von Liebig

Building upon the theoretical basis derived in the previous chapters, we are now able to sketch a more complete picture of PFASST. This picture will contain three basic aspects: the problems we would like to solve, the properties we would like to observe and finally the parameters of PFASST. Each aspect itself holds a huge number of possible experiments and may be studied extensively.

We will confine ourselves to two basic problems: the diffusion and the advection problems in one dimension; each of which is discretized by a simple finite difference scheme. Thus, two frequent elements found in physical systems, are covered.

As for the properties, we observe the spectral radii of the iteration matrix, effects on Fourier modes, and later, the approximation and smoothing property. These properties originate from multigrid theory. In the field of parallel computing, the parallel efficiency and the speedups are usually of interest. In the theory of time-stepping methods, the observation of stability fields takes an important role. This gives us even more interesting properties, we could study. Here, we also confine ourselves to a basic subset.

Not only the property itself, but also the methods to measure the property have to be chosen with care, in order to sketch a complete picture of PFASST. For example, the computation of the eigenvalues turns out to be difficult for some cases, as we will elaborate in Section 3.1.

The third aspect is a change of the algorithm itself, which also harbors plenty of possibilities. For example, we may use a different coarsening strategy, different solvers in space, different transfer operators, or variations of the SDC method.

When each aspect is confined to a manageable extent, we are able to focus on the interaction between the three aspects, thus gain a complete picture and thereby a better understanding of PFASST.

As first step, we introduce the two model problems, which have been known to mathematicians for centuries. Take for instance the Poisson equation [81], discovered by

Siméon Denis Poisson and published in 1813. It is used in physics to describe gravitational and electrostatic laws. It is also the ideal model problem to explain the success of multigrid methods [54]. The heat or diffusion parabolic equation is a time dependent version of the Poisson equation and is equally well understood [82]. Many efficient numerical methods are known and depending on the boundary conditions it is possible to state an analytical solutions. This makes it a natural candidate for the analysis of a multigrid-like time integration method like PFASST. The problem in one spatial dimension is given by

$$\begin{aligned} u_t &= \nu \Delta u, \quad x \in [0,1] \text{ and } t \in [0,T] \\ u(x,0) &= u_0(x), \quad u(0,t) = u(1,t), \quad t \in [0,T], \end{aligned} \tag{3.1}$$

for an end time point $T > 0$ and the diffusion coefficient $\nu > 0$. We use periodic boundaries, which simplifies the notation but does not change the behavior of the equation. To keep the discretization simple, we use a second-order finite differences on a isometric grid it holds

$$X = [x_1, \ldots, x_N], \text{ with } x_j = \frac{j-1}{N} \text{ and } \Delta x = \frac{1}{N}, \tag{3.2}$$

which leads to a system of linear ODEs

$$\begin{aligned} U_t(t) &= \frac{\nu}{\Delta x^2} \mathbf{A}\, U(t), \quad t \in [0,T] \text{ and } U(0) = [u(x_1,0), \ldots, u(x_N,0)], \\ \text{with} \quad \mathbf{A} &= \begin{pmatrix} 2 & -1 & 0 & \cdots & -1 \\ -1 & 2 & -1 & & \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & -1 & 2 & -1 \\ -1 & 0 & \cdots & -1 & 2 \end{pmatrix} \in \mathbb{R}^{N \times N}. \end{aligned} \tag{3.3}$$

When this ODE is solved by a time-stepping method, the spacing in the temporal and spatial dimension becomes an important indicator for the numerical behavior. Therefore, we denote this indicator in form of

$$\mu = \nu \frac{\Delta t}{\Delta x^2}$$

as the dispersion relation number (**DRN**) of the heat equation. At this point, the periodic boundary conditions already pays off, because $\mathbf{A}$ becomes a circulant matrix. This way, the spectral decomposition into eigenvalues and eigenvectors is easily computed.

For the eigenvalues λ_k and normal eigenvectors ψ_k, $k \in \{0, \ldots, N-1\}$, we get

$$\lambda_k = 4\sin^2\left(\frac{k\pi}{N}\right) \quad \text{and} \quad \psi_k = \frac{1}{\sqrt{N}}\left[\exp\left(i\frac{2\pi}{N}k \cdot 0\right), \ldots, \exp\left(i\frac{2\pi}{N}k \cdot (N-1)\right)\right]^T. \tag{3.4}$$

The second model problem is the 1D advection equation, given by

$$\begin{aligned} u_t = cu_x, \quad & x \in [0,1] \text{ and } t \in [0,T] \\ u(x,0) = u_0(x), \quad & u(0,t) = u(1,t) \quad t \in [0,T], \end{aligned} \tag{3.5}$$

with a constant speed $c > 0$. As a result of conservation laws, it mostly appears as a part of more involved fluid dynamics equations [83]. It is classified as a hyperbolic PDE. This class sparked many new mathematical methods, e.g., the finite volume methods [84].

The discretization is done in the same manner as (3.3), but instead of a central difference stencil we use an upwind scheme of the order 2, which introduces a predominant direction in the numerical method itself. By defining

$$c^+ = \max(c, 0)\,, \qquad c^- = \min(c, 0)$$

and

$$u_x^- = \frac{u_i^n - u_{i-1}^n}{\Delta x}\,, \qquad u_x^+ = \frac{u_{i+1}^n - u_i^n}{\Delta x},$$

both directions of the advection are considered. These two conditional equations above, when combined, yield

$$u_i^{n+1} = u_i^n - \Delta t\left[c^+ u_x^- + c^- u_x^+\right]. \tag{3.6}$$

These upwind schemes are stable, as long as the Courant-Friedrichs-Lewy (CFL) condition

$$\left\|\frac{c\Delta t}{\Delta x}\right\| \leq 1 \tag{3.7}$$

is fulfilled, cf. [85].

However, due to the constant speed c, we may formulate the part of (3.6), which describes the derivative in space, in matrix form. For periodic boundary conditions it

yields a circulant matrix

$$\mathbf{F}_D = \begin{pmatrix} -3 & 4 & -1 & \cdots & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & \cdots & -3 & 4 & -1 \\ -1 & \vdots & 0 & -3 & 4 \\ 4 & -1 & \cdots & 0 & -3 \end{pmatrix} \in \mathbb{R}^{N \times N},$$

with eigenvalues

$$\lambda_k = \exp\left(i\frac{2\pi}{N}k\right)\left(-3 + 4\exp\left(i\frac{2\pi}{N}\right) - \exp\left(i\frac{4\pi}{N}\right)\right), \text{where } k \in \{0, \ldots, N-1\}.$$

This matrix leads to a system of linear ODEs

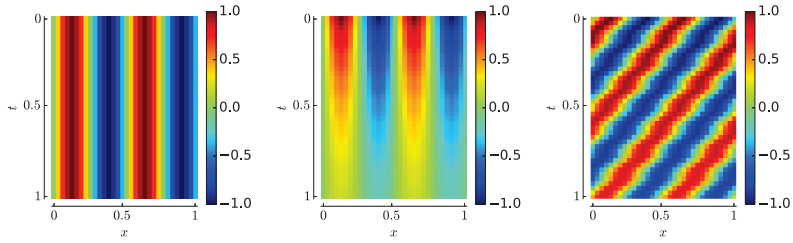
$$U_t(t) = \frac{c}{2\Delta x}\mathbf{F}_D\, U(t), \quad t \in [0, T] \quad \text{and} \quad U(0) = [u(x_1, 0), \ldots, u(x_N, 0)].$$

When this ODE system is solved by an explicit Euler scheme, it yields one of the aforementioned upwind schemes. However, we are interested in the properties of PFASST and therefore will employ it to solve this system of linear ODEs implicitly. As for the first model problem, it holds for the second model problem that the numerical behavior is strongly influenced by an according dispersion relation number. For the advection problem this is the CFL number from Eq. (3.7). With these two model problems and a set of particular experiments we begin our study, which will serve as an appropriate starting point for the following experiments.

3.1. First experiments

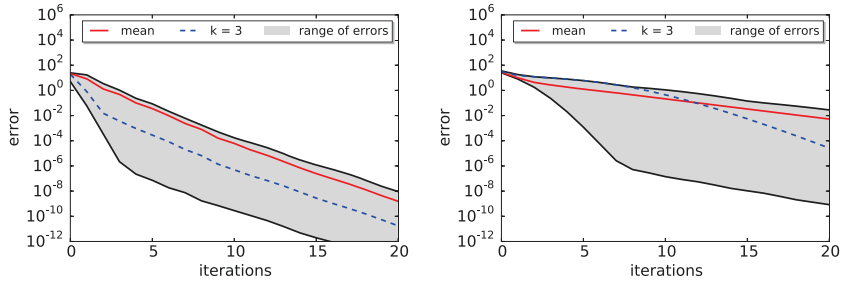
In our first experiments, we use a typical setup for the diffusion and advection problem with an identical decomposition of the space-time domain. We use 8 subintervals with 5 Gauß-Radau quadrature nodes to decompose the time domain $[0, 1]$. In space, we use 32 discretization points for the space domain $[0, 1]$ and an interpolation operator with the order of 6, and injection for the restriction operator. An interpolation order of 6 means that polynomials with the degree up to 6 are interpolated exactly. These transfer operators are used to pass information between the fine and coarse grid. The coarse grid uses the same discretization in the temporal dimension and half of the points in the spatial dimension. On each level, an exact solver is used in space. This could, for example, be a fully converged multigrid in space. For our purposes, it is sufficient to invert the system matrix $\mathbf{A}$ and $\mathbf{F}_D$ respectively. As the starting iteration value, we take the initial value of the ODE and spread it over all time nodes.

In Figure 3.1 we see that the chosen parameters lead to non-pathological setups. This



(a) inital distribution (b) diffusion (c) advection

Figure 3.1.: Initial distribution, generated by evaluating $\sin(4\pi x)$ and spreading it on every time node, and the result of 10 PFASST iterations for the diffusion problem with $\mu = 10^{-2}$ and the advection problem with $c = 1$.



(a) diffusion (b) advection

Figure 3.2.: Errors over iterations for the diffusion problem with $\mu = 10^{-2}$ and advection with $c = 1$. The gray area indicates possible errors resulting from different initial values of the form $\sin(2\pi k x)$ for $k \in \{1, \dots, 15\}$.

means that the choice neither leads to strong numerical diffusion in Figure 3.1c, which sometimes appears for advective problems, nor that the choice of μ leads to vanishing or almost constant results for the diffusion problem, as shown in Figure 3.1b.

Of course, these are only two examples out of an infinite number of possibilities, nonetheless they are an excellent starting position to observe the parameter space systematically. Therefore, our first step is to observe the convergence behavior for different initial values of the form $\sin(2\pi kx)$ for $k \in \{1, \dots, 15\}$. In Fig. 3.2, instead of plotting the error curve for each initial value, we mark the maximal and minimal errors for each iteration with a grey area. In addition, the mean and one example error curve are plotted. We see a gap of the magnitude of 6 to 8 between the best and worst scenario for both problems, showing the crucial influence of the initial values on the convergence behavior. The mean curve shows that most initial values result in an error curve near the maximum. PFASST works significantly better for the diffusion problem than the advection problem, which is observed quite often for other parallel-in-time algorithms, see e.g., [25, 86, 87]. The results we have presented up until now are *a posteriori* and were produced without using the theory from Chapter 2. However, by introducing the theory, we are able to give *a priori* results. Another advantage is the precise control over the spatial problem, which enables an observation in finer nuances. More precisely, we are now able to study the different properties for each fundamental mode in space.

3.1.1. Four strategies

The theory discussed in Chapter 2 gives us the iteration matrix of PFASST and its constituents. We consider the following four strategies, in order to estimate the error vector of the κ-th iteration $\boldsymbol{e}^\kappa$, for which the iteration matrix plays a main role. The first strategy is based on the inequality of consistent matrix norms $\|\cdot\|$ for each $\kappa \in \mathbb{N}$

$$\rho(A) \leq \|A^\kappa\|^{\frac{1}{\kappa}}, \quad \text{and} \quad \|A^\kappa\|^{\frac{1}{\kappa}} \to \rho(A) \text{ for } \kappa \to \infty, \tag{3.8}$$

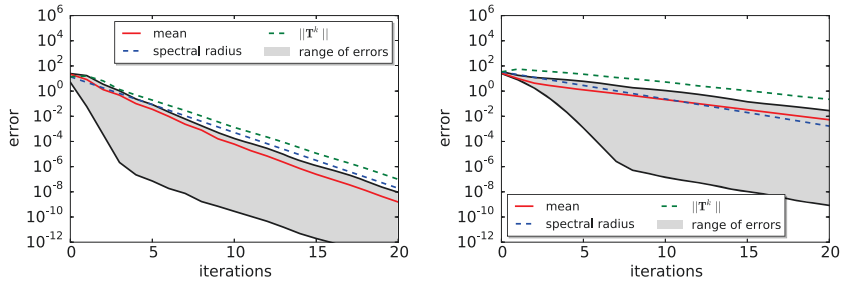
cf. [53].

Note that by using the spectral radius, neither an upper nor lower bound is given for the error of each iteration, rather one gets an asymptotic estimation. In contrast, strategies 2 and 3 rely on the inequality

$$\|\boldsymbol{e}^\kappa\| = \left\|\mathbf{T}^\kappa \boldsymbol{e}^0\right\| \leq \|\mathbf{T}^\kappa\| \left\|\boldsymbol{e}^0\right\| \leq \|\mathbf{T}\|^\kappa \left\|\boldsymbol{e}^0\right\|. \tag{3.9}$$

and therefore give an upper bound for the norm of the error at each iteration. Additionally, the iteration matrix is separated from the initial error vector $\boldsymbol{e}^0$, so that an *a priori* estimation of the relative error reduction is possible for these strategies.

When we apply these strategies to the results depicted in Fig. 3.2, we observe, as Figure 3.3 indicates, Strategy 1 and 3 are capable of estimating the mean convergence behavior of PFASST for the advection and diffusion problem. Since the 2-norm of $\|\mathbf{T}\|$



(a) diffusion (b) advection

Figure 3.3.: Errors over iteration for diffusion with $\mu = 10^{-2}$ and advection with $c = 1$ in comparison to the error estimation of Strategy 1 and 3.

for the diffusion problem is 1.148 and 1.665 respectively for the advection problem. This renders Strategy 2 useless to estimate the convergence behavior of PFASST.

To complete the number of strategies, we count the application of the transformed iteration matrix κ times to the spatial modes of the error vector as the fourth strategy. In short, we summarize the strategies as follows:

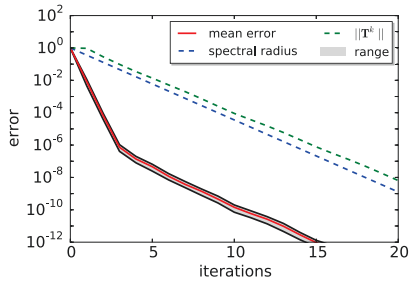
1. use the spectral radius $\rho(\mathbf{T})$,
2. use the norm $\|\mathbf{T}\|$,
3. use the norm $\|\mathbf{T}^\kappa\|$,
4. compute the norm $\left\|\mathbf{T}^\kappa \boldsymbol{e}^0\right\|$.

In contrast to the other strategies, Strategy 4 requires the initial error vector $\boldsymbol{e}^0$ to be known, making it an *a posteriori* strategy. If time collocation blocks are used, the computation following Strategy 4 yields the analytically correct error norm for each iteration.

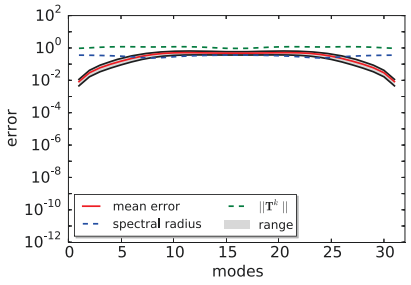
However, the error is usually not known. Thus, instead of taking only one error into account, we partially randomize the error vector and use a stochastic perspective. Therefore, we resort to the results in Section 2.3.1. In particular, we will use the Definition 4 of a particular error vector

$$\boldsymbol{e}_k = \boldsymbol{m}_k \otimes \boldsymbol{w},$$

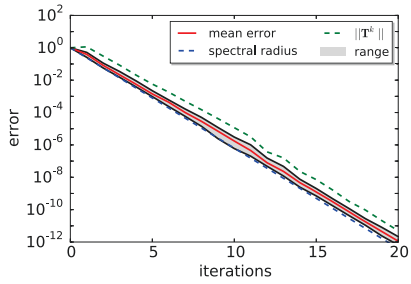
where $\boldsymbol{m}_k$ is a Fourier mode and $\boldsymbol{w}$ is an arbitrary error vector of the temporal dimension. Furthermore, we rely on the relation of the iteration matrices in block form and the



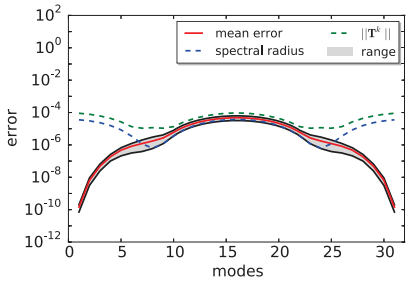
(a) $j = 1$



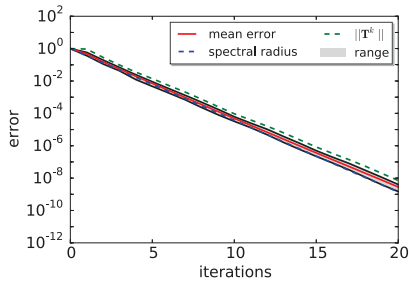
(b) $\kappa = 1$



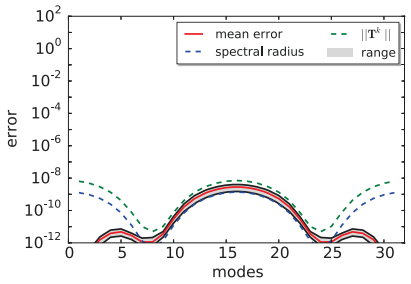
(c) $j = 8$



(d) $\kappa = 10$

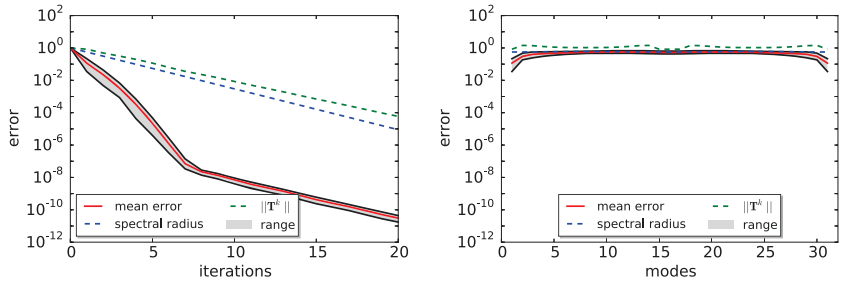


(e) $j = 16$



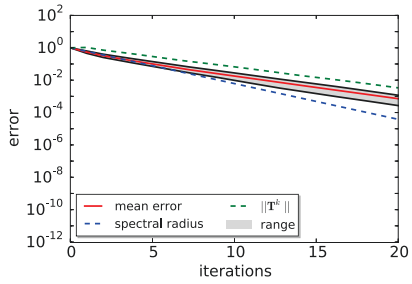
(f) $\kappa = 20$

Figure 3.4.: Error over iteration for the diffusion problem with $\mu = 10^{-2}$, a randomized error in time and certain spatial modes $\boldsymbol{m}_j$ on the left. Respectively, error over modes for a certain iteration κ on the right. The gray area indicates possible errors resulting from the randomization in the temporal dimension.

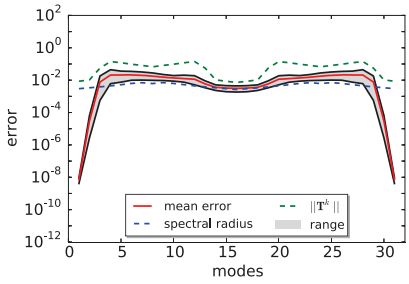


(a) $j = 1$

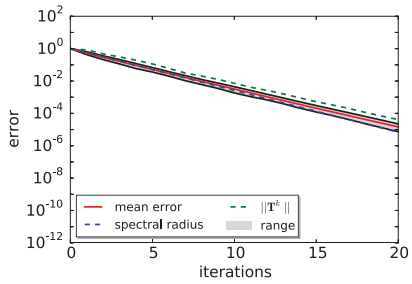
(b) $\kappa = 1$



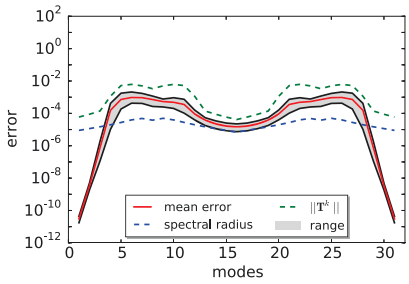
(c) $j = 8$



(d) $\kappa = 10$



(e) $j = 16$



(f) $\kappa = 20$

Figure 3.5.: Error over iteration for the advection problem with $c = 1$, a randomized error in time and certain spatial modes $\boldsymbol{m}_j$ on the left. Respectively, error over modes for a certain iteration κ on the right. The gray area indicates possible errors resulting from the randomization in the temporal dimension.

particular error vectors, worked out in Lemma 5. Due to the special structure of the error vectors, we have control over the spatial dimension and only use randomized and normalized vectors with real entries for $\boldsymbol{w}$.

