An experimental comparison of a space-time multigrid method with PFASST for a reaction-diffusion problem

Pietro Benedusi[a,*], Michael L. Minion[b], Rolf Krause[a]

[a] *Euler Institute, Università della Svizzera italiana (USI), Lugano, Switzerland*
[b] *Lawrence Berkeley National Laboratory, Berkeley, CA, USA*

Abstract

We consider two parallel-in-time approaches applied to a (reaction) diffusion problem, possibly non-linear. In particular, we consider PFASST (Parallel Full Approximation Scheme in Space and Time) and space-time multigrid strategies. For both approaches, we start from an integral formulation of the continuous time dependent problem. Then, a collocation form for PFASST and a discontinuous Galerkin discretization in time for the space-time multigrid are employed, resulting in the same discrete solution at the time nodes. Strong and weak scaling of both multilevel strategies are compared for varying orders of the temporal discretization. Moreover, we investigate the respective convergence behavior for non-linear problems and highlight quantitative differences in execution times. For the linear problem, we observe that the two methods show similar scaling behavior with PFASST being more favorable for high order methods or when few parallel resources are available. For the non-linear problem, PFASST is more flexible in terms of solution strategy, while space-time multigrid requires a full non-linear solve.

Keywords: space-time multigrid, PFASST, parallel-in-time, DG discretization, strong and weak scalability, reaction-diffusion equation
2000 MSC: 65F10, 65L60, 65N55, 65M70, 65Y05

*Corresponding author
Email addresses: benedp@usi.ch (Pietro Benedusi), mlminion@lbl.gov (Michael L. Minion), rolf.krause@usi.ch (Rolf Krause)

1. Introduction

Since the clock frequency of computer processors has not increased significantly in the past fifteen years, an increase in computational performance for numerical algorithms can be achieved only by increasing parallelism, and modern supercomputers now contain many thousands of computing cores. Exploiting the capabilities of such massively parallel systems is not straightforward; algorithms with optimal complexity and excellent scalability must be designed to minimize the run-time of computationally intensive problems, such as the solution of time dependent partial differential equations (PDEs). When dealing with parallel solvers for discretized PDEs, the solution process is traditionally parallelized in space using domain decomposition techniques, until stagnation. Considering the technology trend, the traditional sequential time stepping will increasingly become the bottleneck for computational scalability for many applications. Hence, the development of new parallel methods that exploit concurrency in the time direction has become essential for time dependent problems. However, parallelization in time can be a challenging task, as, for many physical processes, the time direction is governed by a causality principle, with a preferential direction of information flow through the temporal domain, i.e. forward in time. Nevertheless, several new methods for temporal parallelization have been proposed in the last 20 years. For a more comprehensive review regarding the parallel-in-time literature of the past 50 years we refer to [1].

The objective of this work is to compare two of the most relevant recent approaches: PFASST [2] and space-time multigrid methods (STMG) [3, 4, 5, 6, 7, 8]. The current paper is similar in spirit to the comparison presented in [9] (where the authors suggest a future comparison with PFASST). Since many parallel-in-time methods are based on a coupling between coarse and fine time propagators, they can be framed in a multilevel-in-time setting; for example, MGRIT [5], Parareal in [10] or PFASST in [11]. Despite the similarities of these different approaches, the methods can behave quite differently on some problems. To date, the majority of papers on parallel in time methods investigate a single method, and very few have investigated the computational advantages and disadvantages of different methods on well defined benchmarks. Here we attempt to compare two methods using the same

spatial discretizations and solvers to emphasis the differences in scaling in the time direction.

In the remainder of this paper we present such a comparison between PFASST and STMG. In Section 2 we describe the respective time discretizations of the two approaches. In Section 3 we present a reaction-diffusion PDE and its discretization in space and time. In Section 4 we describe the solution methods that will be used in Section 5, where weak and strong scaling experiments are reported.

2. Preliminaries on the time discretizations

Let us introduce the time discretizations that we use for PFASST and the space-time multigrid, respectively. Both formulations are based on an integral form of the continuous problem and converge to the same collocation solution, if the same nodes are used (see e.g. [12, 13]). To illustrate these methods we consider the initial value problem with $U_0 \in \mathbb{R}$, on a single time step $I_n := [T_n, T_{n+1}] \subset \mathbb{R}$

$$u'(t) = f(u(t), t) \quad \text{for} \quad T_n < t < T_{n+1}, \qquad u(T_n) = U_0. \tag{2.1}$$

2.1. Collocation form

The PFASST algorithm is based on the spectral deferred correction (SDC) method, an iterative scheme introduced in [14] based on a collocation approximation of (2.1). Let us consider the Picard integral form of (2.1)

$$u(t) = U_0 + \int_{T_n}^{t} f(u(\tau), \tau) \mathrm{d}\tau, \tag{2.2}$$

and the M right Gauss-Radau nodes $\{t_m\}_{m=1}^{M}$ in I_n with $T_n < t_1 < t_2 < \cdots < t_M = T_{n+1}$. We approximate (2.2) by its collocation form, with $U_m \approx u(t_m)$:

$$\mathbf{U} = \mathbf{U}_0 + \Delta t \mathcal{Q} F(\mathbf{U}), \tag{2.3}$$

where $\Delta t := T_{n+1} - T_n$,

$$\mathbf{U} := [U_1, ..., U_M], \quad \mathbf{U}_0 := [U_0, ..., U_0], \quad F(\mathbf{U}) := [f(U_1, t_1), ..., f(U_M, t_M)], \tag{2.4}$$

$\mathcal{Q}$ is the $M \times M$ matrix $\mathcal{Q} := (q_{m,j})_{m,j=1}^{M}$ with the quadrature weights

$$q_{m,j} := \frac{1}{\Delta t} \int_{T_n}^{t_m} \ell_j(t) \mathrm{d}t,$$

and $\{\ell_j\}_{j=1}^{M}$ are the Lagrange polynomials at the M nodes. An SDC iteration can be considered as a preconditioned Richardson iteration to solve (2.3) (see, e.g. [13, 15]) and, if SDC converges, it is equivalent to an implicit RK method, with $q_{m,j}$ being the values in the corresponding Butcher tableaux. The resulting RK method is $A-$stable and has order of accuracy $2M-1$ for M Radau quadrature nodes [16].

2.2. Discontinuous Galerkin

Variational time-stepping methods are receiving increasing interest by the scientific community, especially in the context of adaptivity in space–time, for example in [17, 18]. Discontinuous Galerkin (DG) methods, in particular, have been widely used to discretize the time direction in the space-time setting as they ensure that the information flows in the positive time direction. They have been employed for a variety of problems such as convection/advection/diffusion equations or the Navier-Stokes equations. For example see the works [19, 20, 21, 22, 23, 24, 6, 25]. The use of DG discretization in time was first introduced in [12] for the discretization of a neutron transport equation. In this paper the authors show that, for finite elements of order q, the method is strongly A-stable, has convergence order $2q+1$ in the nodes, and is equivalent to an implicit (RK) time stepper with q intermediate steps. The first analysis on DG methods as time stepping techniques is provided by [26] and [27], followed by the work in [28, 29, 30]. More recently, specialized solution methods have been introduced, for example by [31, 32, 33, 34]. A *priori* and *posteriori* error analysis have been also provided, e.g. see [30, 17, 35, 36]. See [37] for a recent survey on the topic.

Let us consider the weak formulation of (2.1), where the continuity at T_n is weakly imposed, and its finite dimensional approximation with $u \approx U \in$

$\mathbb{P}_q(I_n)$ (i.e. the space of polynomials of degree q)[1]

$$\int_{T_n}^{T_{n+1}} U'(t)v(t)\,\mathrm{d}t + (U(T_n) - U_0)v(T_n) = \int_{T_n}^{T_{n+1}} f(U(t),t)v(t)\,\mathrm{d}t, \quad (2.5)$$

for all test functions $v \in \mathbb{P}_q(I_n)$. Equivalently, integrating by parts (2.5), we obtain the standard DG formulation:

$$-\int_{T_n}^{T_{n+1}} U(t)v'(t)\,\mathrm{d}t + U(T_{n+1})v(T_{n+1}) - v(T_n)U_0 = \int_{T_n}^{T_{n+1}} f(U(t),t)v(t)\,\mathrm{d}t, \quad (2.6)$$