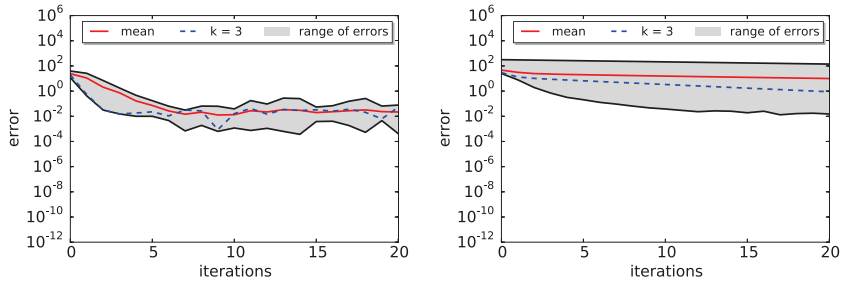
This control also gives us the possibility to use all strategies for separate pairs of harmonic modes, by computing the spectral radii and norms of the blocks instead of the whole iteration matrix. Figures 3.4 and 3.5 depict this control for the diffusion problem and for the advection problem, respectively. There are more observations to be made. For every mode and iteration, the range of errors is in the same order of magnitude, which indicates that the influence of the temporal part of the error is mild. As expected, Strategy 3 gives an upper bound and the spectral radius is only an asymptotic estimation, see, for example, Fig. 3.5f, where the spectral radius clearly lies over or under the mean error.

Of greater interest is that the connection between the harmonic modes are easily seen when errors are plotted over the modes. Each time collocation block of the iteration matrix represents two modes. Therefore, when Strategy 1 to 3 are used to estimate the error of a certain mode, the harmonic mode is inevitably considered as well. While modes with lower frequencies are reduced by PFASST more efficiently, the first three strategies do not predict this, since every low frequency mode has a harmonic mode with a higher frequency. This is a vital distinction to Strategy 4 and also the main reason, together with the accuracy of Strategy 4 for each mode and iteration, to lay the focus on Strategy 4 in the following sections.

Nonetheless, for an *a priori* worst case estimation, a combination of the spectral radius and the norm of the iteration matrix is suitable. Therefore, it is of interest to find efficient ways to compute the spectral radius and norms of the iteration matrices. Using time collocation blocks is a first step, and the use of collocation blocks is the natural second step. In the following section, we explore this second step.

3.1.2. Collocation and time collocation blocks

The main difference between collocation and time collocation blocks is the use of different matrices $\mathbf{E}$, see Section 2.2.4. This modification can be applied to the formulation of PFASST without the decomposition into blocks. With this modification, the collocation blocks are an accurate reformulation of the iteration matrix and not just an estimation. In Figure 3.6, the convergence behavior of this modified time periodic version of PFASST is shown and at first glance it appears to be an inferior idea, when compared with the error plots in Fig. 3.2. From the theory of circulant and Toeplitz matrices [88], we know that under certain conditions the spectrum of the Toeplitz matrices converge to the spectrum of the right circulant counterpart for an increasing size of the matrices. Therefore, we assume that with more subintervals, the convergence behavior becomes increasingly similar. We test this hypothesis by comparing the spectral radius, which has shown to be a possible estimator for the convergence behavior for an exponentially



(a) diffusion (b) advection

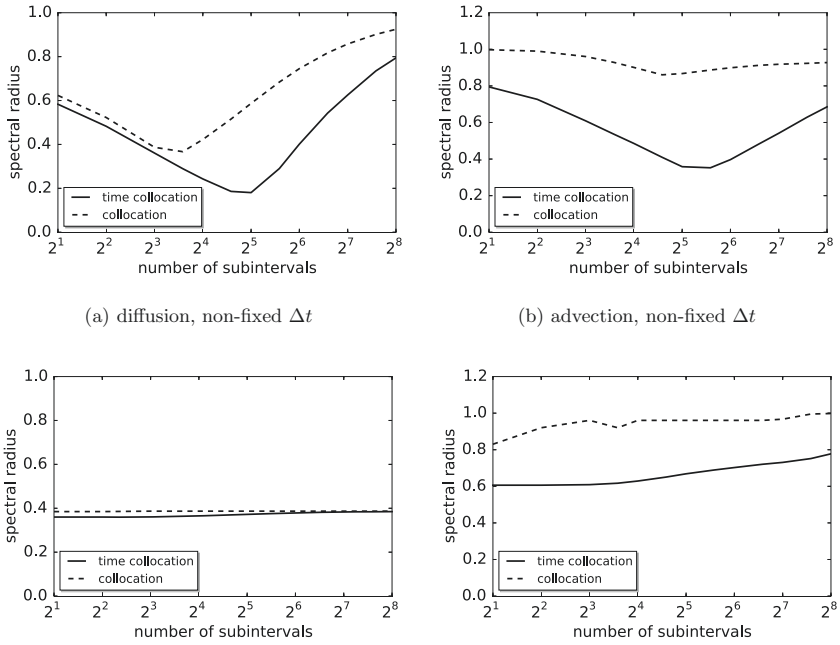
Figure 3.6.: Errors over iteration for diffusion with $\mu = 10^{-2}$ and advection with $c = 1$, using a time periodic version of PFASST. The gray area indicates possible errors resulting from different initial values of the form $\sin(2\pi k x)$ for $k \in \{1, \dots, 15\}$.

increasing number of subintervals and a fixed time domain. Figure 3.7 depicts this for our example and reveals that the asymptotic convergence of $\mathbf{E}$ to $\hat{\mathbf{E}}$ does not necessarily imply the asymptotic convergence of iteration matrices $\mathbf{T}$ to $\hat{\mathbf{T}}$ for an increasing number of subintervals and a fixed time domain. In other words, asymptotic convergence could be observed for more than 256 subintervals, but setups with more than 256 subintervals are not common and therefore not of interest. Another reason could be that the computation of the spectral radius itself is instable, as we will elaborate in Section 3.1.3.

However, from Figure 3.7 we also learn that the number of subintervals used has a significant influence on the expected worst case convergence behavior. This is due to the change of the numerical dispersion relation, which changes with Δt. When the dispersion relation is changed, the numerical problem we attempt to solve changes and the effect of the asymptotic convergence between the two vanishes. Note that the effect of the dispersion relation is discussed more thoroughly in Section 3.2.

The situation changes when the length of subintervals is fixed to Δt. Then, the physical problem is solved on the time domain $[0, \Delta t \cdot L]$, but the dispersion relation number and hence the numerical properties are similar for different number of subintervals. In Figure 3.7c and 3.7d, at least for the diffusion problem, we observe that the spectral radii approach each other.

In the first instance, one would conclude that the use of collocation blocks is not suited as a general strategy, because it is not feasible to check for every new problem and setup, if the use of collocation blocks leads to acceptable estimations. The only exceptions are setups with a sufficiently high number of subintervals, where the computation becomes



(a) diffusion, non-fixed Δt

(b) advection, non-fixed Δt

(c) diffusion, fixed Δt

(d) advection, fixed Δt

Figure 3.7.: Comparison between spectral radii, computed by collocation or time collocation blocks, for the diffusion problem with $\mu = 10^{-2}$ and the advection problem with $c = 1$, for different numbers of subintervals and fixed $\Delta t = 1/8$ or non-fixed $\Delta t = 1/L$, where L is the number of subintervals.

too expensive, when time collocation blocks are used.

However, we obtain contradictory results when this conclusion is checked against Figure 3.8, where spectral radii of the collocation blocks are compared to the actual errors of a setup with 128 subintervals. In detail, we observe that the spectral radii of the collocation blocks give a better estimation of the actual error, compare e.g., Fig. 3.8a to Fig. 3.8c. As aforementioned, one reason could be the computation of the spectral radii of the time collocation blocks, which becomes difficult and unreliable. In the following section we introduce pseudospectra in order to study this effect in more detail.

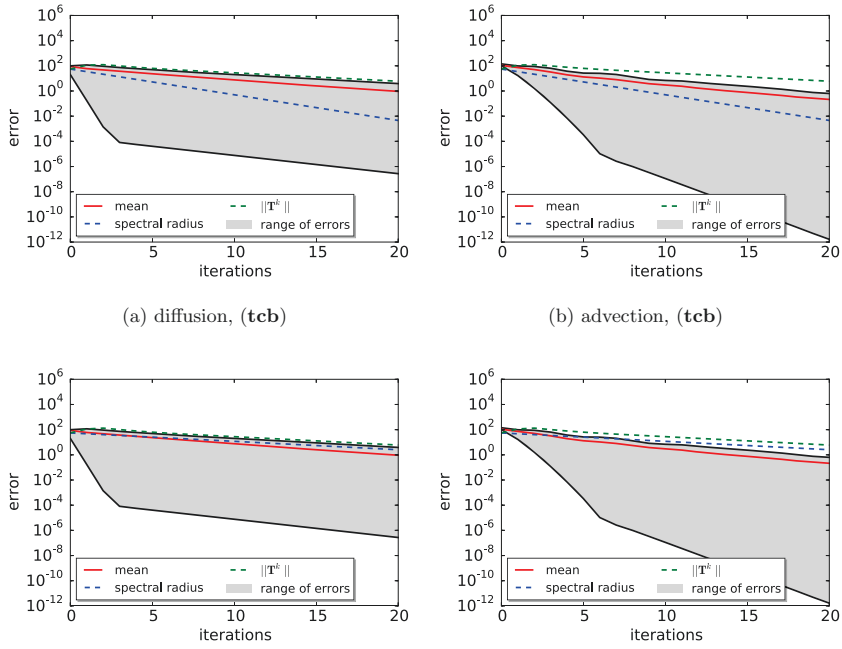
3.1.3. Pseudospectra

In Section 3.1.1, we argued that the spectral radius is a valid indicator to estimate the mean convergence behavior for both problems. This gradually changes when the number of subintervals increases. Figure 3.8 shows that the spectral radius clearly fails to estimate the convergence behavior for this case. The root of this problem is that the computation of the spectral radius is sometimes sensitive to a change in the entries of the underlying matrix. To measure this sensitivity, we use the ε-pseudospectrum of a matrix $\mathbf{A}$, which consists of all eigenvalues of matrices, which are ε-close to $\mathbf{A}$:

$$\Lambda_\epsilon(\mathbf{A}) = \{\lambda \in \mathbb{C} \mid \exists \boldsymbol{x} \in \mathbb{C}^n \setminus \{0\}, \exists \mathbf{F} \in \mathbb{C}^{n\times n} \colon (\mathbf{A}+\mathbf{F})\boldsymbol{x} = \lambda \boldsymbol{x}, \|\mathbf{F}\| \leq \epsilon\}. \tag{3.10}$$

This was first introduced under a different name by Henry Landau in [89] and independently later under different names by different authors, cf. [90–92].

In our case, matrix $\mathbf{F}$ models the numerical accuracy with which we may express the entries in the matrix $\mathbf{A}$. Assume we have computed the eigenvalues λ, we plot this eigenvalue on the complex plane and use the algorithms described in [93–95] to compute $\Lambda_\epsilon(A)$ for a certain ε, e.g. the machine epsilon. The main core of those algorithms is the computation of minimum singular values of $z\mathbf{I} - \mathbf{A}$ for every z in the region of interest. One alternative are path finding algorithms, where one point on the boundary of a desired pseudospectrum is found and then the boundary is traced by finding constant minimum singular values near z, which then gives the next z. When plotted, these sets surround the computed eigenvalues and their extent reveals how sensitive the computation of each eigenvalue is to a disturbance by a matrix $\mathbf{F}$, with $\|\mathbf{F}\| \leq \varepsilon$. Note that these spectral portraits yield a grayscalemap where each shade of gray represents an ε and therefore marks the boundary of a ε-pseudospectrum. Indeed, it holds $\Lambda_\epsilon(A) \subset \Lambda_{\hat{\epsilon}}(A)$ for $\epsilon \leq \hat{\epsilon}$, which enables the representation of the various ε-pseudospectra in one picture. The computation of the pseudo spectra is costly, especially in our case with $L = 128$ the computation for the whole iteration matrix with an acceptable resolution would take several weeks on a desktop computer. Therefore, we compute the pseudospectrum for just one time collocation block, knowing that it represents part of the whole spectrum of the iteration matrix. For one block, we already observe in Figure 3.9b and 3.9d the



(a) diffusion, (**tcb**)

(b) advection, (**tcb**)

(c) diffusion, (**cb**)

(d) advection, (**cb**)

Figure 3.8.: Error over iterations for the diffusion problem with $\mu = 10^{-2}$ and the advection problem with $c = 1$ with a decomposition of the time domain $[0, 1]$ into $L = 128$ subintervals. The gray area indicates possible errors resulting from different initial values of the form $\sin(2\pi kx)$ for $k \in \{1, \ldots, 15\}$. The errors are compared to the estimations of strategy one and three, using collocation (**cb**) or time collocation blocks (**tcb**).

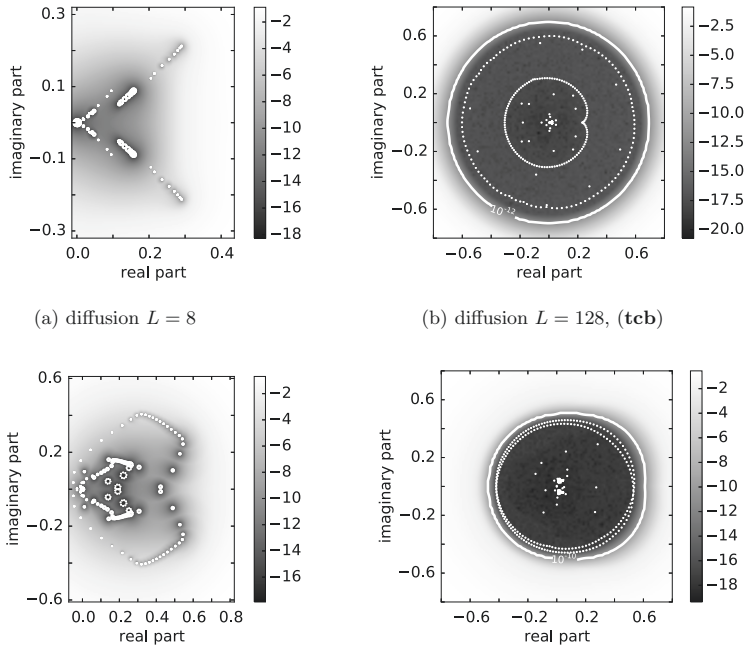
difficulty of computing the spectrum for large L in comparison to the case with $L = 8$. The eigenvalues represented by the white dots are surrounded by a dark grey area, representing the pseudospectrum with the respective ε, which are in the range of the machine accuracy.

Large dark areas mean that the actual position of the eigenvalues found in this area is uncertain. This uncertainty affects the observation of the spectral radius. However, this problem could be fixed by a lower machine epsilon for a certain number of subintervals. At the same time, for an increasing number of subintervals an even lower machine epsilon is needed. This also puts into question the results depicted in Figure 3.7. We may not be certain that the computed spectral radii are accurately computed for increasing L. But on the other hand, this is where the collocation blocks have an advantage over time collocation blocks. Due to the small size of $2M \times 2M$ of each block, the computation of the spectral radii is reliable and at least for many subintervals it may give acceptable estimations, as shown in Figure 3.8c and 3.8d.

Finally, we conclude with our choice of strategy for the following sections. Using the spectral radius has two major flaws: Its computation becomes unreliable and it cannot predict the convergence for low frequency modes due to the relation of the harmonic error mode pairs. One idea to overcome the first flaw for some cases, is to use the collocation blocks for the computation instead. The computation of norms is more reliable, but in most cases the norm $\|\mathbf{T}\|$ is greater than 1, which renders Strategy 2 useless. Strategy 3 shares the latter disadvantage with the use of the spectral radius and it also takes a matrix-matrix-multiplication to predict the error for each iteration, whereas the spectral radius has to be computed just once. Despite this, it is advisable to use Strategy 3 for the computation of a reliable upper bound, as we observed throughout the experiments. However, for our purposes of giving a more complete picture of PFASST, we will rely on Strategy 4. This has two advantages: First, it may be used as a heuristic *a priori* estimator, when semi-randomized error vectors are used. Arguably, the decomposition of blocks and the theory behind is not necessary to generate the same results as Strategy 4, but then it lacks the cost efficiency and the convenience of Lemma 5. Also, one inconvenience would be that for each minor change in the spatial problem a new entity $\mathbf{T}$ has to be generated without matrix symbols, whereas with matrix symbols these minor changes are integrated seamlessly. Second, it gives us the possibility to investigate the role of initial values and starting iteration values, which have a remarkable influence on PFASST. These influences will be investigated in the following section.

3.2. Mode damping fields

As "mode damping fields" we denote the plots in this section, where we illustrate how error modes are damped by PFASST and its constituents. With the help of these mode damping fields, it is possible to depict a set of initial values simultaneously. Furthermore,



(a) diffusion $L = 8$

(b) diffusion $L = 128$, (**tcb**)

(c) advection $L = 8$

(d) advection $L = 128$, (**tcb**)

Figure 3.9.: Pseudospectra of our diffusion and advection problem, for the full iteration matrix in the case of $L = 8$ and for one time collocation block (**tcb**) in the case of $L = 128$. The colors indicate the ε of the pseudospectrum on a logarithmic scale to the base of 10.

different dispersion relation numbers will be presented, in order to increase the level of abstraction and disengage from the two particular examples.

To achieve this, we present the possible error vectors in the form

$$\boldsymbol{e}_{k,j} = \boldsymbol{m}_k \otimes \boldsymbol{w}_j \in \mathcal{E},$$

as defined in Definition 4. Based on this definition and the choice of $\boldsymbol{w}_j$, we introduce two perspectives. In the first perspective the vector $\boldsymbol{w}_j$ contains weights related to the respective time nodes and $\boldsymbol{m}_k$ are the spatial error modes which are distributed and weighted on the different time nodes. This perspective was, for example, used in [58] to discuss the effect of spatial error modes on different positions in the temporal dimension, which relates to the use of the canonical basis vectors for $\boldsymbol{w}_j$. Likewise, the second perspective is found in literature, for example in the analysis of [60]. This second perspective treats the temporal dimension in the same fashion as the spatial dimension and employs oscillating modes

$$w_j(t) = \exp\left(ij2\pi t/T\right), \quad \text{with } j \in \{0, \dots, ML-1\}, \tag{3.11}$$

evaluated at the time points for $\boldsymbol{w}_j$. In this section we present insights in the working mechanisms of PFASST from both perspectives. Each perspective is associated with one type of mode damping field. The damping fields of first type depict the absolute value of the entries of an error vector after a transformation with the operator $\mathcal{F}$, which is mainly a Fourier transformation in the spatial dimension, see Section 2.2.1. It is the same operator that transforms the PFASST iteration matrices into block form. Then, the transformed error vector is cut into LM equal sized pieces and the absolute value of the entries is plotted together, stacked side by side. In this way, we may perceive how the spatial Fourier modes are distributed on the temporal grid after the application of the approximative block Jacobi (BJAC), coarse grid correction (CGC) or PFASST.

In contrast to this, the mode damping fields of second type do not represent individual error vectors, but the 2-norms of NML error basis vectors in one map after the application of BJAC, CGC, or PFASST. The error basis vectors $\boldsymbol{e}_{i,j}$ are generated by the function

$$e_{i,j}(x,t) = m_i(x)w_j(t), \text{ with } i \in \{1, \dots, N\}, j \in \{1, \dots, ML\}, \tag{3.12}$$

where $w_j(t)$ is defined in (3.11) and $m_i(x)$ generates the Fourier modes in space. By evaluating this function on the space-time grid and stacking the results in canonical order and subsequently normalizing the vector, we get $\boldsymbol{e}_{i,j}$. These error vectors refer to the error basis vectors in Definition 4. Accordingly, the application of BJAC and CGC may be performed along the lines of Lemma 5, which means that the respective block form is used and harmonic pairs are considered. Finally, each value of $\|\mathbf{T}\boldsymbol{e}_{i,j}\|_2$ produces one entry in the mode damping fields of second type.

Note that we aim for a certain aspect ratio of the plots and therefore double the number of points in space. To keep the dispersion relation of the standard cases of Section 3.1, we change the standard coefficients μ to $4 \cdot 10^{-2}$ and c to $1/2$.

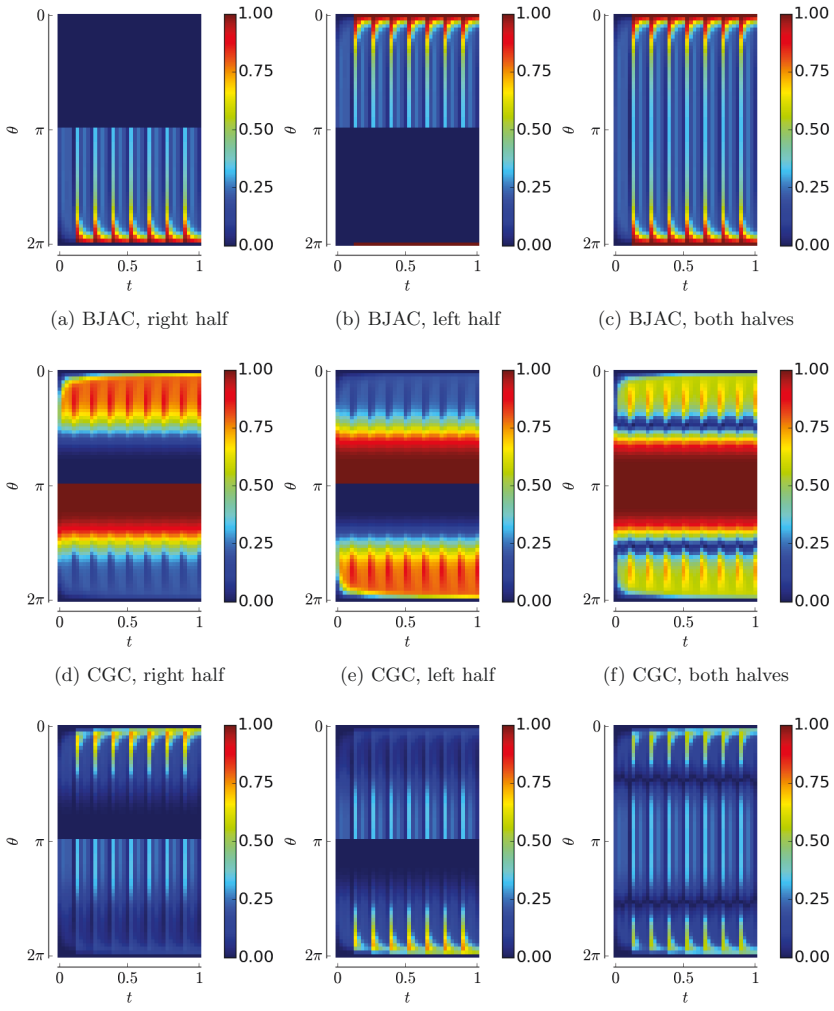
3.2.1. Using the canonical basis in time

The first challenge is to integrate the harmonic mechanics into the computation of the first type of mode damping fields. In our case for the coarse grid correction, always two spatial modes are connected, for each mode with frequency $\theta < \pi$ there is a harmonic mode with frequency $\theta + \pi$. Consequently, the frequencies on the left half ($\theta < \pi$) are connected to the right half ($\theta > \pi$), which also means that frequencies ($3\pi/2 > \theta > \pi/2$) are only connected to low frequencies. For more information about the different frequencies we refer to Section 2.3. However, in Figure 3.10 we see the effect of BJAC, CGC, and PFASST for the 3 cases. For the first case we distribute only modes with frequencies from Θ_{right} equally on each time point, for the second case we employ the frequencies Θ_{right} and for the last case we employ both halves of the frequency spectrum. We see the effect of the coarse grid correction on the right half in Figure 3.10d or the left half in Figure 3.10e. This effect is denoted as mode mixing, because modes of the respective other half appear after the application of CGC. More precisely, the low frequency part of the respective halves are mixed. Because no coarse level is involved for BJAC, this connection between modes is absent and therefore no mode mixing is observed, as Figures 3.10a to 3.10c show. For the whole PFASST iteration we observe mode mixing in Figures 3.10g to 3.10i, due to the CGC. We conclude that the modes are mixed at some point in the calculation anyway and by mixing the modes ourselves beforehand, we may half the number of plots in this section. So, instead of treating both halves separately, the mode damping fields in this section depict the results of

$$\mathcal{B}_k^{(CGC)} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \boldsymbol{w}_j,\ \mathcal{B}_k^{(S)} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \boldsymbol{w}_j \text{ and finally } \mathcal{B}_k^{(IT)} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \boldsymbol{w}_j, \tag{3.13}$$

which relates to equally mixed modes. The result of this kind of mode mixing leads to a symmetric mode damping fields in Figures 3.10c, 3.10f and 3.10i. More precisely, the left half is the mirror image of the right half. This is due to the linearity of the iteration matrix and the symmetry of how the modes are mixed.

Next, for Figure 3.11, we place the spatial error modes on different subintervals on particular quadrature nodes. Once the spatial Fourier error modes are placed at the beginning 3.11a, in the middle 3.11b and at the end 3.11c of a subinterval. The particular decomposition into different subintervals and furthermore into quadrature nodes is reflected in the Figures 3.11d to 3.11f, in form of the effects of the approximative block Jacobi iteration. An error located at the beginning of the subinterval is reduced effectively, an error located in the middle of the subinterval is reduced, but also scattered on

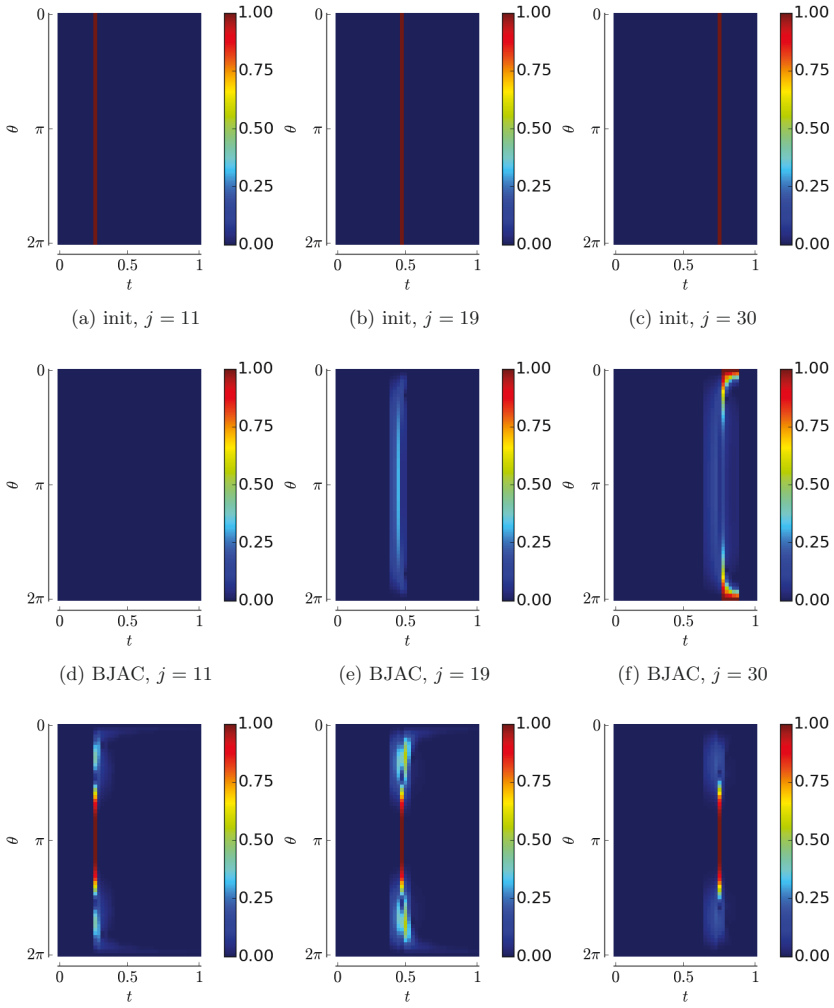


(a) BJAC, right half (b) BJAC, left half (c) BJAC, both halves

(d) CGC, right half (e) CGC, left half (f) CGC, both halves

(g) PFASST, right half (h) PFASST, left half (i) PFASST, both halves

Figure 3.10.: The damping field for the diffusion problem ($\mu = 4 \cdot 10^{-2}$), with only right half frequency modes, only left half frequency modes and finally both halves as suggested in (3.13).

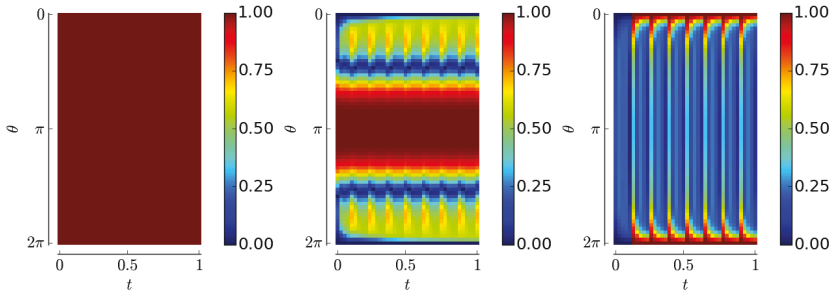


(a) init, $j = 11$ (b) init, $j = 19$ (c) init, $j = 30$

(d) BJAC, $j = 11$ (e) BJAC, $j = 19$ (f) BJAC, $j = 30$

(g) CGC, $j = 11$ (h) CGC, $j = 19$ (i) CGC, $j = 30$

Figure 3.11.: Mode damping fields of first type for the diffusion problem for the coarse grid correction (CGC) and the approximative block Jacobi smoother (BJAC), with all spatial error modes distributed to the j-th time node, where $j \in \{0, \ldots, LM - 1\}$.

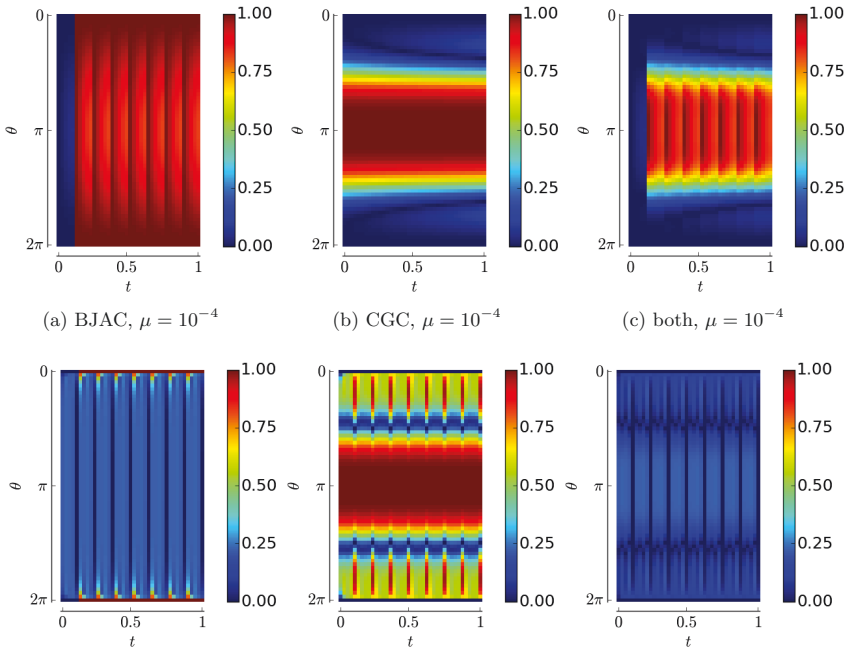


(a) init (b) CGC, $\mu = 4 \cdot 10^{-2}$ (c) BJAC, $\mu = 4 \cdot 10^{-2}$

Figure 3.12.: Mode damping field of first type for the diffusion problem with $\mu = 4 \cdot 10^{-2}$ to illustrate the effect of an approximative block Gauß-Seidel iteration, with transfer to the coarse grid (CGC), and the effect of an approximative block Jacobi iteration (BJAC). For the initial error distribution (init) each Fourier error mode in space is distributed equally on every quadrature node.

the quadrature nodes of the current subinterval and an error on the last quadrature node is scattered on the current and the subsequent subinterval. The last effect arises from the absent communication between two subintervals and mainly generates Fourier error modes in space with low frequencies. In contrast, the error modes are not scattered differently by CGC on the different quadrature nodes, see Figures 3.11g to 3.11i. In addition, the low frequency error modes are reduced efficiently, while the high frequency error modes are not changed.

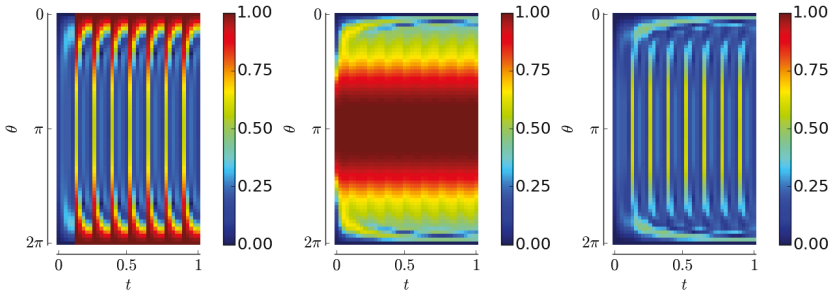
A more complete overview of the whole error reduction effects is given by Figure 3.12, where the same mixed spatial error modes are initially placed at each time node. In general, we observe that high frequency modes are reduced by the Jacobi iteration and the low frequency modes are reduced by the Gauß-Seidel iteration. Yet, by changing the dispersion relation number, we observe a change in the effect of our operators, which is shown in Figure 3.13. In the case of a low dispersion relation number, the block Jacobi smoothing only works on the first subinterval. The error spread at the interface of the subintervals is dominant and the block Gauß-Seidel smoother works almost ideal. This leads to an inferior reduction of the error for the combination of both solvers. For a high dispersion relation, on the other hand the block Jacobi solver reduces every error mode except the one with the lowest frequency and the Gauß-Seidel works exactly the other way around. This conserves the good overall convergence behavior of PFASST in the case of high dispersion relation numbers.



(a) BJAC, $\mu = 10^{-4}$ (b) CGC, $\mu = 10^{-4}$ (c) both, $\mu = 10^{-4}$

(d) BJAC, $\mu = 1.0$ (e) CGC, $\mu = 1.0$ (f) both, $\mu = 1.0$

Figure 3.13.: Mode damping field of first type to illustrate the effect of block Jacobi smoother, coarse grid correction (CGC) and the combination of both starting with the block Jacobi smoother, for a high and low dispersion relation number for the diffusion problem.



(a) BJAC, $c = 1/2$ (b) CGC, $c = 1/2$ (c) both, $c = 1/2$

Figure 3.14.: Mode damping field of first type to illustrate the effect of approximative block Jacobi smoother (BJAC), coarse grid correction (CGC) and the combination of both starting with (BJAC) for the advection problem with $c = 1/2$. The initial distribution depicted in Figure 3.12a is used.

In Figure 3.14, we only show the mode damping fields for the advection problem in the standard case $c = 0.5$, where both smoothers work jointly together. Already for $c = 0.0125$ the mode damping fields of the advection problem resembles the mode damping fields for the diffusion problem 3.13a to 3.13c and for $c = 5$ we observed a resemblance with 3.13d, 3.13e and 3.13f. In contrast to the possible coefficients μ for the diffusion problem, this leaves us with smaller range speeds c, for which PFASST is a suitable solver. Outside of this parameter space, PFASST loses its convergence behavior. The convergence behavior changes very quickly with the dispersion relation number and we will investigate this with mode damping fields of second type in the next section. In this section, the spatial error modes were only distributed homogeneously over the time nodes. When the spatial error modes are distributed in an oscillating fashion, the mode damping fields changes in a way that eventually impedes the convergence properties of PFASST. This is especially true for the advection problem and a good reason to study the second type mode damping fields, which are not bound to one distribution across the time nodes but visualize many possible distribution in time at once. This will further increase the level of abstraction.

3.2.2. Using the Fourier basis in time

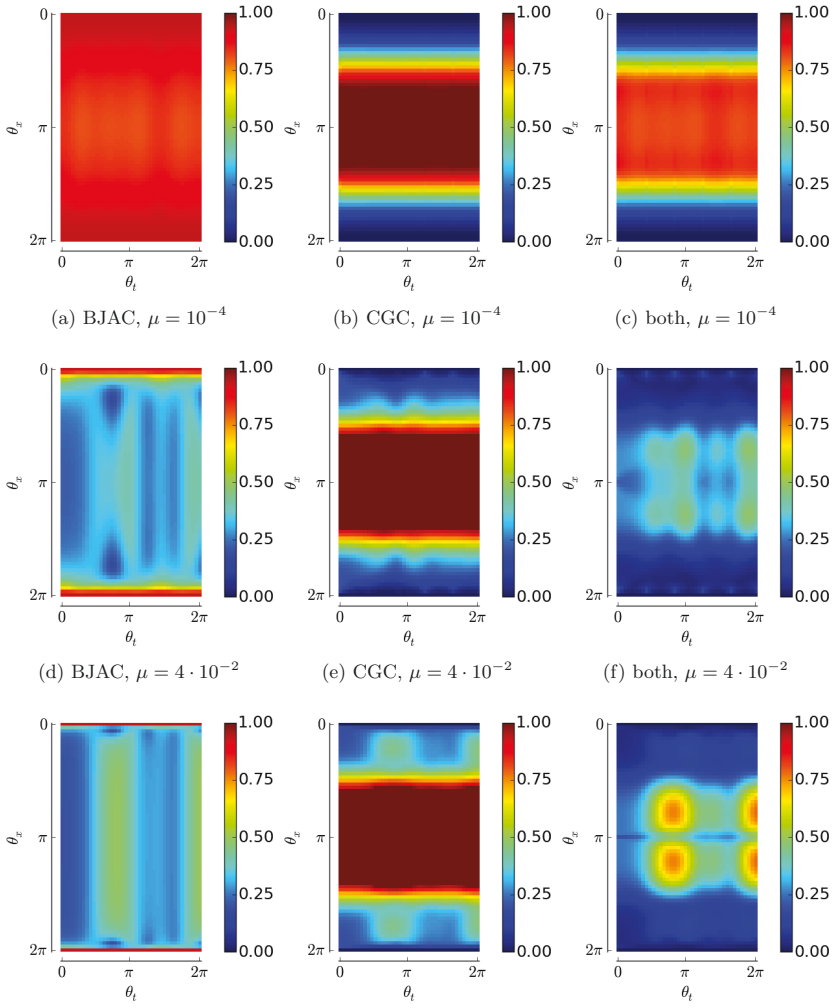
In the mode damping fields of the second type, each point represents a frequency in space and time. Due to the number of space-time grid points, only a finite number of space-time modes is representable on the space time grid, each one is found in the mode damping fields. Which means that almost every iteration starting value is considered, therefore we

are able to study PFASST on a more abstract level. However, as the generating function in (3.12) reveals, we make the assumption of separation of the spatial and temporal dimension. For the advection and diffusion problem, we know that the solution has separated temporal and spatial parts. Therefore, we make the same assumption about the error. For other problems, this may be not the case and hence could be a limiting factor and should be kept in mind.

In the next step, we implement the new mode damping fields by revisiting the diffusion cases depicted in Fig. 3.13 and 3.12. The result is found in Figure 3.15 and similar effects are observed. We see how the two solvers work jointly for the right dispersion relation and how the block Jacobi smoother disrupts the convergence for a small dispersion relation number in 3.15a. Additionally new observations are made, in Fig. 3.15f and 3.15i we see emerging regions of space-time frequencies, which are reduced less with an increasing μ. This definitely shows that it is important how the error modes are distributed across the time nodes.

The behavior of PFASST for the advection problem, cf. 3.16, is more interesting than for the diffusion problem. We notice that the coarse grid correction is not symmetric anymore and even for moderate speeds c we are able to find low frequency space modes, for which the coarse grid correction fails to reduce them, see for example Figure 3.16e. It is worthwhile noting at this point that the colorbars in all plots are clipped to an upper limit of 1. Therefore, regions where $\|\mathbf{T}\boldsymbol{e}_{i,j}\|_2 > 1$ are possible, but are not indicated by a distinctive color. We denote those regions as regions of instability. In these regions the application of BJAC, CGC, or PFASST leads to an amplification of the error, which increases with the respective dispersion relation number. The mode damping fields of the BJAC are no exception. They become asymmetric and show regions of instability. An unfortunate deformation of the regions of non-reduced space time frequency error modes, cf. 3.16d and 3.16h, leads to small regions of instability in Figure 3.16f. These regions also contain low frequency spatial error modes. These kind of instabilities are not observed for the diffusion problem in the region with spatial low frequency. We also find regions in the mode damping fields of the coarse grid correction of the diffusion problem, but generally not in the mode damping fields of the block Jacobi smoother. The experiments seem to suggest an upper bound less than 1 for all error basis vectors for BJAC. For all observed diffusion cases the combination of both smoothers also seem to be free of such regions of instability.

In conclusion, the success of PFASST is highly dependent on the dispersion relation number and the appearing modes. In this section we presented three dispersion relation numbers for both problems. For the advection problem the different speeds $c = 0.0125$, $c = 0.5$ and $c = 5$ and for the diffusion problem the different values $\mu = 10^{-4}$, $\mu = 4 \cdot 10^{-2}$ and $\mu = 1.0$ relate to low, moderate and high dispersion relation numbers. Each dispersion relation number leads to a different grade of stiffness and therefore results in a distinctive behavior of PFASST. In this chapter we have illuminated the mechanisms of convergence, yet we are not able to predict how PFASST compares to a serial alternative.

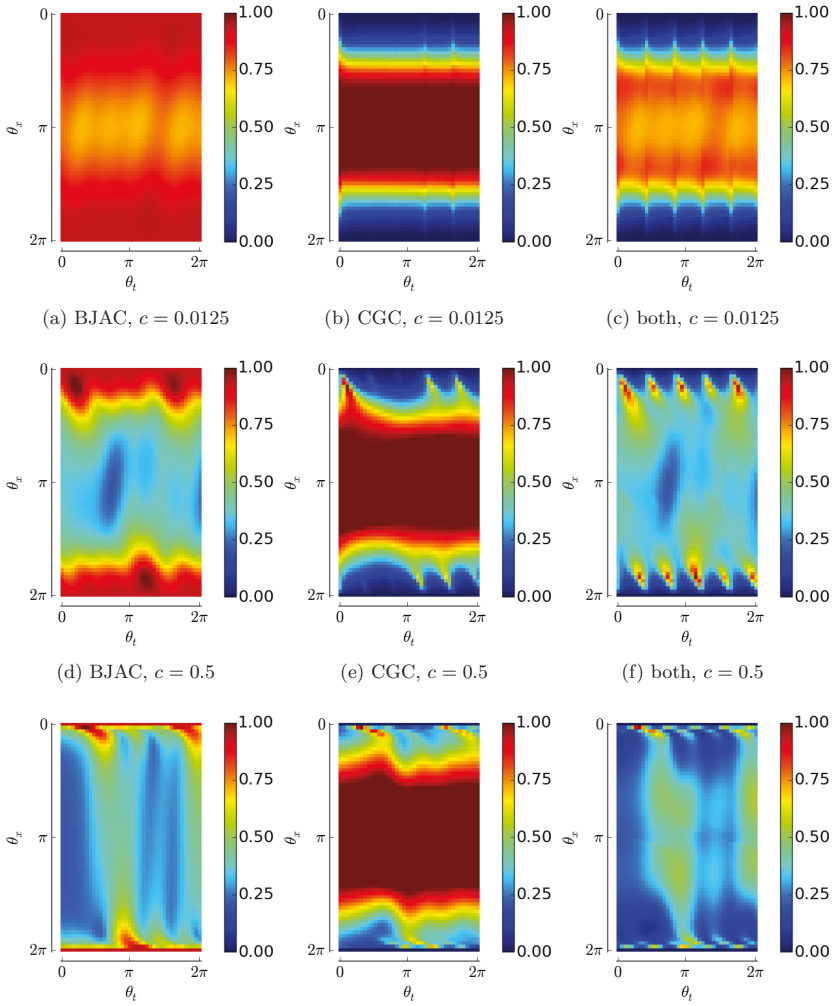


(a) BJAC, $\mu = 10^{-4}$ (b) CGC, $\mu = 10^{-4}$ (c) both, $\mu = 10^{-4}$

(d) BJAC, $\mu = 4 \cdot 10^{-2}$ (e) CGC, $\mu = 4 \cdot 10^{-2}$ (f) both, $\mu = 4 \cdot 10^{-2}$

(g) BJAC, $\mu = 1.0$ (h) CGC, $\mu = 1.0$ (i) both, $\mu = 1.0$

Figure 3.15.: Mode damping field of second type to illustrate the effect of approximative block Jacobi smoother (BJAC), coarse grid correction (CGC) and the combination of both starting with (BJAC), for the diffusion problem with different μ.