where the *upwind flux* is given by the $v(T_n)U_0$ term. In the interval I_n, we construct the approximation U in the nodal form,

$$U(t) = \sum_{m=1}^{M} U_m \ell_{n,m}(t), \quad (2.7)$$

where $\{\ell_{n,m}\}_{m=1}^{M}$ is the basis of Lagrange polynomials of degree q at the $q+1 = M$ Gauss-Radau nodes in I_n. We can rewrite (2.6), using the approximation in (2.7) and the definitions in (2.4), as

$$\mathcal{K}_q \mathbf{U} = \mathcal{J}_q \mathbf{U}_0 + \mathcal{M}_q F(\mathbf{U}), \quad (2.8)$$

with $\mathcal{K}_q, \mathcal{M}_q, \mathcal{J}_q \in \mathbb{R}^{M\times M}$ given by

$$\mathcal{K}_q := \left[-\int_{T_n}^{T_{n+1}} \ell'_{n,i}(t)\ell_{n,j}(t)\,\mathrm{d}t + \ell_{n,i}(T_{n+1})\ell_{n,j}(T_{n+1}) \right]_{i,j=1}^{M}, \quad (2.9)$$

$$\mathcal{M}_q := \left[\int_{T_n}^{T_{n+1}} \ell_{n,i}(t)\ell_{n,j}(t)\,\mathrm{d}t \right]_{i,j=1}^{M}, \qquad \mathcal{J}_q := [\ell_i(T_n)\ell_j(T_{n+1})]_{i,j=1}^{M}. \quad (2.10)$$

Let us remark the similarity between (2.3) and (2.8) and that $\mathcal{J}_q \mathbf{U}_0 = [U_0, 0, ..., 0]^T$. For multiple adjacent time elements equation (2.8) can be naturally extended, with obvious notation, as

$$\mathcal{K}_q \mathbf{U}_n = \mathcal{J}_q \mathbf{U}_{n-1} + \mathcal{M}_q F(\mathbf{U}_n). \quad (2.11)$$

[1]The DG solution U can be discontinuous at the nodes. Introducing a left/right nodal values of U we have: $U(T_n)^- = U_0$ and $U(T_n)^+ = U(T_n)$. Then, in general, $U_0 \neq U(T_n)$.

3. Problem setting and discretization

Let $\Omega = (0, X)$ be the spatial domain and $T \in \mathbb{R}^+$ the final time. We consider the following non-linear reaction diffusion equation:

$$\begin{cases} \partial_t u - \partial_{xx} u + \gamma(u^3 - u) = 0, & \text{for} \quad (t, x) \in (0, T) \times \Omega, \\ \partial_x u = 0, & \text{for} \quad (t, x) \in (0, T) \times \partial\Omega, \\ u = u_0, & \text{for} \quad t = 0 \quad \text{and} \quad x \in \Omega, \end{cases} \tag{3.1}$$

where $u := u(t, x)$, $u_0 := u_0(x)$, and $\gamma \geq 0$ controls the intensity of the reaction term. Equation (3.1) is known as the monodomain model or Allen-Cahn equation, and can be used to describe the progressive activation of excitable media. For example, in the context of computational medicine, it is employed to simulate the propagation of the electrical potential in the human heart [38]. The cubic term is a FitzHugh-Nagumo-type reaction, with three zeros $\{-1, 0, 1\}$ corresponding, respectively, to a resting state, a threshold and an activation state. For $\gamma = 0$ equation (3.1) is reduced to the linear heat equation.