(a) BJAC, $c = 0.0125$ (b) CGC, $c = 0.0125$ (c) both, $c = 0.0125$

(d) BJAC, $c = 0.5$ (e) CGC, $c = 0.5$ (f) both, $c = 0.5$

(g) BJAC, $c = 5$ (h) CGC, $c = 5$ (i) both, $c = 5$

Figure 3.16.: Mode damping field of second type to illustrate the effect of the approximative block Jacobi smoother (BJAC), coarse grid correction (CGC) and the combination of both starting with (BJAC), for the advection problem with different c.

Therefore, we take over these dispersion relation numbers to the following chapter, where we analyze the parallel performance of PFASST by studying the speedup.

4. Parallel performance

> Everyone knows Amdahl's law, but quickly forgets it.
>
> Thomas Puzak, IBM,2007

In this chapter, we conclude the analysis of PFASST with the study of the parallel performance, under the conditions adopted from the previous chapter. We begin by briefly introducing the basic terms of how to measure parallel performance. Then, we derive two main strategies to distribute the workload of PFASST on multiple processes. Finally, in the last section, we present numerical results.

4.1. Basics

Amdahl's law [96], dating back to 1967, belongs to the curriculum of every computer science student. It states the theoretical limit of the speedup of an algorithm, which has a fixed workload consisting of two parts. One part is only executable in serial, the other part is perfectly parallelizable. Let p be the percentage of the latter part and P the number of the used processes, then Amdahl's law states that the speedup S is limited by

$$S \leq \frac{1}{(1-p) + \frac{p}{P}}.$$

When divided by the number of processes, it yields the parallel efficiency. For a parallel efficiency of 1, it is necessary that every part of the workload is parallelizable. We speak of strong scaling, if a fixed workload is distributed effectively. In 1988, Gustafson reevaluated Amdahl's law [97] by introducing the term of weak scaling. This term describes the case, in which the workload increases proportionally with the number of processes. Thus, Gustafson's law states the theoretical limit

$$S \leq (1-p) + P \cdot p$$

for the speedup.

However, neither of these laws are not directly applicable for our purpose. The reason for this is the non-linear relation between the number of processes and the workload

of PFASST. In detail, when we change the number processes, we need to change the number of subintervals for PFASST. This changes the setup, the numerical behaviour, the number of iterations, as well as the workload. Since, in most cases, PFASST does not perform optimally for only one subinterval, a speedup computed by the comparison to such a PFASST setup would provide no significant statement. Therefore, it is an additional challenge to find a serial method for comparison and hence for the computation of a reasonable speedup.

4.2. Two parallelization strategies

The very first thing we assume is the optimal parallelization of the spatial problem. This means that we have several sets of processes for the parallelization of the spatial problem. The natural way to distribute the work of PFASST is to assign one set of processes to each subinterval. Within the set of processes, communication is needed to solve the spatial problem. Between the set of processes, we have the typical PFASST communication, consisting of forward-only communication on the different levels. The only blocking communication, where one set of processes has to wait for the preceding to finish its computations, is found on the coarsest level. On the finer levels, the communication between subintervals may be performed, while SDC sweeps are performed on the coarser levels. Of course, parts of the set of processes have to interrupt their computations to receive and send information on the finer levels. Nevertheless, this communication is non-blocking and we may consider the time needed to receive and send information as a negligible part of the time needed to perform one fine SDC sweep T_F. Note that we assume a pipelined arrangement of the SDC sweeps as depicted in Figure 1.2 and 4.1, although the formulation of the algorithm is given in form of a two-grid operator, which may not suggest such an arrangement at first glance. In the following sections we distinguish two parallelization strategies. The first strategy assigns only one subinterval per set of processes ($P = L$), whereas the second strategy assigns multiple subintervals to one set of processes ($P < L$). Subsequently, we will estimate the wall-time for both strategies.

4.2.1. Estimating the wall-time of PFASST

In Table 4.1, we find the quantities which are required for the computation of the speedup. The speedup is the ratio between the wall-time of a serial method to the wall-time of a parallel counterpart. Ideally, the parallel method should be compared to the best serial method available. The theory of time stepping methods is still an evolving field of research, already today this field holds many answers regarding the best choice of the method. This choice depends strongly on the requirements of problem, which we try to solve. Here, we assume that we are able to find a quasi-optimal configuration of SDC for the problem at hand and use this as our serial method, which leads to the serial

Name	Description
T_F	time needed for the fine time stepping method
T_C	time needed for the coarse time stepping method
α	ratio T_C/T_F
L	number of subintervals
γ_F	time needed to communicate data on the fine level
γ_C	time needed to communicate data on the coarse level
K_p	number of PFASST iterations needed to achieve a certain accuracy ϵ
K_s	number of fine SDC sweeps performed in serial needed overall to achieve a certain accuracy ϵ
γ_0	time needed to interpolate and restrict between fine and coarse level
P	number of set of processes used
ν	number of relaxation steps in each PFASST iteration
k	ratio L/P

Table 4.1.: Quantities needed for the computation of the parallel efficiency

wall-time

$$T_s(K_s) = \sum_{k=1}^{L} K_{\mathrm{SDC},k} \cdot T_F = K_s \cdot T_F.$$

Here, $K_{\mathrm{SDC},k}$ is the number of SDC sweeps needed on the k-th subinterval and K_s thus the sum of SDC sweeps. Consequently, our parallel counterpart is a PFASST algorithm, which employs this SDC method on different levels with different spatial grids. It is not just a redistribution of the workload of the SDC steps, which are performed in serial, but a new time stepping method with new properties for every change in configuration, like a different number of subintervals or a different coarse level.

When a proper configuration of PFASST proves to be successful, the resulting work-

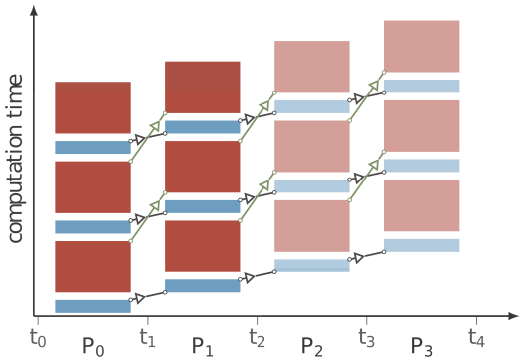


Figure 4.1.: The distribution of the fine SDC sweeps (red blocks) and the coarse SDC sweeps (blue blocks) for PFASST in the case of $P = L$.

load may be distributed to different numbers of set of processes. The maximal number of set of processes is the number of subintervals ($P = L$), since we use only one instance of PFASST and not several consecutive instances of PFASST. This is illustrated in Figure 4.1. First of all, we have to account for the coarse SDC sweeps until the last subinterval, which leads to LT_C. In this sketch, we see 3 PFASST iterations. For each PFASST iteration, we need coarse SDC sweeps, ν fine SDC sweeps, time to receive the new right-hand side on the fine and coarse level $\gamma_C + \gamma_F$, and finally the time for interpolation and restriction $2\gamma_0$. The wall-time of a two-level PFASST sums up to

$$\tilde{T}_p(K_p, L) = (L-1)(\gamma_G + T_C) + K_p(\nu T_F + \gamma_F + \gamma_C + 2\gamma_0 + T_C).$$

In the case of $P < L$, the implementation gets more involved and the memory requirements are higher, since one set of processes now has to handle more than one subinterval. Consequently, it has to store the states of multiple subintervals. If we, for example, have 8 subintervals but only 2 set of processes at our disposal, one set of processes has to store the states of all 4 even subintervals and the other set stores the states of the odd subintervals. The scenario with 3 set of processes is less desirable because, following the same logic, each set of processes has to cycle through the subintervals. Therefore, for the execution of the sweeps, the states and new right-hand sides of all subintervals are needed. When we do not cycle through the subintervals, we have to assign an unbalanced number subintervals to the different sets of subintervals. In our scenario, this could mean that we assign 6 subintervals to two set of processes, 3 each, and the remaining 2 subintervals to the last set of processes. Inevitably, this would result in waiting time for, at least, one set of processes. To avoid this intricate scenario, we assume that the number of subintervals is a multiple of the number of set of processes.

If the last set of processes carried out the SDC sweep on the coarsest level and needs to transfer the new right-hand side to the next subinterval, then a receiving set of processes is needed. At this point, we face two possibilities: first the initial set of processes is still computing, or it has completed the current SDC sweep and is waiting for the new initial value. Both cases are depicted in Figure 4.2. The first case is preferable because it does not result in idle time. To account for this idle time in the latter case, we adjust the time of the fine sweep

$$T_F \leftarrow \begin{cases} T_F & \text{, for } (P-1)(\gamma_C + T_C) < T_F\nu + \gamma_F + \gamma_C + 2\gamma_0 \\ (P-1)(\gamma_C + T_C) & \text{, else} \end{cases} \tag{4.1}$$

by using the time of the coarse sweeps and communication, instead of the time of the usual fine sweep. This leads to the main constrain of the second strategy ($P < L$)

$$(P-1)(\gamma_C/T_F + \alpha) < \nu + (\gamma_F + \gamma_C + 2\gamma_0)/T_F, \tag{4.2}$$

which simply reads $P\alpha \leq \nu + \alpha$ in the case of negligible communication and transfer costs γ_F, γ_C, and γ_0. With this adjustment in mind, we estimate the wall-time of K_p PFASST iterations on $P \leq L$ set of processes as

$$\tilde{T}_p(K_p, P, L) = (P-1)(\gamma_C + T_C) + K_p L/P(\nu T_F + \gamma_F + \gamma_C + 2 \cdot \gamma_0 + T_C).$$

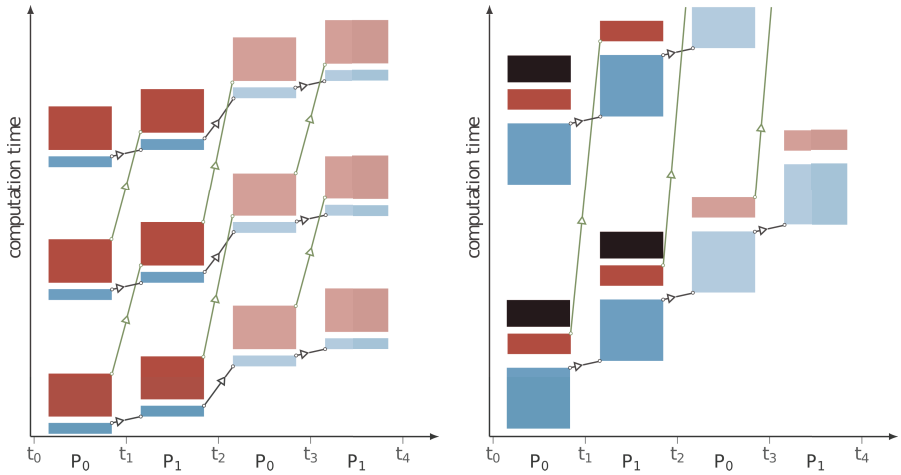
Figure 4.2a depicts the ideal case, in which the fine sweep takes more time than the coarse sweeps on $P-1$ sets of processes. We observe graphically that the faint blue and red blocks of the third and fourth subinterval fit perfectly between the non-transparent blocks of the first two subintervals. Then, the wall-time is measured similar to the case $P = L$, but with $P-1$ instead of $L-1$ coarse sweeps and L/PK_p iterations instead of K_p iterations. In the non-ideal case, depicted in Figure 4.2b, the faint blue and red blocks have more than enough space to fit between the non-transparent blocks of the first two subintervals. Thus, idle time emerges, which is depicted in form of black blocks.

With the estimation of the wall-time, we are able to estimate the speedups in the following sections.

4.2.2. Speedup for the case $P = L$

Henceforth, we assume that the cost of communication, interpolation and restriction is negligible and, therefore, we set all values in question to zero. Thus, the speedup reads

$$S(K_s, K_p, L) = \frac{T_s(K_s)}{T_p(K_p, L)} = \frac{K_s T_F}{(P-1)T_C + K_p(\nu T_F + T_C)}.$$



(a) ideal (b) non ideal

Figure 4.2.: The distribution of the fine SDC sweeps (red blocks) and the coarse SDC sweeps (blue blocks) for PFASST in the case of $P < L$. The black blocks represent idle time, which is the result of a violation of the constraint in Equation (4.2). Note that the more transparent blocks are assigned to the same set of processes.

The theoretical limit of the PFASST speedup is highly dependent on the relation between the iteration numbers $\frac{K_s}{K_p}$ and the relation between the time needed for the fine and coarse step $\frac{T_C}{T_F}$. For example, we notice that

$$\text{if} \quad \frac{T_C}{T_F} \to 0, \quad \text{then} \quad S \to \frac{K_s}{\nu K_p}.$$

This means that PFASST needs to converge reasonably-well, in order to keep the number of iterations low enough. Note that K_p relates to the simultaneous treatment of several subintervals, whereas K_s relates to the treatment of all subintervals one by one.

This speedup has already been described in [17], the original PFASST paper. Additionally, a comparison to Parareal was made as follows: Assume that the serial solver is time stepping method of order $2m$, which, with the right collocation points, is achieved by $2m$ SDC iterations on each subinterval, cf. [66]. Thus, we have $K_s = P \cdot 2m$ and therefore the parallel efficiency reads

$$\frac{S(K_s, K_p, L)}{P} = \frac{2mT_F}{(P-1)T_C + K_p(\nu T_F + T_C)} \to \frac{2m}{\nu K_p}, \quad \text{for} \quad \frac{T_C}{T_F} \to 0.$$

For Parareal and the same limit, we observe a parallel efficiency of $1/K_p$, see Section 1.1.1 for details. In comparison, PFASST has a significantly improved parallel efficiency when a high-order method is employed.

However, in this work we will view the serial method as an iterative method, which needs different numbers of iterations depending on the problem at hand. Thus, we will measure the iterations K_p and K_s for the problem setups introduced in Chapter 3. Additionally, the quantities T_C and T_F are easy to estimate because they mainly depend on the number of right-hand side evaluations needed to solve the space problem. As a rough estimation for T_C/T_F one may assume the value $1/2^d$, where d is the dimension of the spatial problem.

In the same sense we will compute the speedup for $P < L$ in the following section.

4.2.3. Speedup for the case $P < L$

For the second parallelization strategy, it is of interest how the wall-times with different number of set of processes for one configuration of PFASST relate to each other.

Therefore, we investigate the relation

$$\begin{aligned}\frac{T_p(K_p,L)}{T_p(K_p,L,P)} &= \frac{(L-1)T_C + K_p\,(\nu T_F + T_C)}{(L/k-1)T_C + K_p k\,(\nu T_F + T_C)} = \frac{(L-1)\alpha + K_p\,(\nu+\alpha)}{(L/k-1)\alpha + K_p k\,(\nu+\alpha)} \\ &= \frac{L-1+K_p\,(\nu/\alpha+1)}{L/k-1+K_p k\,(\nu/\alpha+1)} = \frac{k}{1+(k-1)\left(\frac{(k+1)K_p(1+\nu/\alpha)-1}{L-1+K_p(1+\nu/\alpha)}\right)} \\ &= \frac{k}{1+(k-1)\,\varsigma},\end{aligned} \tag{4.3}$$

where we denote $k = L/P$ and

$$\varsigma = \frac{(k+1)K_p\,(1+\nu/\alpha)-1}{L-1+K_p\,(1+\nu/\alpha)}.$$

Then, we observe that

$$\text{if} \quad K_p \ll L, \quad \text{then} \quad \varsigma \to 0 \qquad \text{and, therefore,} \quad \frac{T_p(K_p,L)}{T_p(K_p,L,P)} \to k$$

$$\text{and if} \quad K_p \gg L, \quad \text{then} \quad \varsigma \to k+1 \qquad \text{and, therefore,} \quad \frac{T_p(K_p,L)}{T_p(K_p,L,P)} \to \frac{1}{k}.$$

These are the two extremes. When P sets of processes are used, the wall-time is either k times longer in one extreme or k times shorter in the other extreme. This would indicate a higher parallel efficiency, when less processes are used. Not every relation is possible due to the constrain in Eq. (4.2), as we will show by contradiction. To this end, we begin with some simple algebraic manipulations, which yield

$$\text{that} \quad \frac{T_p(K_p,L)}{T_p(K_p,L,P)} = z \quad \text{is equivalent to} \quad \varsigma = \frac{k/z-1}{k-1}.$$

Then, we choose $z \geq 1$, which is the case where the wall-time is reduced when less sets of processes are employed. Thus, we claim

$$\varsigma \leq \frac{k/z-1}{(k-1)} = 1.$$

Additionally, we assume that T_F is not adjusted, which, according to (4.2), is equivalent to the constraint $P\alpha < \nu$. Together, this leads to the inequality

$$\frac{(k+1)K_p\,(1+\nu/\alpha)-1}{L-1+K_p\,(1+\nu/\alpha)} \leq 1, \text{ which is equivalent to } K_p\,(\alpha+\nu) \leq P\alpha \leq \nu+\alpha,$$

with α positive, $\nu \geq 1$ and $K_p \geq 1$ this yields a contradiction for $K_p > 1$. For $K_p = 1$ and $P\alpha = \alpha + \nu$ it holds that $T_p(K_p, L)/T_p(K_p, L, P) = 1$. This is the only case in which the wall-times of both strategies are equal. Therefore, the wall-time is shortest for the case in which one subinterval is assigned to one set of processes ($L = P$). Nonetheless, an improvement of the parallel efficiency is still possible because when $P < L$ set of processes are employed, the wall-time is still less than k times the wall-time of $T_p(K_p, L)$. To verify this, we assume $z > 1/k$, which leads to the inequality

$$\varsigma < \frac{k^2 - 1}{k - 1} = k + 1, \text{ which is equivalent to } 0 < (L - 1)(k + 1).$$

Fortunately, this is always fulfilled. This means that it is possible to improve in parallel efficiency by using less processes. Note that this statement could also be established by a careful observation of Equation (4.3).

Next, we will investigate how the second strategy affects the speedup of PFASST. When communication is neglected in the estimation of the wall-time, the computation of the speedup reduces to

$$\begin{aligned} S(K_s, K_p, L, P) = \frac{T_s(K_s)}{T_p(K_p, L, P)} &= \frac{K_s T_F}{(P-1)T_C + L/P K_p(\nu T_F + T_C)} \\ &= \frac{K_s}{(P-1)\alpha + L/P K_p(\nu + \alpha)}. \end{aligned} \tag{4.4}$$

Similar to the previous section, we notice that

$$\text{if} \quad \frac{T_C}{T_F} \to 0, \text{then } S \to \frac{K_s}{k\nu K_p}.$$

This means that in this limit, with the same number of iterations K_p, the speedup is only one-k-th of the speedup of the first strategy on P subintervals. Now, if we assume that we have K_{p_1} PFASST iterations on L subintervals and K_{p_2} PFASST iterations with P subintervals, then for the case $K_{p_1} k \leq K_{p_2}$, the speedup is improved, and for the opposite case reduced. Thus, it depends on the convergence properties for different setups of PFASST.

Next, we study this effect from a slightly different perspective by observing a relation similar to the relation in (4.3), which reads

$$\frac{S(K_s, K_p, L, P)}{S(K_s, K_p, L, 2P)} = \frac{T(K_p, L, 2P)}{T(K_p, L, P)} = \frac{2P - 1 + \frac{L}{2P} K_p(\nu + \alpha)}{P - 1 + \frac{L}{P} K_p(\nu + \alpha)}. \tag{4.5}$$

This relation is of interest because it gives an upper and lower bound for the speedup when the number of set of processes is halved. By simple algebraic manipulation we rec-

ognize that the relation is monotonically decreasing in K_p and monotonically increasing in P. For the lower bound, we let $K_p \to \infty$, then the relation converges to $\frac{1}{2}$. So, in the best case, half the processes means half the speedup and, therefore, constant parallel efficiency. The worst case occurs for just one iteration and $P = 1 + \nu/\alpha$, according to the constraint from Equation (4.1). Thus, in summary it holds

$$\frac{1}{2} \leq \frac{S(K_s, K_p, L, P)}{S(K_s, K_p, L, 2P)} \leq \frac{1 + 2\frac{\nu}{\alpha} + \frac{k}{2}(\nu + \alpha)}{\frac{\nu}{\alpha} + k(\nu + \alpha)}.$$

Note that the estimations in this section are constrained by $P\alpha < \nu + \alpha$, to avoid adjusting T_F according to (4.1). In summary, the theoretical bounds for the second strategy appear to be promising. We conclude the theoretical part of this chapter with an outlook of the multi-level case.

4.2.4. Estimating the speedup of multi-level PFASST

If we use $l + 1$ levels with PFASST, we just need to substitute the original T_F with

$$\begin{aligned}\hat{T}_F &= T_F + T_{m_1} + \ldots + T_{m_l} \\ &= T_F \sum_{i=0}^{l} q^i = T_F \cdot \frac{1 - q^{l+1}}{1 - q}\end{aligned}$$

where we assume $T_{m_i} = q^i \cdot T_F$ with $q \in (0, 1)$. Furthermore, we assume $T_C = q^{l+1} T_F$ for the coarsest level. Together this yields a new estimation for the wall-time of PFASST and, therefore, the speedup reads

$$S(K_s, K_p, L, P, l, q) = \frac{T_s(K_s)}{T_p(K_p, L, P, l, q)} = \frac{K_s}{(P-1)q^{l+1} + kK_p\left(\nu\frac{1-q^{l+1}}{1-q} + q^{l+1}\right)}.$$

For the asymptotic case $q \to 0$, we get the same limit as in the previous section. Furthermore

$$S \to \frac{K_s}{k\nu K_p \frac{1}{1-q}} \quad \text{for} \quad l \to \infty.$$

Therefore, when the number of iterations K_p remains constant, the speedup declines when more than two levels are used. Every additional level has to be justified with an improved convergence of PFASST.

However, using the two-level version of PFASST and the examples introduced in the previous chapter, we will assess the parallel performance.

4.3. Performance analysis

PFASST is designed to be used parallel in time and should be evaluated with this in mind. With the speedup introduced in Sec. 4.2, the computation of parallel efficiency reads

$$\frac{S(K_s, K_p, L, P)}{P} = \frac{K_s}{P(P-1)\alpha + LK_p(\nu + \alpha)}. \tag{4.6}$$

In literature, the plot of the parallel efficiencies is more commonly found because one can immediately read the improvement on a system with more cores. Nonetheless, we will only present speedup plots in this work. The main reason for this is that we still study PFASST on a qualitative level and the effects of changes in the setup of PFASST are more easily illustrated with the speedup than with the parallel efficiency. Another reason is that we are primarily interested in time-to-solution, which is also more easily illustrated with the speedup.

The estimation of the speedup itself is only dependent on the estimation of iteration numbers for different setups of PFASST, SDC, and MLSDC. We will now discuss a way to count these iterations.

4.3.1. Counting iterations

In order to count iterations, we need a setup of PFASST in the form of an iteration matrix, as well as an initial error vector. It is beneficial to have a matrix symbol of the iteration matrix as found in Theorem 6, since it enables us to observe different harmonic pairs individually. Furthermore, it is beneficial to use particular error vectors along the lines of Definition 4 in combination with Lemma 5, since it facilitates the application of the iteration matrix on these error vectors. Then, for a given tolerance, we apply the iteration matrix until the norm of the resulting error vector reaches the tolerance.

This approach has two drawbacks. First of all, the solution to the problem is unknown to most cases. Thus, it is impossible to measure the error. Usually, the tolerance is set for the residual and not for the error. To overcome this drawback, we make the bold but necessary assumption that the residual relates directly to the error. The second drawback has more serious implications, it concerns the temporal part of the error vector. If we examine the last columns in Figure 3.15 and 3.16, we realize that the frequency of the temporal error modes has a significant influence on the convergence behavior. In contrast to the spatial dimension, where we only have mode mixing between two modes, the modes of the temporal dimension are mixed with more than one counterpart. It is therefore more laborious to follow a temporal mode, than to follow a harmonic pair of spatial modes. Furthermore, it is unlikely that every space-time error mode has the same probability of appearance, especially when a reasonable initial estimation is made. For example, an efficient predictor step could yield such a reasonable initial estimation.

Here, we use a random but normalized vector $\boldsymbol{w}$ for the temporal dimension. In Section 3.1, and more precisely in Figures 3.4 and 3.5, we already employed this strategy and observed only a moderate variance in the convergence behavior. This suggests only a small expected variance in the number of iterations we count.

This choice of the error vector reflects the case, in which the spatial problem is well-known. In this case, for example, an engineer or scientist would be able to make an educated guess about the spatial error modes for his or her problem in an initial state. Based on this, we could easily confine our analysis to these error modes and, thus, make a more accurate prediction about the convergence behavior of PFASST for the respective problem. For cases where the comprehension of the spatial problem is insufficient, it is more appropriate to use fully randomized error modes in space and time with sufficient samples.

However, our knowledge of the spatial problem for our two model problems is extensive and our goal is to show the variety of parallel performances related to the initial spatial error modes. Especially helpful for this goal is the theory introduced in Chapter 2, since it enables us to treat the different spatial error modes individually. The measured parallel performance itself will come in a statistical form, since we have less knowledge about the problem in the temporal dimension. We will count the number of iterations several times until we get a statistically consolidated estimation of the iteration number. Subsequently, the standard deviation is used as uncertainty. The uncertainty of the estimated speedup is computed, by the rules of error propagation, cf. [98]. Let y be a quantity, which is dependent on a set of uncertain values $\{x_1, \dots, x_n\}$, afflicted with the uncertainties $\{u_1, \dots, u_n\}$ and independent of each other, then it holds

$$u_y = \sqrt{\left(\frac{\partial y}{\partial x_1} \cdot u_1\right)^2 + \left(\frac{\partial y}{\partial x_2} \cdot u_2\right)^2 + \cdots},$$

for the uncertainty of the quantity y. Our uncertain quantities are the number of iterations K_s and K_p. Following this rule, it yields for the speedup (4.4) the uncertainty

$$u_S(K_s, K_p, P) = \sqrt{\left(\frac{u_{K_s}}{(P-1)\alpha + \frac{LK_p}{P}(\nu+\alpha)}\right)^2 + \left(\frac{\frac{LK_s}{P}(\nu+\alpha)\, u_{K_p}}{\left((P-1)\alpha + \frac{LK_p}{P}(\nu+\alpha)\right)^2}\right)^2}. \tag{4.7}$$

We have established how to count iterations and how we handle uncertainties. Now, we have to define the setups for which we have to count iterations. In the numerical examples below, we divide the time domain into 64 equally sized subintervals and use μ and c such that the dispersion relation numbers of the previous chapter emerge once again. Since we used 8 subintervals in Section 3.2, this is achieved by multiplying the

factor 64/8 to each μ and c, in order to compensate for the smaller Δt of the new decomposition of the time domain.

For SDC or MLSDC, we count the number of iterations as follows: we iterate SDC or MLSDC on the first subinterval until the error tolerance is met, then the result is passed to the next subinterval and again the iterations are counted until every subinterval is covered. This is repeated 20 times with semi-randomized error vectors.

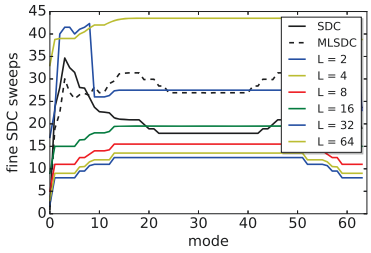
When PFASST is used, this procedure is applied for several setups of PFASST. Each time another power of 2 is chosen as a number of subintervals, which PFASST covers with one iteration. For example, let $L = 16$, then we iterate PFASST on the domain, which consists of the first 16 subintervals, until the error tolerance is reached. Then we pass the result to the next assemblage of 16 subintervals and perform and count iterations until the error tolerance is reached. This is repeated until the whole time domain is covered and subsequently the whole procedure is repeated 20 times with semi-randomized error vectors.

Note that it may at first sound unjustified that PFASST has to reach the same error tolerance for an assemblage of subintervals, while SDC has to reach a certain error tolerance at each individual subinterval. This means that the required error bound at the end of the assemblage is guaranteed by PFASST, but not by SDC. The first suggestion that comes to mind is to put a stricter error tolerance after a single subinterval, than after an assemblage of subintervals. Unfortunately, this cannot be done without a presumption being made of how an error associated to one subinterval effects the errors of the following subintervals or the assemblages of subintervals. Assume we have an ODE with a high sensitivity to the initial value, then a small error may end in a quite different result at the end of 64 subintervals, even when the error tolerance is achieved by SDC on each subinterval. This does not happen when PFASST is set up with $L = 64$; making it more stable to sensitive ODEs. As long as the sensitivity of the underlying ODE is moderate, this effect is not dominant and it is appropriate to expect the same tolerance on each subinterval and assemblage of subintervals.

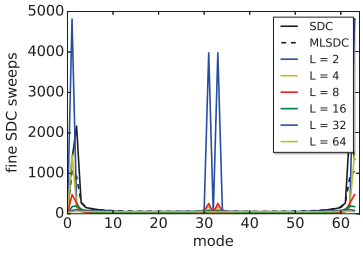
By adhering to a consistent set of rules regarding the process of counting iterations, we are able to make estimations of the iterations and, thus, the speedup. In particular, we are able to depict how many iterations are needed not only in sum, but for a certain spatial Fourier mode on each subinterval or assemblage of subintervals. To make the iteration counts comparable, we translate iterations into a number of fine SDC sweeps.

4.3.2. Distribution of iterations

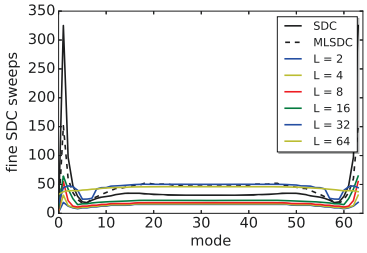
Before we examine the distribution of iterations across the subintervals, we look at the total sum of iterations across the different modes in Figure 4.3. We observe that the number of iterations is higher for the advection problem, than for the diffusion problem when we compare the plots on both sides. Also, we observe that with an increasing dispersion relation number, the iterations fall heavily for the diffusion problem



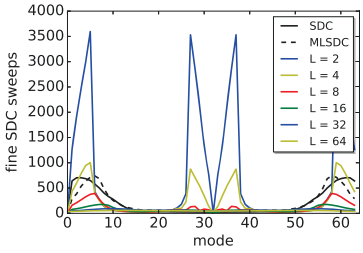
(a) $\mu = 1$



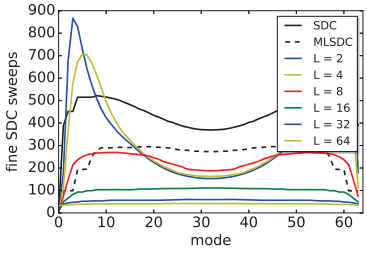
(b) $c = 5$



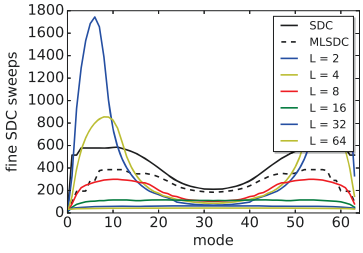
(c) $\mu = 4 \cdot 10^{-2}$



(d) $c = 0.5$



(e) $\mu = 10^{-4}$



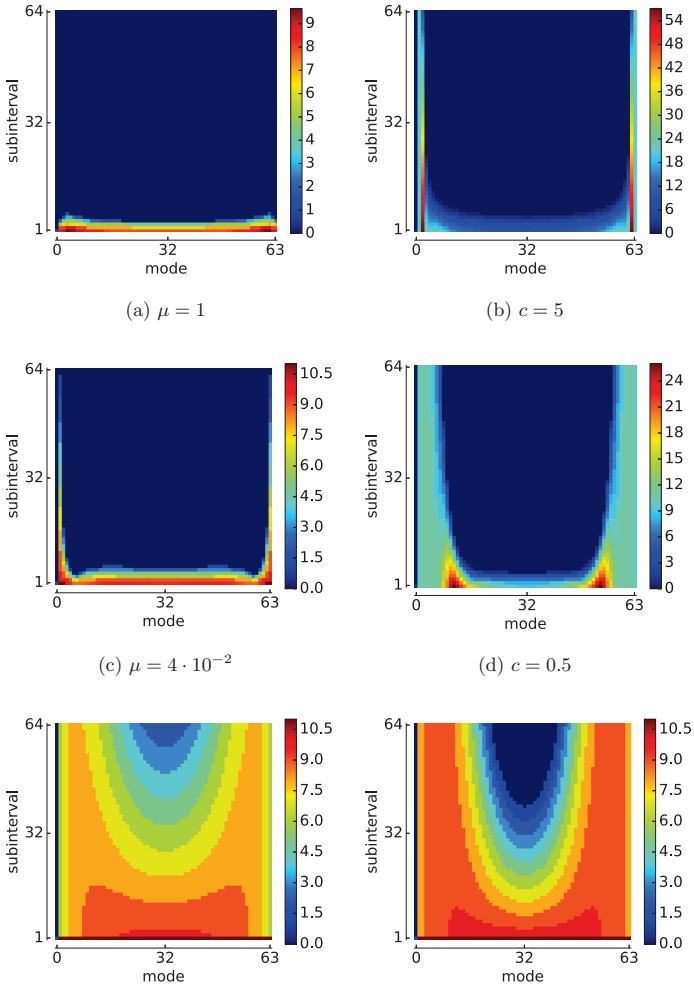
(f) $c = 0.0125$

Figure 4.3.: Total sum over assemblages of subintervals of iterations are measured in number of fine SDC sweeps to allow comparability and to reflect the wall-time. On the left we find the diffusion problem and on the right the advection problem. A error tolerance of $\epsilon = 10^{-6}$ and a total number of subintervals $L = 64$ is used.

but increase slightly for the advection problem. Another interesting finding is how diverse the different setups of PFASST become for different modes, when the dispersion relation is low for the diffusion problem and in general for the advection problem; Compare for example Figure 4.3a with Figure 4.3e or 4.3f. In the previous sections, we used setups of PFASST with $L = 8$ and observed a decent damping of spatial high frequency modes for moderate dispersion relation numbers and for both problems, see Figure 3.15f or 3.16f. Therefore, the high iteration count for spatial high frequency modes in Figure 4.3d or 4.3b was not anticipated by the mode damping field in Section 3.2. Finally, we point out that MLSDC seems to be the better choice for low dispersion relation numbers.

In regard to the distribution of the iterations across the subintervals, which for SDC are found in Figure 4.4, we observe for increasing dispersion relation numbers and lower spatial mode frequencies, that more iterations are needed in the later subintervals. The error advances further to the end of the time domain. This is observed for both problems and the same effect is observed for one setup of PFASST with a fixed L. However, what happens for different numbers of subintervals? The answer is found in Figure 4.5; where with the increasing size of the assemblages of subintervals, the error also advances further to the end of the time domain, the iteration numbers are reduced overall and are more evenly distributed over the whole time domain. Each effect observed here influences the speedup and therefore the parallel efficiency for the different setups.

Until now, only mean values have been presented. However, we must also consider the statistics. In Table 4.2, the mean over the different PFASST setups and error modes of the standard deviations of measured iterations and speedups are presented, along with the respective maxima. Admittedly, the information in the table is very condensed, nonetheless it shows that the computation of the speedup is very robust. Since the sum of iterations does not change significantly over the 20 semi-randomised initial values, the deviation of the iteration is small relatively to the absolute sum of iterations. Take for example the case with the diffusion problem and a high dispersion relation number. We observe that the maximum number of PFASST iterations K_p observed is 8. This is the maximum over all setups of PFASST, over all spatial error modes and over 20 different semi-randomized initial values. The mean over all these cases is 7.2. We use the 20 different semi-randomized initial values to compute the deviation δK_p for all setups and for all spatial error modes. We observe that the maximal deviation of K_p is 0.5 and that the mean is one magnitude smaller. This shows the reliability of the measurement of K_p for this case, since we have a mean relative deviation of 0.6 percent. The measurement of the number of fine SDC sweeps K_s needed for the serial run shows a similar reliability, since the mean relative deviation is around 1.6 percent for a mean iteration number of around 22 and a mean deviation of 0.35. It is worth noting that we are not able to read the relative worst case deviation from the table, because the maximal deviation could be associated with a much smaller iteration number than the mean or maximal number. This is not a problem, since the computation of the speedup and the deviation of the speedup according to Equation (4.7) considers the relation between number of iterations

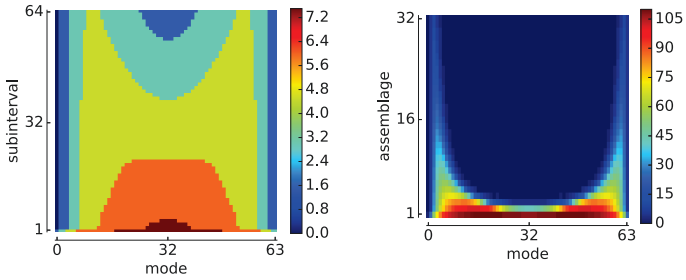


(a) $\mu = 1$ (b) $c = 5$

(c) $\mu = 4 \cdot 10^{-2}$ (d) $c = 0.5$

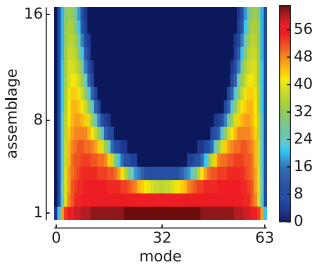
(e) $\mu = 10^{-4}$ (f) $c = 0.0125$

Figure 4.4.: SDC iteration distributions over subintervals and spatial error modes of the diffusion problem on the left and for the advection problem on the right. A error tolerance of $\epsilon = 10^{-6}$ and a total number of subintervals $L = 64$ is used. Note that the color bars have different scales.

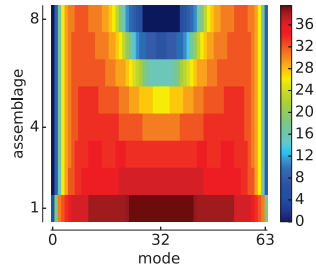


(a) $P = 1$

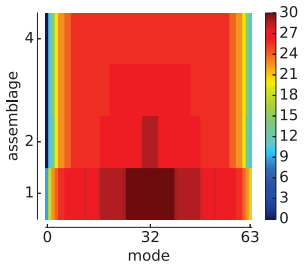
(b) $P = 2$



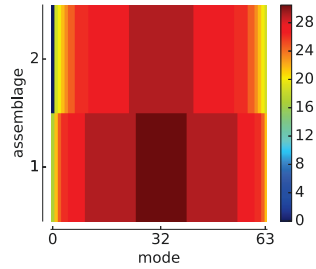
(c) $P = 4$



(d) $P = 8$



(e) $P = 16$



(f) $P = 32$

Figure 4.5.: Iteration distributions over assemblages of subintervals and spatial error modes of PFASST for the diffusion problem with $\mu = 10^{-4}$, measured in number of fine SDC sweep iterations. Except for the case $P = 1$, where MLSDC is used. A error tolerance of $\epsilon = 10^{-6}$ and a total number of subintervals $L = 64$ is used. Note that the color bars have different scales.

and the associated deviation. Since we are interested in the shortest time-to-solution, we are in particular interested in the highest speedup. For our case we measure a maximal speedup of 4.33 with a maximal deviation of 0.14, which is a very reliable result. For the diffusion case the maximal deviation of the speedup declines together with dispersion relation number, whereas the mean speedup rises, thus making the measurement of the speedups more reliable. For the advection problem we are not able to observe the same behavior. Nonetheless, the maximal deviation of the speedup is just 0.46 and it may be traced back to the high frequency error modes and the PFASST setup with $L = 2$. This is the only case, in which the computed speedup is significantly influenced by the initial distribution in the temporal dimension. Furthermore, the mean value of the deviation of the speedup never exceeds 0.03. Because of the otherwise small deviations of the speedup, we will introduce the speedup plots without error bars in the following section.

4.3.3. Resulting speedups

Overall, for both strategies and both problems, we encountered a great variety of speedups across Fourier modes in space and across dispersion relation numbers of the problem. Figure 4.6 presents this variety for the first strategy. In each plot of Figure 4.6, the gray area represents the possible speedups depending on the spatial error modes used in the initial value. Take, for example, the case with $c = 0.0125$, which is depicted in Figure 4.6f. First, we notice the dashed line representing the ideal speedup according to Amdahl's law. This represents the ideal case, in which a fixed workload is distributed perfectly over the available processes. Thus, at first glance, it is whimsical that the gray area surpasses the ideal speedup for $P = 1$ and $P = 2$ cores. PFASST is equivalent to MLSDC for $P = 1$. Therefore, we plotted the speedup resulting from using MLSDC instead of SDC. This shows that the workload is not just distributed, but that for every setup, a slightly different method with different properties and, therefore, different workload is applied to the problem. Additionally, we plotted 3 colored lines to represent the speedups for different spatial error modes. Each one represents one mode. For the high, moderate and low frequency, we use $\boldsymbol{m}_k$ with $k = 32$, $k = 16$, and $k = 1$ respectively. The blue speedup line, related to the low frequency, starts for PFASST at $P = 2$ and behaves similar to a typical speedup curve according to Amdahl's law. It starts with an almost ideal speedup and then converges to an upper bound. The other two speedup curves behave atypically. For example, the speedup related to the high frequency first declines with more processes and then rises again for 32 and 64 processes. The other plots also show atypical speedup, which is highly dependent on the frequency of the spatial error mode and the dispersion relation number of the respective problem. It shows, for example, in Figure 4.6c that for small dispersion relation numbers, MLSDC has a better convergence rate than SDC. Furthermore, it shows that for the diffusion problem and moderate or high dispersion relation numbers, PFASST is a good choice, as long as only two subintervals are computed at once. For the advection problem we recognize

		Dispersion relation number					
		high		moderate		low	
		mean	max	mean	max	mean	max
Diffusion	K_s	21.7468	35.0	41.9945	325.0	442.625	522.0
	δK_s	0.34113	0.58949	0.41344	0.66895	0.0	0.0
	K_p	7.17213	8.0	9.30403	25.0	106.894	567.0
	δK_p	0.04075	0.5	0.14953	0.57227	0.01348	0.49749
	S	1.32592	4.33125	1.83386	17.1052	4.80615	13.2051
	δS	0.02349	0.13952	0.02482	0.10003	0.00048	0.03352
Advection	K_s	165.080	2168.0	206.433	704.0	430.812	586.0
	δK_s	0.42556	0.71414	0.25037	0.65383	0.0	0.0
	K_p	57.5048	3198.0	136.843	2386.0	128.868	1152.0
	δK_p	0.43065	19.1102	1.67827	127.573	0.01691	0.5
	S	2.38166	36.5292	2.32251	17.3827	4.63581	14.9275
	δS	0.02647	0.45952	0.00992	0.06060	0.00165	0.15740

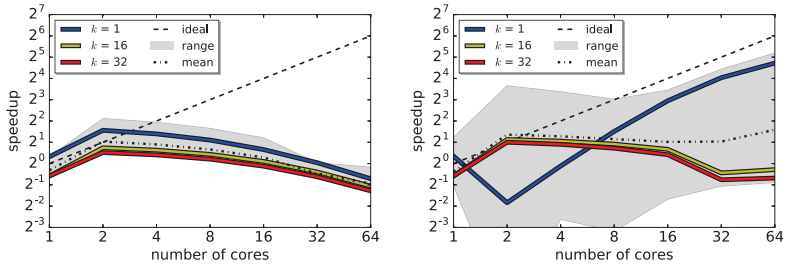
Table 4.2.: Maximal and mean over all measured setups of the sum of iterations for the serial case K_s, the parallel case K_p and for the speedup S for different. The deviation is noted by a δ in front of the quantity and computed over the 20 semi-randomized initial values. Then, the maximal and mean over all measured setups of the deviations are presented.

two trends. First, with decreasing dispersion relation numbers and increasing sizes of assemblages, the speedup gathers more around its mean value across the spatial Fourier modes. Second, the mean speedup improves with smaller dispersion relation numbers. This second trend is also found for the diffusion problem. In comparison to the advection problem, the diffusion problem shows less variety across the spatial Fourier modes.