Let $N_t, N_x \in \mathbb{N}$ be the number of time and space elements respectively, and define the following uniform partitions in time and space:

$$\begin{aligned} t_i &:= i\Delta t, \qquad i = 0, \dots, N_t, \qquad \Delta t := T/N_t, \\ x_j &:= jh, \qquad j = 0, \dots, N_x, \qquad h := X/N_x. \end{aligned}$$

In space, we approximate (3.1) with linear finite elements, constructing the stiffness and mass matrices $\mathcal{K}_h, \mathcal{M}_h \in \mathbb{R}^{(N_x+1)\times(N_x+1)}$

$$\mathcal{K}_h := \left[\int_\Omega \varphi_i'(x)\varphi_j'(x)\mathrm{d}x\right]_{i,j=0}^{N_x}, \qquad \mathcal{M}_h := \left[\int_\Omega \varphi_i(x)\varphi_j(x)\mathrm{d}x\right]_{i,j=0}^{N_x}, \tag{3.2}$$

using the linear Lagrange basis functions $\{\varphi_i\}_{i=0}^{N_x} \subset H^1(\Omega)$. Referring to Section 2.2, we can consider a space-time finite element approximation of (3.1) in $[t_n, t_{n+1}]$ with a tensor structure:

$$u(x, t) \approx U(x, t) = \sum_{i=1}^{M} \sum_{j=0}^{N_x} u_{i,j}^{n+1} \ell_{n,i}(t)\varphi_j(x) \tag{3.3}$$

and assemble the non-linear space-time system of size $N_t M(N_x+1)$

$$\begin{bmatrix} \mathcal{A}_{q,h} & & & \\ \mathcal{B}_{q,h} & \mathcal{A}_{q,h} & & \\ & \ddots & \ddots & \\ & & \mathcal{B}_{q,h} & \mathcal{A}_{q,h} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{N_t} \end{bmatrix} + \mathcal{I}_t \otimes \mathcal{M}_{q,h} \begin{bmatrix} r(\mathbf{u}_1) \\ r(\mathbf{u}_2) \\ \vdots \\ r(\mathbf{u}_{N_t}) \end{bmatrix} = \begin{bmatrix} -\mathcal{B}_{q,h}\mathbf{u}_0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \tag{3.4}$$

with

$$\mathcal{A}_{q,h} := \mathcal{K}_q \otimes \mathcal{M}_h + \mathcal{M}_q \otimes \mathcal{K}_h, \qquad \mathcal{B}_{q,h} := -\mathcal{J}_q \otimes \mathcal{M}_h, \qquad \mathcal{M}_{q,h} := \mathcal{M}_q \otimes \mathcal{M}_h, \tag{3.5}$$

$\mathcal{I}_t$ being the identity of size N_t. For $n = 1, ..., N_t$, the solution vector for the nth time element $\mathbf{u}_n$ has size $M(N_x+1)$, with values grouped together according to the ordering in (3.5), i.e.

$$\mathbf{u}_n := \left[u^n_{1,0}, u^n_{1,1}, ..., u^n_{1,N_x}, u^n_{2,0}, ..., u^n_{2,N_x}, ..., ..., u^n_{M,0}, ..., u^n_{M,N_x}\right], \tag{3.6}$$

and the point-wise reaction defined by

$$[r(\mathbf{u}_n)]_k := \gamma\left([\mathbf{u}_n]^3_k - [\mathbf{u}_n]_k\right) \quad \text{for} \quad k = 0, ..., N_x M. \tag{3.7}$$

The initial condition is imposed through

$$\mathbf{u}_0 := [0, ..., 0, u_0(x_0), u_0(x_1), .., u_0(x_{N_x})],$$

having $(N_x+1)(M-1)$ zeros. Let use mention that, in the space-time context, increasing M leads to denser blocks. For a detailed description of the weak formulation of (3.1) (for $\gamma = 0$), the assembly and spectral analysis of system (3.4), in a more general finite element framework, we refer to [25]. With respect to standard methods, the storage of system (3.4) can be expensive in terms of memory; nevertheless, the space-time formulation can be convenient, in terms of scaling and run-time, if (3.4) is distributed among multiple processors and solved in parallel. Let us remark that we assemble (3.4) just in the space-time multigrid case; when using PFASST the assembly of the spatial operators in (3.2) is sufficient. For technical limitations related to the current PFASST implementation, we replace the mass matrix $\mathcal{M}_h$ with its lumped version for both discretizations.

4. Solution methods

Here we introduce the two solution strategies that will be the object of the comparison.