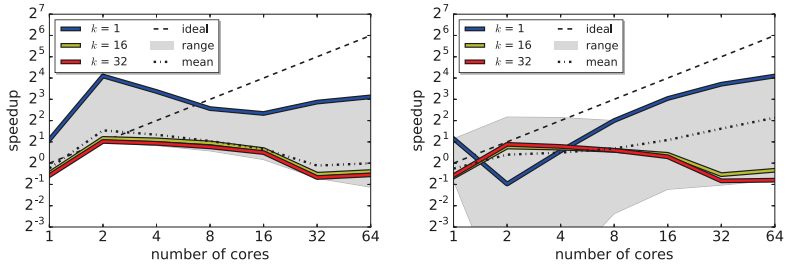
Regardless of this scattering of speedups, it is possible to achieve better speedups for the advection problem than for the diffusion problem when only low frequency spatial error modes form the initial error vector. In Figure 4.6b, we see the maximal speedup of 36.5 achieved for a low frequency mode and we also observe a very low speedup of 0.6, both for the case with 64 processes. Figure 4.6e, on the other hand, shows a reliable speedup between 8 and 16 for all spatial Fourier modes and $P = 64$. Independent of these different ranges of speedups, this clearly shows that the scaling limits are expected to be reached rather early. Already for 64 processes, we only reach a maximal parallel efficiency of 56 percent. This emphasizes the point that a parallelization in space should always be implemented first, since it is usually easier to achieve a higher parallel efficiency in the spatial dimension. Nonetheless, PFASST and parallel-in-time methods in general are a possible way to push scaling limits further.

Note that the presented results are not comparable to the results in the literature, e.g. [17]. The main reason for this is the employed serial solver. In [17], the serial solver is a SDC method of a fixed order m, which means that m fine SDC sweeps were performed on each subinterval. Let $m = 5$, then for all cases the serial cost equates to 320 fine SDC sweeps. Compared to the measured numbers of SDC sweeps in Figure 4.3, this would result in different speedups. For example, for the diffusion problem with $\mu = 1$ in Fig. 4.3a, the highest number of fine SDC sweeps measured in the serial case is 34. When this number is substituted by 320, the resulting speedup is almost 10 times greater. On the other hand for the advection problem with $c = 0.0125$, see Fig. 4.3f, the highest number of fine SDC sweeps measured in the serial case is 590. In this case, the resulting speedup would be smaller than the speedup presented in this section.

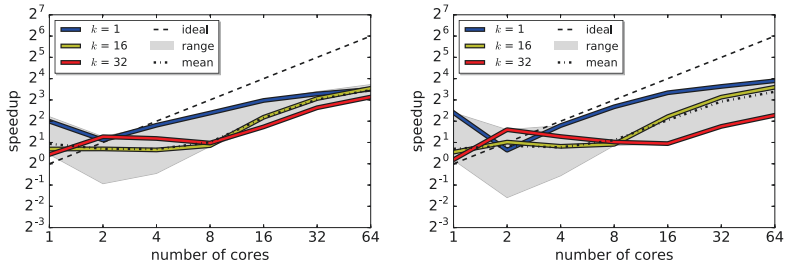
Next, we present the second parallelization strategy and the resulting speedups. When a reasonable PFASST configuration is found, it is possible to distribute the resulting workload to fewer processes P than the number of subintervals in the assemblage, as described in Section 4.2.3. Let us focus on Figure 4.7f, where we find the two bold speedup curves for a low and high frequency. These are the same two speedup curves as in Figure 4.6f. Additionally, we see branches of the same color starting at the respective speedups of the case $P = L$. Following these branches from right to left, we may observe how the speedup changes, when the workload of the case $P = L$ is distributed to less processes. Take, for example, the first blue branch that starts at the far right. First, the speedup declines significantly when the workload of PFASST with $L = 64$ is distributed to 32 processes. Then, the speedup remains nearly constant until the point at which the workload is distributed on 4 processes. Finally, the speedup declines again with at most the slope of the ideal speedup curve. These are the 3 phases. Each branch shows



(a) $\mu = 1$ (b) $c = 5$



(c) $\mu = 4 \cdot 10^{-2}$ (d) $c = 0.5$

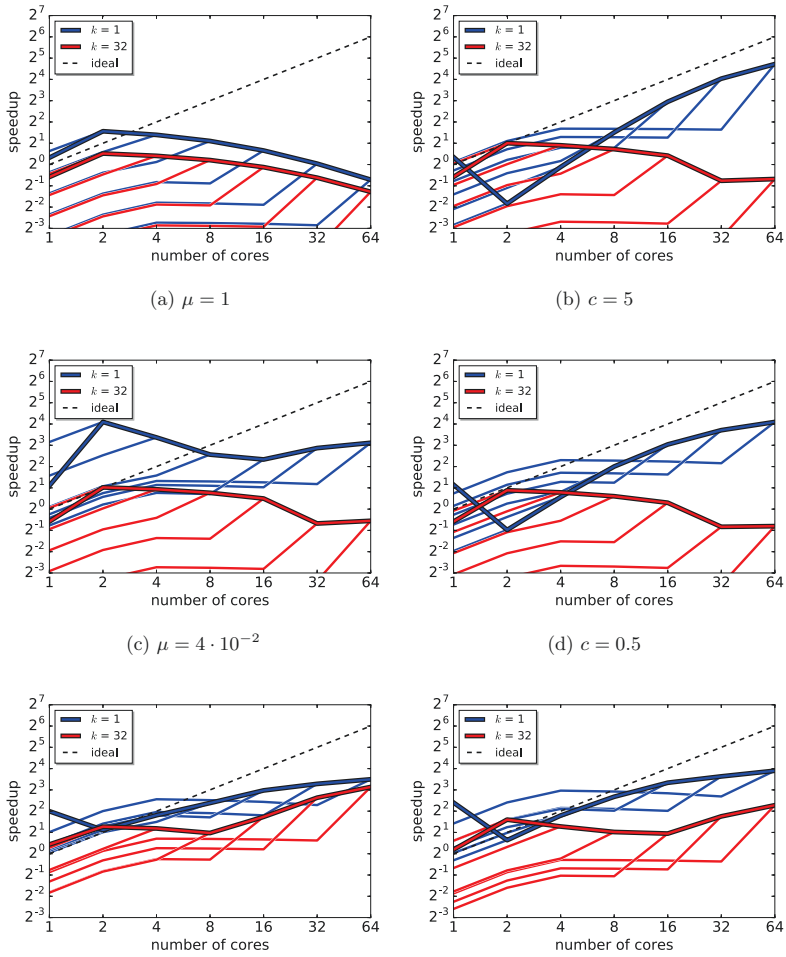


(e) $\mu = 10^{-4}$ (f) $c = 0.0125$

Figure 4.6.: On the left the speedup plots for the diffusion problem and on the right for the advection problem are depicted. Only speedups for certain Fourier modes in space are plotted. Also the mean speedup and the range of speedups is plotted. For $P = 1$ MLSDC was used to compute the speedup.

up to 3 of these phases, which may be explained along the lines of Section 4.2.3. The first phase, again read from right to left, consists of a drop in speedup due to restriction along the lines of (4.1), which is found immediately after the branch is forked. The next phase persists as long as $P\alpha > \nu + \alpha$ holds and results in a stagnating speedup. This way, the parallel performance is improved for decreasing P until the last phase begins. The last phase is explained by the asymptotic behavior of Equation (4.5) in the case $P \ll LK_p(\nu + \alpha)$. In this case, the speedup has at most the slope of the ideal speedup curve. Again, it depends on various parameters, whether this strategy is successful or not. In most cases the decrease of the first phase is severe enough to let this strategy fail for all numbers of processes $P < L$. See for example Figure 4.7a, where each branch has a lower speedup than the main speedup of the case $P = L$. For some cases the decrease is intercepted by the second phase for a sufficient number of configurations of P, so that the parallel performance is improved in comparison to the main speedup curve. See for example the blue line for all advection cases in Figure 4.7. There the first branch from the right crosses the main branch near $P = 8$ and from there improves the parallel performance of PFASST in comparison to the previous parallelization strategy. Note that with this strategy, PFASST is sometimes even able to surpass MLSDC, when the workload of $L = 2$ is distributed to one single process, for example in Figures 4.7a and 4.7c.

In conclusion, the numerical experiments so far primarily showed that there is a great variety of speedups across the dispersion relation numbers and across the spatial error modes. It shows the importance of the theory established in Chapter 2, which enables us to easily study the different error modes in spatial dimension individually. It also enables us to study the influence of the spatial and temporal part of the error vector separately. This shows us that the influence of the temporal part of the error vector is much weaker. We observed these influences for two different ways of parallelizing PFASST and we found configurations, which yield reasonable speedups for both model problems. Under the right conditions, it is therefore possible to push the scaling limits using PFASST. In the following chapter we will summarize the work and give ideas to improve the parallel performance and the understanding of PFASST.



(a) $\mu = 1$

(b) $c = 5$

(c) $\mu = 4 \cdot 10^{-2}$

(d) $c = 0.5$

(e) $\mu = 10^{-4}$

(f) $c = 0.0125$

Figure 4.7.: On the left the speedup plots for the diffusion problem and on the right for the advection problem are depicted. The dispersion relation number decreases from top to bottom. Only speedups for certain Fourier modes in space are plotted. For $P = 1$ MLSDC was used to compute the speedup. Each main speedup line has branching lines, showing the speedup stemming from distributing the workload of the main branch to fewer processes P.

5. Outlook

> Casus ubique valet; semper tibi pendeat hamus. Quo minime credas gurgite, piscis erit.
>
> Ovid

Although we have already investigated many aspects of PFASST in the previous chapters, some elemental aspects and ideas are still waiting to be explored. We will now conclude our investigation of PFASST by exploring a number of practical and theoretical ideas. In the first section we will begin by introducing some practical ideas, such as time coarsening and multi-level PFASST, along with some proof-of-concept results for these practical ideas. Developing these ideas is primarily a question of implementation in a high performance environment. Subsequently, we will present a second set of ideas and open questions, which are of a theoretical nature. The success of these ideas is rather a question of mathematical proficiency. Finally, we conclude the dissertation with a summary of the work presented in the previous chapters.

5.1. Future work

In this section, practical ideas are developed to enhance PFASST in the direction of multigrid using the matrix formulation. We begin by introducing a new coarsening strategy.

5.1.1. Time coarsening

Originally, we only presented coarsening in space, since this is the form of PFASST found in the literature. In Section 1.2, we find two successful examples, in which the combinations of coarsening in time and space is employed by multigrd methods. The first example is the space-time multigrid by Horten and Vandewalle [60]. The second example is the space-time multigrid by Neumüller and Gander [63]. Both methods adjust their coarsening strategy to the dispersion relation number of the problem on the respective level. We will now outline several ways to introduce coarsening in time. The first option is to use less quadrature nodes on the coarser level, while keeping the number of subintervals constant. This option is limited to a few levels, since the number of quadrature nodes is usually not greater than 10. The second option is to reduce the

number of subintervals on the coarser levels. To this end, we take all time nodes on the fine level and coarse level with a coarsening factor of 2

$$\{\tau_1, \tau_2, \ldots, \tau_M, \Delta t + \tau_1, \ldots, \Delta t + \tau_M, \ldots, \Delta t(L-1) + \tau_M\} \quad \text{and}$$
$$\{2\tau_1, 2\tau_2, \ldots, 2\tau_M, 2\Delta t + 2\tau_1, \ldots, 2\Delta t + 2\tau_M, \ldots, 2\Delta t(L/2-1) + 2\tau_M\}.$$

Next, we define the interpolation and restriction between the two sets as $\hat{\mathbf{T}}_C^F$ and $\hat{\mathbf{T}}_F^C$. To construct the interpolation operator, we proceed as follows: Each row contains the values needed to get the interpolation value of the respective time point t_j. Depending on the order of the interpolation, l, we find the l next neighbors among the time points on the coarser level. Next, we construct the Lagrange polynomials according to Equation (1.14). Then, we fill the row of the operator with either 0, when the respective time point on the coarser level $\tilde{t}_i$ is not a next neighbor, or with $\ell_i(t_j)$ when $\tilde{t}_i$ is a next neighbor of t_j. This is done for each time point on the fine level t_j and each row of the operator $\hat{\mathbf{T}}_C^F$ respectively. The restriction operator $\hat{\mathbf{T}}_F^C$ is constructed in a similar fashion, by finding next neighbors of $\tilde{t}_i$ in the set of time points on the fine level.

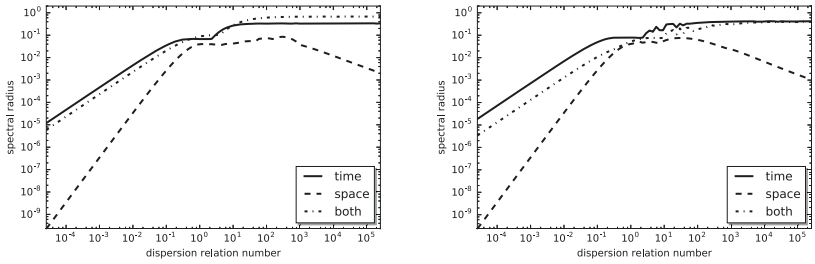
With this approach, the communication costs of PFASST may be increased because some time points on the coarse level are found on a subinterval, which is treated by another set of processes. It is possible to combine the different coarsening strategies with one another. In our examples, we will only use coarsening in the number of subintervals with a coarsening factor of 2; the coarsening in space, as it is used throughout this work, and the combination of both.

To implement time coarsening for PFASST, we simply substitute the interpolation and restriction operator

$$\mathbf{T}_C^F = \mathbf{I}_M \otimes \bar{\mathbf{T}}_C^F \text{ and } \mathbf{T}_F^C = \mathbf{I}_M \otimes \bar{\mathbf{T}}_F^C \quad \text{by} \quad \mathbf{T}_C^F = \hat{\mathbf{T}}_C^F \otimes \bar{\mathbf{T}}_C^F \text{ and } \mathbf{T}_F^C = \hat{\mathbf{T}}_F^C \otimes \bar{\mathbf{T}}_F^C.$$

This confirms the advantage of the implementation of PFASST in matrix formulation, introduced in this work, over the implementation in algorithmic form.

In Figure 5.1, we find the spectral radius of the PFASST iteration matrix for the 3 different coarsening strategies for the diffusion problem plotted over the dispersion relation number. We observe that the difference between the configurations with high and low interpolation order in Figure 5.1a and 5.1b is rather small. The biggest difference is found for the spatial coarsening configuration for the highest dispersion relation number with $2 \cdot 10^{-3}$ for a low interpolation order and around $1 \cdot 10^{-3}$ for the high interpolation order. This would roughly half the number of iterations needed, when the interpolation order is high enough. In all cases, the spatial coarsening strategy yields the smaller spectral radii. This shows that PFASST behaves contrary to the space-time multigrid methods found in [60, 63]. Therefore, the results do not indicate that a combination of the strategies would improve the convergence of PFASST. On the other hand, the computational costs are reduced, which could have a positive effect on the parallel



(a) low interpolation order

(b) high interpolation order

Figure 5.1.: Spectral radius of an iteration matrix stemming from different PFASST setups for the diffusion problem with 4 subintervals. The setups of PFASST differ in the coarsening strategies used. Coarsening in time, space and in both directions is employed. Furthermore, two different interpolation orders for both directions are used. For the spatial dimension interpolation orders 1 and 7 are used, whereas in the temporal dimension 1 and 5 were used. Time coarsening is performed by the reduction of subintervals.

performance of PFASST.

It is a goal for the future to assess the parallel performance of PFASST for these new coarsening strategies. Therefore, it is important to find ways to distribute the workload for these strategies. Another goal is the investigation of different interpolation strategies on the convergence of PFASST. Since the spectral radius is only a mediocre indicator for the convergence of PFASST as we pointed out in Section 3.1.3, it is advisable to use a better indicator for the study of these novel coarsening strategies. Altogether, novel coarsening and interpolation strategies for PFASST are an interesting study subject for the future.

5.1.2. Multi-level and inexact PFASST

In this section, we will exploit the matrix formulation of PFASST a little further, by exchanging the spatial solver and introducing additional levels. Exchanging the spatial solver is done by substituting

$$\mathbf{P}_{[t_l,t_{l+1}]} = \mathbf{I} - \Delta t \mathbf{Q}_\Delta \otimes \mathbf{A} \quad \text{by} \quad \mathbf{P}_{[t_l,t_{l+1}]} = \mathbf{I} - \Delta t \mathbf{Q}_\Delta \otimes \hat{\mathbf{A}},$$

where $\hat{\mathbf{A}}$ is a preconditioning matrix for the spatial system matrix. On the coarsest level we will keep $\tilde{\mathbf{A}}$. The multi-level PFASST is constructed recursively, by substituting

the preconditioning matrix of coarse grid correction with the preconditioning matrix of another PFASST iteration.

In Figure 5.2, the convergence plots for both enhancements are depicted. These plots are similar to the convergence plots in Chapter 3. On the left, we find the two level version and on the right the five-level version of PFASST. We observe that the five-level version is slightly worse than the two-level version. Furthermore, we observe that the convergence behavior is greatly dependent on the spatial solver. The more accurate the spatial solver, the better the convergence behavior of PFASST.

These results are only a fraction of what has to be studied for a more complete picture. For a fair comparison between the exact, the Gauß-Seidel, and the weighted Jacobi spatial solver, one has to consider the parallel performance, as well as other problems besides the diffusion problem.

Note that in [99], this idea was already investigated to some extend. In particular, a multigrid solver was employed for the spatial problem. This yields convergence results which range between the convergence results of Figures 5.2a and 5.2c.

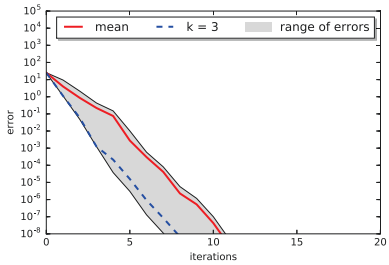
5.1.3. Stability and smoothing properties

In Section 1.1.1, we discussed the stability of Parareal. The stability property of the time stepping method is defined by its ability to solve the test equation. Since the test equation has no spatial dimension, it is not possible to define a coarser level in space. Therefore, it is not possible to apply the PFASST version with spatial coarsening to the test equation, and the stability property is not applicable in this form.

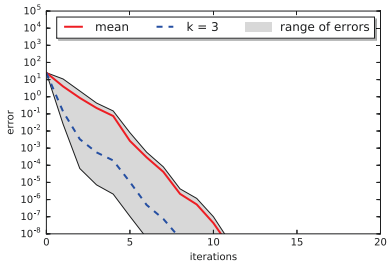
Instead, we briefly study the stability of the approximative block Jacobi solver in the weighted form

$$\boldsymbol{U}^{k+1} = \boldsymbol{U}^k + \omega \hat{\mathbf{P}}^{-1} \left(\boldsymbol{c} - \mathbf{M}\boldsymbol{U}^k \right),$$

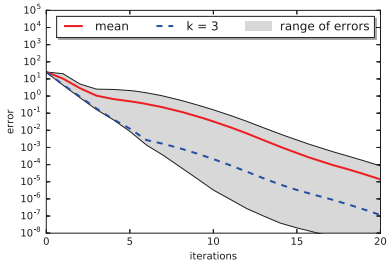
similar to the approximative block Jacobi solver in Lemma 3, which is a part of PFASST. In Figure 5.3, we used the matrix symbols introduced in Section 2.3.1 to generate the stability plots with and without the assumption of periodicity in time. More precisely, we plot the spectral radius of the iteration matrix of the weighted block Jacobi for the coefficient ρ of the test equation. We use three different weights ω for the block Jacobi solver. Since the optimal convergence of a classical multigrid method depends on the right choice of ω, it is worth a try to see if an adjusted weight improves the convergence properties of our approximative block Jacobi solver. For the classical weighted Jacobi solver and the standard diffusion problem, $\omega = 2/3$ is an optimal choice. In this case, high frequency modes are reduced most effectively by the weighted Jacobi solver. Neumüller and Gander use $\omega = 1/2$ with good results. This is not the case for the approximative block Jacobi solver used by PFASST. For $\omega = 1$, we observe a stronger error reduction than for $\omega = 2/3$ or $\omega = 1/2$, compare Figure 5.3a with Figure 5.3c or 5.3e. When we



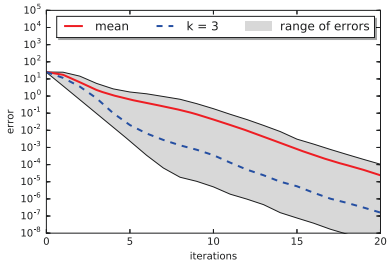
(a) exact, two-level



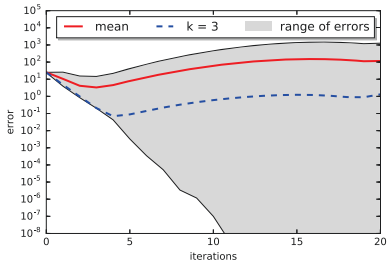
(b) exact, five-level



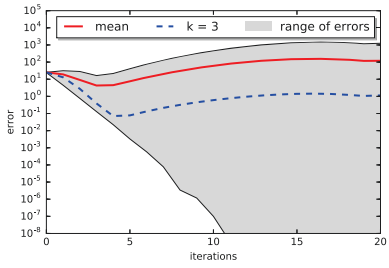
(c) Gauß-Seidel, two-level



(d) Gauß-Seidel, five-level



(e) Jacobi, two-level



(f) Jacobi, five-level

Figure 5.2.: Convergence behavior of PFASST for the diffusion problem ($\mu = 5 \cdot 10^{-1}$) with 4 subintervals, two and five levels, and 3 different spatial solver. As spatial solvers we employ: solving the spatial problem directly, one Gauß-Seidel iteration step or one Jacobi iteration step weighted with $\omega = 2/3$. In all setups, a spatial interpolation of order 4 is used. The gray area indicates possible errors resulting from different initial values of the form $\sin(2\pi kx)$ for $k \in \{1, \ldots, 15\}$.

use the matrix symbols stemming from to the assumption of periodicity in time, the stability plots change significantly. The most concise difference is observed for $\rho \to 0$. This case may be related to $\Delta t \to 0$. For a consistent time stepping method we expect that the spectral radius also converges to 0. Only Figure 5.3a shows this consistency.

Figure 5.4 depicts the spectral radii of $\mathbf{M}_{[t_0,T]} \cdot \mathbf{T}_{\text{bJac}}^{\nu}$, where $\mathbf{T}_{\text{bJac}}^{\nu}$ is the iteration matrix of the approximative block Jacobi solver. Computing these spectral radii, provides an empirical way to study the smoothing property for the approximative block Jacobi solver. Note that the spectral radius is preferable over a matrix norm, since it is a lower bound for all matrix norms. More precisely, according to Gelfand's formula it holds

$$\forall \varepsilon > 0 \quad \exists N \in \mathbf{N} \quad \forall k \geq N \quad \rho(\mathbf{A}) - \varepsilon < \left\| \mathbf{A}^k \right\|^{\frac{1}{k}} < \rho(\mathbf{A}) + \varepsilon,$$

for an arbitrary matrix $\mathbf{A}$. The smoothing property holds when

$$\left\| \mathbf{M}_{[t_0,T]} \cdot \mathbf{T}_{\text{bJac}}^{\nu} \right\| \leq \frac{\eta(\nu)}{h^{\alpha}} \quad \text{for all} \quad 1 \leq \nu < \bar{\nu}(h),$$

for a number α and with $\bar{\nu}(h) \to \infty$ for $h \to 0$ as well as $\eta(\nu) \to 0$ for $\nu \to \infty$. The functions η and $\bar{\nu}$ are independent of h. The value h may reflect the distance between grid points in different ways; for PFASST a suitable way has yet to be found. Independent of the way in which h is used, the process $h \to 0$ may be associated with a path on the stability plot running towards the origin. In Figure 5.4a, the stability plot in the case $\omega = 1$ is depicted. In this case we observe, at least empirically, that the spectral radii converges to 0 for $h \to 0$.

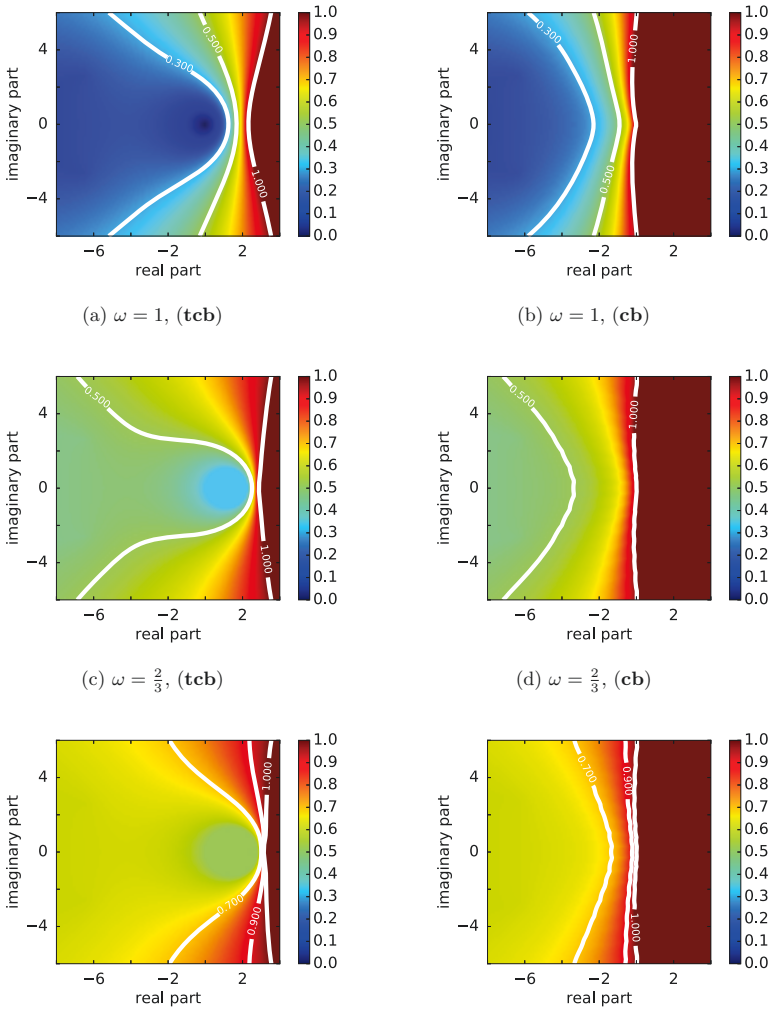
Furthermore, when we compare Figure 5.4b with Figure 5.4a, we observe that with every additional iteration of the smoother the spectral radii is reduced further. These first observations do not contradict the smoothing property. Thus, further investigations of the smoothing property are an interesting task for the future.

Another important property is the approximation property, which reads

$$\left\| \mathbf{M}_{[t_0,T]}^{-1} - \mathbf{T}_C^F \tilde{\mathbf{M}}_{[t_0,T]}^{-1} \mathbf{T}_F^C \right\| \leq C h^{\alpha},$$

for a constant C and the same α as for the smoothing property. When the approximation and smoothing property is fulfilled, this leads to a h independent convergence.

Note that these properties and the deduction of the h independent convergence are a part of the qualitative analysis, which was introduced in 1982 by Hackbusch in [100]. Since then, this qualitative analysis was successfully applied to a wide range of multigrid methods. Therefore, using this qualitative analysis for PFASST with the novel formulation of this work is another interesting subject for future work.



(a) $\omega = 1$, (**tcb**)

(b) $\omega = 1$, (**cb**)

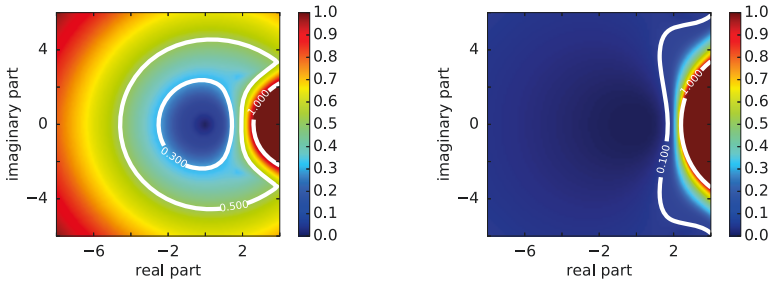
(c) $\omega = \frac{2}{3}$, (**tcb**)

(d) $\omega = \frac{2}{3}$, (**cb**)

(e) $\omega = \frac{1}{2}$, (**tcb**)

(f) $\omega = \frac{1}{2}$, (**cb**)

Figure 5.3.: Stability plots of the weighted block Jacobi smoother of PFASST with 4 subintervals and the weight ω. On the left we find the matrix symbols related to time collocation blocks (**tcb**) and on the right the matrix symbols related to collocation blocks (**cb**) were used.



(a) $\nu = 1$ (b) $\nu = 3$

Figure 5.4.: Spectral radii of $\mathbf{M}_{[t_0,T]} \cdot \mathbf{T}_{\text{bJac}}^{\nu}$ with 4 subintervals and ν iterations.

5.1.4. Open questions and theoretical ideas

In the next section, we explore a multitude of open questions and theoretical ideas encountered in the course of researching this work. The first ideas are inspired by the analysis of Parareal and modifications introduced in Section 1.1.2:

Symplecticity

It may be possible to achieve simplecticity for PFASST along the lines of the work by Bal in [46].

Adaptivity

There are numerous ways to increase the accuracy of PFASST in exchange for computational costs. For example, the number of spatial solver steps or the number of quadrature nodes used could be changed, which then could be used for an adaptive usage of PFASST.

Analysis

The analysis in this work is greatly influenced by multigrid theory. Alternatively, we could try to analyse PFASST with the means of time stepping theory. In the appendix we find the proof of Theorem 1. This proof reflects the iterative nature of Parareal and thus also of PFASST. Therefore, it could serve as a prototype for another analysis strategy of PFASST.

IMEX

The IMEX principle is easily implemented for PFASST, since we are able to employ IMEX-SDC; even multi-rate SDC versions are possible, cf. [76].

The final idea in this list brings us to the idea of substituting SDC with a iterative time stepping method. For example, in [101] an iterative and parallel solver for one Runge-Kutta step is introduced. It would be of interest to study if the properties of the Runge-Kutta method transfer to the resulting PFASST method. It is also of interest to compare SDC to these Runge-Kutta methods when they are a part of PFASST.

The interpretation and analysis of multigrid methods from the Fourier perspective is a very classical one. Representatives of this perspective are the works by Hackbusch and Brandt. New interpretations and theories have emerged since the works of Hackbusch and Brandt. Examples are the half space analysis by Diskin in [102], the subspace splitting by Bramble in [103], and the truncated multigrid methods by Gräser in [104]. These novel theories and methods have one principle in common: they employ a functional analysis perspective instead of a Fourier perspective. This new perspective could also be beneficial for the understanding of PFASST. It would also be an interesting attempt to study PFASST along the lines of the wave form relaxation theory by Lubich and Ostermann in [59].

To study PFASST in asymptotic cases, such as $\Delta t \to 0$, $\Delta x \to 0$ or $L \to \infty$, would also be of great interest. Take, for example, the matrix symbol of the approximative block Jacobi solver according to Theorem 6, which reads on the diagonal

$$\begin{aligned}
&\mathbf{I} - \left(\mathbf{B}^{(P)}(\theta)\right)^{-1} \mathbf{B}^{(M)}(\theta) \\
=&\mathbf{I} - \left(\mathbf{I}_L \otimes \mathbf{I}_M - \Delta t \cdot \lambda^{(A)}(\theta)\mathbf{I}_L \otimes \mathbf{Q}_{\boldsymbol{\Delta}}\right)^{-1} \left(\mathbf{I}_L \otimes \mathbf{I}_M - \mathbf{E} \otimes \mathbf{N} - \Delta t \cdot \lambda^{(A)}(\theta)\mathbf{I}_L \otimes \mathbf{Q}\right) \\
=&\left(\mathbf{I}_L \otimes \mathbf{I}_M - \Delta t \cdot \lambda^{(A)}(\theta)\mathbf{I}_L \otimes \mathbf{Q}_{\boldsymbol{\Delta}}\right)^{-1} \left(\mathbf{E} \otimes \mathbf{N} + \Delta t \cdot \lambda^{(A)}(\theta)\mathbf{I}_L \otimes (\mathbf{Q} - \mathbf{Q}_{\boldsymbol{\Delta}})\right).
\end{aligned}$$

We can show that this term converges towards $\mathbf{E} \otimes \mathbf{N}$ for $\Delta t \to 0$, independently of the eigenvalues $\lambda^{(A)}(\theta)$. Since $\mathbf{E} \otimes \mathbf{N}$ is a strictly lower triangular matrix, the spectral radius is 0. Therefore, the block Jacobi solver yields the exact solution after one step in this asymptotic case. This confirms the observation made in Figure 5.3a. It seems feasible to equip this procedure with more mathematical thoroughness, then apply it to the remaining matrix symbols of PFASST and produce convergence results for the asymptotic cases $\Delta t \to 0$ and $\Delta x \to 0$. For the case $L \to \infty$, the matrix symbols relating to the time collocation blocks are of increasing size. In contrast, the size of the matrix symbols relating to collocation blocks remains constant despite an increasing number of subintervals L. Thus, making them an option for the study of the asymptotic case $L \to \infty$. This option requires some sort of asymptotic equivalence, as introduced in [88], between the iteration matrices with and without the assumption of periodicity in time. More ideas to treat this asymptotic case may be found in [105].

5.2. Conclusion

The starting point of this work was PFASST, an interesting and highly developed parallel-in-time method with a very scarce mathematical basis. The main goal of this work was to strengthen this mathematical basis. In Chapter 1, Parareal was introduced as a fundamental parallel-in-time method with such a solid mathematical basis. We outlined the steps necessary in the development from Parareal to PFASST. Then, several space-time multigrid methods were introduced in order to motivate the multigrid perspective and to present relating methods. Altogether, we presented ideas and methods from the field of parallel-in-time integration, some of which provided the basis for developing a deeper understanding of PFASST. Next, we introduced SDC as the basic building block of PFASST, and MLSDC as an intermediate step in the development of PFASST. Both of these methods were already introduced in matrix formulation, which was made possible through the assumption of a linear autonomous ordinary differential equation. This assumption is fundamental for this work. PFASST itself was first presented in algorithmic form on the basis of SDC and MLSDC, following the original papers. This form was then transformed into a novel matrix form in Chapter 2. To this end, first the composite collocation problem and then the approximative block Jacobi solver and the approximative block Gauß-Seidel solver were introduced. Both solvers are special stationary iterative solvers for the composite collocation problem. In Theorem 3, these two solvers were assembled to create PFASST. This gave us the new multigrid perspective needed for a better understanding of PFASST. Now we are able to describe PFASST in the same terms as other multigrid methods. In particular, we were able to identify the typical elements of multigrid. The approximative block Jacobi solver is the smoother, the approximative Gauß-Seidel solver is the coarse grid solver used for the coarse grid correction. Using these different elements, we formulated the iteration matrix of PFASST. The iteration matrix determines how an error vector is reduced after one iteration of PFASST; this represents the key element of the mathematical basis in this work. The logical next step was the study of this iteration matrix in Section 2.2, where we also decomposed this iteration matrix with and without the assumption of periodicity in time. Without the assumption the decomposition yields the time collocation blocks, which have twice the size of the degrees of freedom in the temporal direction. With the assumption of periodicity in time it yields the collocation blocks, which have the size of the number of quadrature points.

However, due to this new decomposed forms, we were able to observe the different spatial error modes individually, since each time collocation block relates to a harmonic pair of spatial error modes. With Lemma 5, we were further able to connect the decomposition of the iteration matrix with the special structure of the error vectors from Definition 4. This enabled us to separate the temporal part from the spatial part of the error vector, in turn we were able to treat the spatial part of the problem with established mathematical tools. In a nutshell, we separated the temporal and spatial part of

the iteration matrix, decomposed the spatial part into its fundamental elements, and, finally, established an efficient way to apply the iteration matrix to the error vectors in this particular decomposition. This served as a basis for the convergence studies in Chapter 3. In this chapter, we focused on two model problems, namely the diffusion and the advection problem. For both problems, we found representative setups. We applied PFASST to these problems and presented the first convergence behavior across different initial values. Then, we introduced 4 prediction strategies and evaluated them. We decided to compute the error vectors directly in the following sections, since this strategy is the most accurate of the four strategies. It is also the only *a posteriori* prediction strategy. We also ruled out the use of collocation blocks for most cases and studied the reliability of the eigenvalue computations using pseudospectra in Section 3.1.3.

In Section 3.2, we introduced two types of mode damping fields in order to visualize the influence of the temporal part of the initial error. In particular, the mode damping fields of the second type depict the effect of the constituents of PFASST on all space-time error modes. This revealed the underlying mechanism of PFASST, since it shows how the approximative block Jacobi and the coarse grid correction complement each other by reducing different error modes. On the basis of these mode damping fields, we studied the role of the dispersion relation numbers for the modes of operation of PFASST. For both problems, we found 3 representative dispersion relation numbers, which invoked 3 characteristic modes of operation of PFASST. With these characteristic modes in mind, we studied the parallel performance of PFASST in Chapter 4.

There, we briefly introduced Amdahl's and Gustafson's law and argued that PFASST will not obey both laws, since the execution of PFASST yields a non-fixed, usually unknown workload. The reason for this is that different numbers of iterations of PFASST are needed in order to achieve a certain accuracy for different setups. Next, we introduced two different parallelization strategies. The first strategy assumes that each set of processes is in charge of one subinterval, whereas the second strategy assigns several subintervals to one set of processes. For both strategies, we estimated the theoretical limits of the resulting speedup in Section 4.2. In a wide range of numerical experiments in Section 4.3, we introduced a way to estimate the speedup of PFASST by counting the number of iterations for SDC and PFASST. Additionally, we developed the idea of semi-randomized error vectors in order to assess the influence of the temporal part of the initial error vector. Finally, in Section 4.3.3, we presented the estimated speedups for both parallelization strategies, which showed that the parallel performance is highly dependent on the dispersion relation number of the problem and the initial error vector. This also demonstrates the importance of the theoretical framework developed in Chapter 2, since this framework made it feasible to study the different error modes individually and revealed the functionality of PFASST for different dispersion relation numbers. Throughout this process, we gathered ideas to enhance the performance, and to improve the understanding of PFASST. Some of these ideas were briefly presented in this chapter.

In summary, the main achievement of this work is a new perspective of the PFASST method, which enabled us to study the convergence and the parallel performance for model problems. Through this, we were able to strengthen the mathematical basis of PFASST significantly. In the future, this work can be the foundation for an even more profound and rigorous mathematical understanding of PFASST. Eventually, it may even inspire novel parallel-in-time methods.

Appendices

A. Proof of Theorem 1

Theorem (Lions, Maday and Turinici 2001)**.** *The Parareal scheme is of order k, i.e. it exists c_k so that*

$$\left|U_n^k - u\left(t_n\right)\right| + \max_{t\in[t_n,t_{n+1}]}\left|\mathcal{F}(t_n,t,U_n^k) - u\left(t\right)\right| \leq c_k \Delta t^k.$$

Proof. We proof by induction over the iterations k. Before we start with the case $k=1$, we denote

$$\mathcal{F}(t_n,t_{n+1},U_n^k) = u_n^k(t_{n+1}) = e^{-a\Delta t}U_n^k \text{ and } \mathcal{G}(t_n,t_{n+1},U_n^k) = u_n^k(t) = (1-\Delta t a)\,U_n^k,$$

since the fine propagator is exact and the coarse propagator is an explicit Euler step. Also we use the analytical solution of our simple problem $u(t) = e^{-at}u_0$. First we transform (1.2) according to the requirements

$$\begin{aligned}
U_{n+1}^{k+1} &= e^{-a\Delta t}U_n^k + (1-a\Delta t)U_n^{k+1} - (1-a\Delta t)U_n^k \\
&= \sum_{j=0}^{n}(1-\Delta t)^{n-j}\left(U_{n-i}^k\left(e^{-a\Delta t} - (1-a\Delta t)\right)\right) + (1-a\Delta t)^{n+1}\,u_0 \\
&= e^{-a\Delta t}U_n^k + \sum_{j=0}^{n}(1-\Delta t)^{n-j}\left(e^{-a\Delta t}U_j^k - U_{j+1}^k\right).
\end{aligned}$$

In the case of $k=1$, we use the starting iteration values

$$U_n^0 = (1-a\Delta t)^n u_0,$$

which are generated by the explicit Euler scheme. Then the first iteration of Parareal reads

$$\begin{aligned}
U_{n+1}^1 &= (n+1)(1-a\Delta t)^n(e^{-a\Delta t} - 1 + a\Delta t)u_0 + (1-a\Delta t)^{n+1}u_0 \\
&= (1-a\Delta t)^n\left((n+1)(e^{-a\Delta t} - 1 + a\Delta t) + (1-a\Delta t)\right)u_0.
\end{aligned}$$

With the use of the Taylor series it holds

$$\begin{aligned} U_{n+1}^1 - u(t_{n+1}) &= (1-a\Delta t)^n \left((n+1)(e^{-a\Delta t} - 1 + a\Delta t) + (1-a\Delta t) - \frac{e^{-(n+1)a\Delta t}}{(1-a\Delta t)^n} \right) u_0 \\ &= (1-a\Delta t)^n \left((n+1) \sum_{j=2}^{\infty} \frac{(-a\Delta t)^j}{j!} + \frac{(1-a\Delta t)^{n+1} - \sum_{j=0}^{\infty} \frac{(-a\Delta t)^j}{j!}}{(1-a\Delta t)^n} \right) u_0. \end{aligned}$$

In this form and with the assumption that $a\Delta t$ is sufficiently small, we get

$$\left| U_{n+1}^1 - u(t_{n+1}) \right| \leq (L+1) C_1 \Delta t^2 u_0 + C_2 \Delta t u_0 \leq C_3 \Delta t.$$

For the second part of the inequality it holds

$$\begin{aligned} \left| u_n^1(t) - u(t) \right| &= \left| (1-\Delta t a)^n \left(e^{-(t-n\Delta t)a} \right) u_0 - e^{-ta} u_0 \right| \\ &= \left| \left(\left((1-a\Delta t)\, e^{a\Delta t} \right)^n - 1 \right) e^{-ta} u_0 \right| \qquad \leq C \Delta t e^{-ta} u_0, \end{aligned}$$

for $t \in [t_n, t_{n+1}]$. Now we assume that the proposition is true for the k-th Parareal iteration and denote $\epsilon_j^k = U_j^k - u(t_j)$, then it holds

$$\begin{aligned} \left| U_{n+1}^{k+1} - u(t_{n+1}) \right| &= \left| \sum_{j=0}^{n-1} (1-a\Delta t)^{n-j} \left(e^{-a\Delta t} \epsilon_j^k - \epsilon_{j+1}^k \right) + e^{-a\Delta t} \epsilon_n^k \right| \\ &= \left| \sum_{j=0}^{n-1} (1-a\Delta t)^{n-j} \left(e^{-a\Delta t} \epsilon_j^k \right) - \sum_{j=1}^{n} (1-a\Delta t)^{n-j+1} \left(\epsilon_j^k \right) + e^{-a\Delta t} \epsilon_n^k \right| \\ &= \left| \sum_{j=0}^{n} (1-a\Delta t)^{n-j} \epsilon_j^k \left(e^{-a\Delta t} - (1-a\Delta t) \right) \right| \leq c_k \Delta t^{k+1} \sum_{j=0}^{n} (1-a\Delta t)^j \end{aligned}$$

Finally, this yields

$$\begin{aligned} \left| u_n^{k+1}(t) - u(t) \right| &= \left| e^{-(t-\Delta t n)a} U_n^{k+1} - u(t) \right| \\ &= \left| e^{-(t-\Delta t n)a} (U_n^{k+1} - u(t_n) + u(t_n)) - u(t) \right| \\ &= e^{-(t-\Delta t n)a} \left| \epsilon_n^{k+1} \right| \leq c_{k+1} \Delta t^{k+1}. \end{aligned}$$

for $t \in [t_n, t_{n+1}]$. □

Bibliography

[1] D. Atkins, S. Baker, et al. National science foundation advisory committee for cyberinfrastructure task force on data and visualization final report. *National Science Foundation*, 2011.