4.1. PFASST

The parallel full approximation scheme in space and time was introduced by Emmett and Minion in 2012 [2]. As the name suggests, PFASST can be described in the context of a multigrid in time method based on a FAS correction on coarse levels [11]. An alternative perspective on how the PFASST method is organized is to view it as a way to perform SDC iterations for the collocation Eq. (2.3) on multiple time steps simultaneously. For parallel efficiency, the SDC iterations are done on a hierarchy of levels as in the multilevel SDC method [39] with communication of new initial conditions passed forward in time between processors after each SDC iteration on each level. Since the communication is only serial on the coarsest level, the SDC iterations on the finest level are done concurrently, resulting in a potential parallel speedup if the total number of PFASST iterations needed to converge on all the time steps remains relatively small. One advantage of viewing PFASST from this SDC perspective is that variants of the original SDC method such as semi-implicit SDC (SISDC) [40], can be easily used in the PFASST context. SISDC methods (also known as implicit-explicit or IMEX) are appropriate for differential equations for which the right hand side of (2.1) can be split into stiff and non-stiff parts. These methods are often employed in situations where the non stiff component is nonlinear and the stiff term is linear, so that only a linear implicit equation needs to be solved in each time step. In Section 5.4, an IMEX treatment is used to treat the nonlinear reaction terms explicitly and the linear diffusion terms implicitly.

4.2. Space-time multigrid

Specialized parallel solvers have been recently developed for large linear systems arising from space-time discretizations. We mention in particular the parallel STMG proposed by [6], the parallel preconditioners for space-time isogeometric analysis proposed by [41] and [34] as well as the block preconditioned GMRES by [42]. When dealing with a space-time discretization,

where time is somehow considered as an additional spatial dimension, it is natural to extend the same paradigm for the solving process and consider space-time multigrid type algorithms.

Multigrid solvers are optimal preconditioners for elliptic problems, and they have proven to be efficient, with some precautions, also for space-time discretizations of parabolic problems, such as the heat equation. The heat equation is first order in time and its discretization introduces non symmetric lower bi-diagonal blocks in the space-time system (3.4). When dealing with anisotropic problems standard multigrid convergence rates deteriorate; see, e.g., [43]. Traditionally there are various ways to address this problem such as accounting for anisotropy in the particular choice of line smoothers and/or adopting a semi-coarsening strategy. In [4, 10, 7, 44], for example, the authors explain how the STMG convergence depends critically on the ratio $\mu := \Delta t/h^2$, unless semi-coarsening strategies are adopted. In particular, for $\mu \ll 1$ (resp. $\mu \gg 1$) coarsening only in time (resp. space) is an effective strategy.

Let us consider a hierarchy of L space-time grids denoted with $l = 1, ..., L$ and $l = L$ corresponding to the coarsest one. We construct the space-time restriction operator $\mathcal{R}_l^{l+1}$ from level l to level $l+1$ as

$$\mathcal{R}_l^{l+1} = \mathcal{T}_l^{l+1} \otimes \mathcal{P}_l^{l+1} \otimes \mathcal{S}_l^{l+1}, \quad \text{for} \quad l = 1, ..., L-1, \tag{4.1}$$

where $\mathcal{T}_l^{l+1}$ and $\mathcal{S}_l^{l+1}$ are restriction operators in time and space respectively and $\mathcal{P}_l^{l+1}$ is responsible for $M-$coarsening in time, i.e. varying the number of Gauss-Radau nodes M along the multilevel hierarchy. Interpolation operators are obtained considering the transpose of $\mathcal{R}_l^{l+1}$. Definitions of these operators will be provided in the next section. Let us mention that any restriction operator in (4.1) can be replaced by a suitable identity matrix, resulting in various semi-coarsening strategies. If $M-$coarsening is present, i.e. if $\mathcal{P}_l^{l+1}$ is not an identity, the resulting multigrid is also known as $p-$multigrid, or, more precisely, as a $p-$multigrid in time (as PFASST).