[2] Kevin Burrage. Parallel methods for ODEs. *Advances in Computational Mathematics*, 7:1–3, 1997.

[3] A. Iserles and S.P. Nørsett. On the theory of parallel Runge–Kutta methods. *IMA Journal of numerical Analysis*, 10(4):463–488, 1990.

[4] J.C. Butcher. Order and stability of parallel methods for stiff problems. *Advances in Computational Mathematics*, 7(1):79–96, 1997.

[5] A. J. Christlieb, C. B. Macdonald, and B. W. Ong. Parallel high-order integrators. *SIAM Journal on Scientific Computing*, 32(2):818–835, 2010.

[6] M. J. Gander and S. Güttel. PARAEXP: A parallel integrator for linear initial-value problems. *SIAM Journal on Scientific Computing*, 35(2):C123–C142, 2013.

[7] Y. Maday and E. M. Rønquist. Parallelization in time through tensor-product space-time solvers. *Comptes Rendus Mathematique*, 346(1–2):113 – 118, 2008.

[8] D. Sheen, I. H. Sloan, and V. Thomée. A parallel method for time discretization of parabolic equations based on Laplace transformation and quadrature. *IMA Journal of Numerical Analysis*, 23(2):269–299, 2003.

[9] M. J. Gander. A waveform relaxation algorithm with overlapping splitting for reaction diffusion equations. *Numerical Linear Algebra with Applications*, 6(2):125–145, 1999.

[10] S. Vandewalle and D. Roose. The parallel waveform relaxation multigrid method. *Parallel Processing for Scientific Computing*, pages 152–156, 1989.

[11] J. Nievergelt. Parallel methods for integrating ordinary differential equations. *Commun. ACM*, 7(12):731–733, 1964.

[12] P. Chartier and B. Philippe. A parallel shooting technique for solving dissipative ODE's. *Computing*, 51(3-4):209–236, 1993.

[13] A. Bellen and M. Zennaro. Parallel algorithms for initial-value problems for difference and differential equations. *Journal of Computational and Applied Mathematics*, 25(3):341 – 350, 1989.

[14] J. Lions, Y. Maday, and G. Turinici. A "Parareal" in time discretization of PDEs. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 332(7):661–668, 2001.

[15] M. J. Gander, YL. Jiang, and RJ. Li. Parareal Schwarz waveform relaxation methods. In *Domain Decomposition Methods in Science and Engineering XX*, volume 91 of *Lecture Notes in Computational Science and Engineering*, pages 451–458. Springer Berlin Heidelberg, 2013.

[16] M. L. Minion. A hybrid Parareal spectral deferred corrections method. *Communications in Applied Mathematics and Computational Science*, 5(2):265–301, 2010.

[17] M. Emmett and M. L. Minion. Toward an efficient parallel in time method for partial differential equations. *Communications in Applied Mathematics and Computational Science*, 7:105–132, 2012.

[18] R. Speck, D. Ruprecht, M. Emmett, M. Bolten, and R. Krause. A space-time parallel solver for the three-dimensional heat equation. In *Parallel Computing: Accelerating Computational Science and Engineering (CSE)*, volume 25 of *Advances in Parallel Computing*, pages 263 – 272. IOS Press, 2014.

[19] M. Emmett and M. L. Minion. Efficient implementation of a multi-level parallel in time algorithm. In *Proceedings of the 21st International Conference on Domain Decomposition Methods*, Lecture Notes in Computational Science and Engineering, 2012.

[20] M. J. Gander and S. Vandewalle. Analysis of the Parareal time-parallel time-integration method. *SIAM Journal on Scientific Computing*, 29(2):556–578, 2007.

[21] L. Euler. *Institutionum calculi integralis*, volume 1. imp. Acad. imp. Saènt., 1768.

[22] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I*, volume 8. 1993.

[23] G. Wanner and E. Hairer. *Solving Ordinary Differential Equations II*, volume 1. Springer-Verlag, Berlin, 1991.

[24] M. J. Gander and S. Vandewalle. On the superlinear and linear convergence of the Parareal algorithm. In *Domain Decomposition Methods in Science and Engineering*, volume 55 of *Lecture Notes in Computational Science and Engineering*, pages 291–298. Springer Berlin Heidelberg, 2007.

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

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

[27] G. A. Staff and E. M. Rønquist. Stability of the Parareal algorithm. In *Domain Decomposition Methods in Science and Engineering*, volume 40 of *Lecture Notes in Computational Science and Engineering*, pages 449–456, Berlin, 2005. Springer.

[28] G. Bal. On the convergence and the stability of the Parareal algorithm to solve partial differential equations. In *Domain decomposition methods in science and engineering*, pages 425–432. Springer, 2005.

[29] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zrah. Parallel-in-time molecular-dynamics simulations. *Phys. Rev. E*, 66:057701, 2002.

[30] Y. Maday and G. Turinici. A Parareal in time procedure for the control of partial differential equations. *Comptes Rendus Mathmatique*, 335(4):387–392, 2002.

[31] M. Sarkis, C. E. Schaerer, and T. Mathew. Block diagonal Parareal preconditioner for parabolic optimal control problems. In *Domain Decomposition Methods in Science and Engineering XVII*, volume 60 of *Lecture Notes in Computational Science and Engineering*, pages 409–416. Springer Berlin Heidelberg, 2008.

[32] A. Lapin and A. Romanenko. Udzawa-type iterative method with Parareal preconditioner for a parabolic optimal control problem. *IOP Conference Series: Materials Science and Engineering*, 158(1):012059, 2016.

[33] X. Du, M. Sarkis, C. E. Schaerer, and D. B. Szyld. Inexact and truncated Parareal-in-time Krylov subspace methods for parabolic optimal control problems. *Electrontic Transactions on Numerical Analysis*, 40:36–57, 2013.

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

[35] Y. Maday and G. Turinici. Parallel in time algorithms for quantum control: Parareal time discretization scheme. *Int. J. Quant. Chem.*, 93(3):223–228, 2003.

[36] J. M. F. Trindade and J. C. F. Pereira. Parallel-in-time simulation of the unsteady Navier-Stokes equations for incompressible flow. *International Journal for Numerical Methods in Fluids*, 45(10):1123–1136, 2004.

[37] P. F. Fischer, F. Hecht, and Y. Maday. A Parareal in time semi-implicit approximation of the Navier-Stokes equations. In *Domain Decomposition Methods in Science and Engineering*, volume 40 of *Lecture Notes in Computational Science and Engineering*, pages 433–440, Berlin, 2005. Springer.

[38] J. M. F. Trindade and J. C. F. Pereira. Parallel-in-time simulation of two-dimensional, unsteady, incompressible laminar flows. *Numerical Heat Transfer, Part B: Fundamentals*, 50(1):25–40, 2006.

[39] D. Guibert and D. Tromeur-Dervout. Parallel deferred correction method for CFD problems. In *Parallel Computational Fluid Dynamics 2006*, pages 131 – 138. Amsterdam, 2007.

[40] Y. Liu and J. Hu. Modified propagators of Parareal in time algorithm and application to Princeton Ocean model. *Int. J. for Numerical Methods in Fluids*, 57(12):1793–1804, 2008.

[41] J. Steiner, D. Ruprecht, R. Speck, and R. Krause. Convergence of Parareal for the Navier-Stokes equations depending on the Reynolds number. In *Numerical Mathematics and Advanced Applications - ENUMATH 2013*, volume 103 of *Lecture Notes in Computational Science and Engineering*, pages 195–202. Springer International Publishing, 2015.

[42] LP. He and M. He. Parareal in time simulation of morphological transformation in cubic alloys with spatially dependent composition. *Communications in Computational Physics*, 11:1697–1717, 2012.

[43] J. M. Reynolds-Barredo, D. E. Newman, R. Sanchez, and L. A. Berry. Modelling Parareal convergence in 2D drift wave plasma turbulence. In *High Performance Computing and Simulation (HPCS), 2012 International Conference on*, pages 726–727, 2012.

[44] M. Duarte, M. Massot, and S. Descombes. Parareal operator splitting techniques for multi-scale reaction waves: Numerical analysis and strategies. *ESAIM: Mathematical Modelling and Numerical Analysis*, 45:825–852, 8 2011.

[45] D. S. Daoud. Stability of the Parareal time discretization for parabolic inverse problems. In *Domain Decomposition Methods in Science and Engineering XVI*, volume 55 of *Lecture Notes in Computational Science and Engineering*, pages 275–282. Springer Berlin Heidelberg, 2007.

[46] G. Bal and Q. Wu. Symplectic Parareal. In *Domain Decomposition Methods in Science and Engineering XVII*, volume 60 of *Lecture Notes in Computational Science and Engineering*, pages 401–408. Springer Berlin Heidelberg, 2008.

[47] D. Guibert and D. Tromeur-Dervout. Parallel adaptive time domain decomposition for stiff systems of ODEs/DAEs. *Computers & Structures*, 85(9):553 – 562, 2007.

[48] B. Lepsa and A. Sandu. An efficient error control mechanism for the adaptive 'Parareal' time discretization algorithm. In *Proceedings of the 2010 Spring Simulation Multiconference*, pages 87:1–87:7, San Diego, CA, USA, 2010. Society for Computer Simulation International.

[49] Z. Wang and SL. Wu. Parareal algorithms implemented with IMEX Runge–Kutta methods. *Mathematical Problems in Engineering*, 2015.

[50] M. L. Minion and S. A. Williams. Parareal and spectral deferred corrections. In *AIP Conference Proceedings*, volume 1048, page 388, 2008.

[51] R. P. Fedorenko. A relaxation method for solving elliptic difference equations. *USSR Computational Mathematics and Mathematical Physics*, 1(4):1092–1096, 1962.

[52] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, 2000.

[53] T. C. Kelley. Iterative methods for linear and nonlinear equations. *Raleigh N. C.: North Carolina State University*, 1995.

[54] W. L. Briggs, S. F. McCormick, et al. *A multigrid tutorial*. SIAM, 2000.

[55] A. Brandt. Multi-level adaptive techniques (MLAT) for singular-perturbation problems. 1978.

[56] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of computation*, 31(138):333–390, 1977.

[57] K. J. Brabazon, M. E. Hubbard, and P. K. Jimack. Nonlinear multigrid methods for second order differential operators with nonlinear diffusion coefficient. *Computers & Mathematics with Applications*, 68(12):1619–1634, 2014.

[58] W. Hackbusch. Parabolic multigrid methods. *Computing Methods in Applied Sciences and Engineering, VI*, pages 189–197, 1984.

[59] Ch. Lubich and A. Ostermann. Multi-grid dynamic iteration for parabolic equations. *BIT Numerical Mathematics*, 27(2):216–234, 1987.

[60] G. Horton and S. Vandewalle. A space-time multigrid method for parabolic partial differential equations. *SIAM Journal on Scientific Computing*, 16(4):848–864, 1995.

[61] R. D. Falgout, S. Friedhoff, Tz. V. Kolev, S. P. MacLachlan, and J. B. Schroder. Parallel time integration with multigrid. *SIAM Journal on Scientific Computing*, 36(6):C635–C661, 2014.

[62] V. Dobrev, Tz. Kolev, N. A. Petersson, and J. Schroder. Two-level convergence theory for parallel time integration with multigrid. *SIAM Journal on Scientific Computing, LLNL-JRNL*, 692418, 2016.

[63] M. J. Gander and M. Neumüller. Analysis of a new space-time parallel multigrid algorithm for parabolic problems. *SIAM Journal on Scientific Computing*, 38(4):A2173–A2208, 2016.

[64] M. Bolten, D. Moser, and R. Speck. A multigrid perspective on the parallel full approximation scheme in space and time. *arXiv preprint arXiv:1603.03586*, 2016.

[65] J. Berrut and L. N. Trefethen. Barycentric Lagrange interpolation. *SIAM review*, 46(3):501–517, 2004.

[66] A. Dutt, L. Greengard, and V. Rokhlin. Spectral deferred correction methods for ordinary differential equations. *BIT Numerical Mathematics*, 40(2):241–266, 2000.

[67] R. Frank and Ch. W. Ueberhuber. Iterated defect correction for the efficient solution of stiff systems of ordinary differential equations. *BIT Numerical Mathematics*, 17(2):146–159, 1977.

[68] J. Huang, J Jia, and M. L. Minion. Accelerating the convergence of spectral deferred correction methods. *Journal of Computational Physics*, 214(2):633 – 656, 2006.

[69] A. T. Layton and M. L. Minion. Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics. *Journal of Computational Physics*, 194(2):697 – 715, 2004.

[70] M. L. Minion. Semi-implicit projection methods for incompressible flow based on spectral deferred corrections. *Applied Numerical Mathematics*, 48(3–4):369 – 387, 2004. Workshop on Innovative Time Integrators for PDEs.

[71] A. Bourlioux, A. T. Layton, and M. L. Minion. High-order multi-implicit spectral deferred correction methods for problems of reactive flow. *Journal of Computational Physics*, 189(2):651 – 675, 2003.

[72] D. Guibert and D. Tromeur-Dervout. Parallel deferred correction method for CFD problems. In *Parallel Computational Fluid Dynamics 2006*, pages 131 – 138. Elsevier Science B.V., Amsterdam, 2007.

[73] M. Weiser. Faster SDC convergence on non-equidistant grids by DIRK sweeps. *BIT Numerical Mathematics*, 55(4):1219–1241, 2015.

[74] M. Winkel, R. Speck, and D. Ruprecht. A high-order Boris integrator. *Journal of computational physics*, 295:456–474, 2015.

[75] M. L. Minion. Semi-implicit spectral deferred correction methods for ordinary differential equations. *Communications in Mathematical Sciences*, 1(3):471–500, 2003.

[76] E. L. Bouzarth and M. L. Minion. A multirate time integrator for regularized stokeslets. *Journal of Computational Physics*, 229(11):4208–4224, 2010.

[77] D. Ketcheson and U. bin Waheed. A comparison of high-order explicit Runge–Kutta, extrapolation, and deferred correction methods in serial and parallel. *Communications in Applied Mathematics and Computational Science*, 9(2):175–200, 2014.

[78] R. Speck, D. Ruprecht, M. Emmett, M. L. Minion, M. Bolten, and R. Krause. A multi-level spectral deferred correction method. *BIT Numerical Mathematics*, 55(3):843–867, 2015.

[79] F. Koehler. PFASST TikZ. `https://github.com/Parallel-in-Time/pfasst-tikz`, 2015.

[80] S. Friedhoff and S. MacLachlan. A generalized predictive analysis tool for multigrid methods. *Numerical Linear Algebra with Applications*, 22(4):618–647, 2015.

[81] D. Hilbert and R. Courant. *Methoden der mathematischen Physik*. 1924.

[82] J. R. Cannon. *The one-dimensional heat equation*. Number 23. Cambridge University Press, 1984.

[83] G. K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, 2000.

[84] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*, volume 31. Cambridge University Press, 2002.

[85] C. Hirsch. *Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics*. Butterworth-Heinemann, 2007.

[86] X. Dai and Y. Maday. Stable Parareal in time method for first- and second-order hyperbolic systems. *SIAM Journal on Scientific Computing*, 35(1):A52–A78, 2013.

[87] T. Haut and B. Wingate. An asymptotic parallel-in-time method for highly oscillatory PDEs. *SIAM Journal on Scientific Computing*, 2014.

[88] R. M. Gray. *Toeplitz and Circulant Matrices: A Review*. now publishers inc, 2006.

[89] H. J. Landau. On Szegös eigenvalue distribution theorem and non-Hermitian kernels. *Journal d'Analyse Mathématique*, 28(1):335–357, 1975.

[90] J. M. Varah. On the separation of two matrices. *SIAM Journal on Numerical Analysis*, 16(2):216–222, 1979.

[91] S. K. Godunov. Spectral portraits of matrices and criteria of spectrum dichotomy. In *Computer Arithmetic and Enclosure Methods: Proc. Third IMACS-GAMM Symp. Computer Arithmetic and Scientific Computing (SCAN-91), L. Atanassova and J. Herzberger, eds., Amsterdam*, 1992.

[92] L. N. Trefethen and M. Embree. Pseudospectra gateway. *Web site. URL: http://web. comlab. ox. ac. uk/projects/pseudospectra*.

[93] T. Braconnier and N. J. Higham. Computing the field of values and pseudospectra using the Lanczos method with continuation. *BIT Numerical Mathematics*, 36(3):422–440, 1996.

[94] T. Braconnier, R. A. McCoy, and V. Toumazou. Using the field of values for pseudospectra generation. *European Centre for Research and Advanced Training in Scientific Computation (CERFACS), Toulouse, France, Tech. Rep. TR/PA/97/28*, 1997.

[95] T. G. Wright and L. N. Trefethen. Large-scale computation of pseudospectra using ARPACK and eigs. *SIAM Journal on Scientific Computing*, 23(2):591–605, 2001.

[96] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485. ACM, 1967.

[97] J. L. Gustafson. Reevaluating Amdahl's law. *Communications of the ACM*, 31(5):532–533, 1988.

[98] H. H. Ku. Notes on the use of propagation of error formulas. *Journal of Research of the National Bureau of Standards*, 70(4), 1966.

[99] M. L. Minion, R. Speck, M. Bolten, M. Emmett, and D. Ruprecht. Interweaving PFASST and parallel multigrid. *SIAM Journal on Scientific Computing*, 37(5):S244–S263, 2015.

[100] W. Hackbusch. Multi-grid convergence theory. In *Multigrid methods*, pages 177–219. Springer, 1982.

[101] P. J. van der Houwen and B. P. Sommeijer. Iterated Runge–Kutta methods on parallel computers. *SIAM Journal on Scientific and Statistical Computing*, 12(5):1000–1028, 1991.

[102] B. Diskin and J. L. Thomas. Half-space analysis of the defect-correction method for Fromm discretization of convection. *SIAM Journal on Scientific Computing*, 22(2):633–655, 2000.

[103] J. H. Bramble and X. Zhang. The analysis of multigrid methods. *Handbook of numerical analysis*, 7:173–415, 2000.

[104] C. Gräser, U. Sack, and O. Sander. Truncated non-smooth Newton multigrid methods for convex minimization problems. In *Domain Decomposition Methods in Science and Engineering XVIII*, pages 129–136. Springer, 2009.

[105] A. Böttcher and S. M. Grudsky. *Toeplitz matrices, asymptotic linear algebra, and functional analysis*. Birkhäuser, 2012.

Schriften des Forschungszentrums Jülich
IAS Series

Band / Volume 24
Automated Optimization Methods for Scientific Workflows in e-Science Infrastructures
S. Holl (2014), xvi, 182 pp
ISBN: 978-3-89336-949-2
URN: urn:nbn:de:0001-2014022000

Band / Volume 25
Numerical simulation of gas-induced orbital decay of binary systems in young clusters
A. C. Korntreff (2014), 98 pp
ISBN: 978-3-89336-979-9
URN: urn:nbn:de:0001-2014072202

Band / Volume 26
UNICORE Summit 2014
Proceedings, 24th June 2014 | Leipzig, Germany
edited by V. Huber, R. Müller-Pfefferkorn, M. Romberg (2014), iii, 60 pp
ISBN: 978-3-95806-004-3
URN: urn:nbn:de:0001-2014111408

Band / Volume 27
Automatische Erfassung präziser Trajektorien in Personenströmen hoher Dichte
M. Boltes (2015), xii, 308 pp
ISBN: 978-3-95806-025-8
URN: urn:nbn:de:0001-2015011609

Band / Volume 28
Computational Trends in Solvation and Transport in Liquids
edited by G. Sutmann, J. Grotendorst, G. Gompper, D. Marx (2015)
ISBN: 978-3-95806-030-2
URN: urn:nbn:de:0001-2015020300

Band / Volume 29
Computer simulation of pedestrian dynamics at high densities
C. Eilhardt (2015), viii, 142 pp
ISBN: 978-3-95806-032-6
URN: urn:nbn:de:0001-2015020502

Band / Volume 30
Efficient Task-Local I/O Operations of Massively Parallel Applications
W. Frings (2016), xiv, 140 pp
ISBN: 978-3-95806-152-1
URN: urn:nbn:de:0001-2016062000

Schriften des Forschungszentrums Jülich
IAS Series

Band / Volume 31
A study on buoyancy-driven flows: Using particle image velocimetry for validating the Fire Dynamics Simulator
by A. Meunders (2016), xxi, 150 pp
ISBN: 978-3-95806-173-6
URN: urn:nbn:de:0001-2016091517

Band / Volume 32
Methoden für die Bemessung der Leistungsfähigkeit multidirektional genutzter Fußverkehrsanlagen
S. Holl (2016), xii, 170 pp
ISBN: 978-3-95806-191-0
URN: urn:nbn:de:0001-2016120103

Band / Volume 33
JSC Guest Student Programme Proceedings 2016
edited by I. Kabadshow (2017), iii, 191 pp
ISBN: 978-3-95806-225-2
URN: urn:nbn:de:0001-2017032106

Band / Volume 34
Multivariate Methods for Life Safety Analysis in Case of Fire
B. Schröder (2017), x, 222 pp
ISBN: 978-3-95806-254-2
URN: urn:nbn:de:0001-2017081810

Band / Volume 35
Understanding the formation of wait states in one-sided communication
M.-A. Hermanns (2018), xiv, 144 pp
ISBN: 978-3-95806-297-9
URN: urn:nbn:de:0001-2018012504

Band / Volume 36
A multigrid perspective on the parallel full approximation scheme in space and time
D. Moser (2018), vi, 131 pp
ISBN: 978-3-95806-315-0
URN: urn:nbn:de:0001-2018031401

Weitere ***Schriften des Verlags im Forschungszentrum Jülich*** unter
http://wwwzb1.fz-juelich.de/verlagextern1/index.asp

IAS Series
Band / Volume 36
ISBN 978-3-95806-315-0

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
Forschungszentrum