For smoothing, we employ GMRES preconditioned with an incomplete LU factorization (ILU(0)-PGMRES). When multiple parallel cores are used, GMRES is preconditioned using a block-Jacobi preconditioner, with blocks of size $(N_x+1)N_tM/\text{Cores}$; for each diagonal block ILU(0) is then employed. For specialized preconditioners and corresponding tensor solvers applied to

system (3.4) see, e.g., [45, 46, 34]. On the coarsest level, an LU factorization is used and coarse problems are assembled through Galerkin assembly. If $\gamma \neq 0$, equation (3.4) is non-linear and the STMG algorithm is encapsulated in a Newton iteration.

5. Experiments

5.1. Implementation

For the numeric examples in this section, as well as throughout this paper, we used the PETSc framework [47, 48] and the C++ embedded domain specific language Utopia[2] [49] for the parallel linear algebra and the linear and non-linear solvers. For PFASST we use the modern Fortran library LibPFASST[3] that was extended to use the same PETSc data structures and linear solvers as the STMG code to make the comparison as fair as possible[4]. The two discretizations produce, up to machine precision, the same solution in T.

Parallel numerical experiments have been performed on the multi-core partition of the supercomputer Piz Daint of the Swiss national supercomputing centre (CSCS)[5].

5.2. Solvers specifics and notation

Next we introduce some of the notation that we are going to use in the following numerical experiments:

- $\boxed{\text{SMG}_{\nu}^{L}}$

  Multigrid with L levels and spatial coarsening, with $\mathcal{S}_l^{l+1}$ in (4.1) being standard linear restriction, using the stencil $[1\,2\,1]/4$ and the operator $\mathcal{S}_l^{l+1}$ having size $(1 + N_x/2^{l+1}) \times (1 + N_x/2^l)$. As time coarsening is not employed, time transfers in (4.1) are replaced by identities, i.e.

[2] `https://bitbucket.org/zulianp/utopia`

[3] `https://pfasst.lbl.gov/codes`

[4] For both implementations we used the PETSc type `MATMPIAIJ` to store distributed sparse matrices and the `KSP` type for Krylov solvers.

[5] `https://www.cscs.ch/computers/piz-daint`

$\mathcal{T}_l^{l+1} = \mathcal{I}_t \in \mathbb{R}^{N_t \times N_t}$ and $\mathcal{P}_l^{l+1} = \mathcal{I}_M \in \mathbb{R}^{M \times M}$ for all $l = 1, ..., L-1$. We use V-cycling, with ν smoothing iterations of ILU(0)-PGMRES. We refer to [50] for a convergence analysis in this setting.

- $\boxed{\text{STMG}_\nu^L}$

  Space-time multigrid with L levels and $\mathcal{T}_l^{l+1}$ and $\mathcal{S}_l^{l+1}$ in (4.1) being standard linear restriction operators, both using the same stencil introduced for SMG_ν^L; the operator $\mathcal{T}_l^{l+1}$ has size $N_t/2^{l+1} \times N_t/2^l$. We set $\mathcal{P}_l^{l+1} = \mathcal{I}_M$ for all $l = 1, ..., L-1$, i.e. M is constant along the multilevel hierarchy. We use V-cycling, with ν smoothing iterations of ILU(0)-PGMRES.

- $\boxed{\text{SMMG}_\nu^L}$

  As SMG_ν^L, but using $M-$coarsening in time, also known as $p-$multigrid in time. To perform $M-$coarsening we consider a non-nested hierarchy of Gauss-Radau grids on a reference time element, where the number of time nodes M is reduced progressively on the hierarchy until $M = 1$ is reached, i.e. having $\max\{M - l + 1, 1\}$ nodes on level l. In (4.1) $\mathcal{P}_l^{l+1}$ is obtained through linear interpolation between successive Gauss-Radau nodes.

- $\boxed{\text{PFASST}_\nu^L}$

  PFASST solver, as described in Section 4.1, with L levels and ν sweeps per level. We use ILU(0)-PGMRES as a spatial solution method and standard linear restriction to create coarse spatial problems. Regarding temporal coarsening, for performance reasons, we use $M = 1$ on all coarse levels.

In the numerical results the run-times are expressed in seconds; the assembly of discrete problems and transfer operators are not included in the run-times. The number of iterations to convergence, if present, is reported in square brackets. Convergence is reached when the relative or the absolute preconditioned residual is less then a tolerance of 10^{-9}. The acronym "n.c." stands for "not converged", denoting an increasing residual or if 1000 iterations are

exceeded. The tests are restricted to temporal parallelism, i.e. #Cores $\leq N_t$, with solvers parameters (i.e. L and ν) which minimize run-time for both approaches. Linear and non-linear iterative solvers are initialized with the zero vector in the space-time case. The spatial diffusion solver in PFASST (PGMRES), are initialized with the best available guess, i.e. the solution at the previous iteration.

5.3. Linear example: the heat equation

In this section we consider the heat equation, i.e. in the following experiments we set $\gamma = 0$ in (3.1), and the initial condition

$$u_0(x) = \cos(\pi x) + 2\cos(3\pi x) + 3\cos(4\pi x) \quad \text{for} \quad x \in [0, X], \tag{5.1}$$

with the corresponding analytical solution

$$u(x,t) = \cos(\pi x)e^{-\pi^2 t} + 2\cos(3\pi x)e^{-9\pi^2 t} + 3\cos(4\pi x)e^{-16\pi^2 t}. \tag{5.2}$$

We also consider the analytical solution $\tilde{u}$, obtained after spatial discretization, to focus on the error introduced just by temporal discretization:

$$\tilde{u}(x,t) = \cos(\pi x)e^{\rho_1 t} + 2\cos(3\pi x)e^{\rho_2 t} + 3\cos(4\pi x)e^{\rho_3 t}, \tag{5.3}$$

with

$$\rho_1 = (2\cos(\pi h) - 2)/h^2, \quad \rho_2 = (2\cos(3\pi h) - 2)/h^2, \quad \rho_3 = (2\cos(4\pi h) - 2)/h^2.$$

We show, in Figure 5.1, how the error behaves as a function of the temporal discretization parameters, i.e. N_t and M, for problem (3.1) with $T = X = 1$, $\gamma = 0$ and u_0 from equation (5.1), discretized, according to (3.4) with $N_x = 1024$. It is possible to observe, in the left plot in Figure 5.1, that the error compared to the exact solution decreases as the number of time steps increases until the spatial error of roughly 10^{-9} dominates. For this reason the solver tolerance is set to 10^{-9} in the following numerical experiments. The right-hand plot shows that the temporal error decreases with the correct order $2M - 1$ until machine precision is reached.

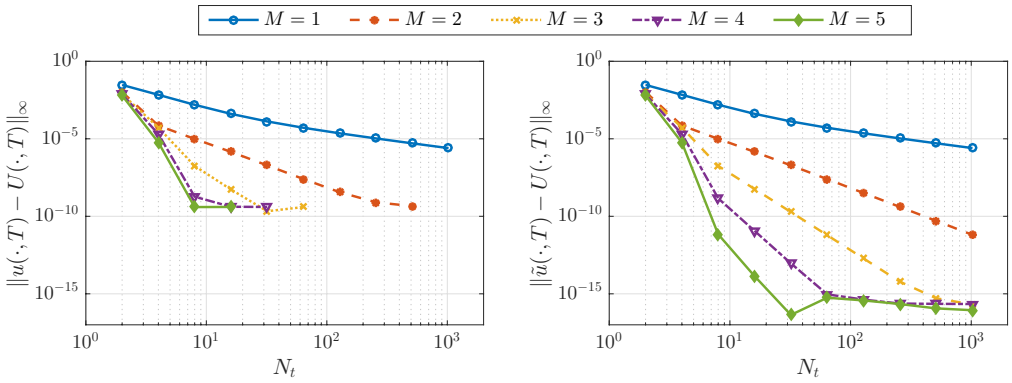

Figure 5.1: Left: error at the end node w.r.t. the analytical solution (5.2). Right: error w.r.t. (5.3), i.e. the error of the discrete ODE. In both cases, the errors decrease, with the expected order $(2M-1)$, until the spatial error dominates.

In Example 5.1-5.2 we report strong and weak scaling results varying the discretization order M. We avoid over-resolving in time by reducing N_t as M increases, according to Figure 5.1. In particular, we set N_t, depending on M, to be the minimum power of two for which $\|u(\cdot,T) - U(\cdot,T)\|_\infty < 10^{-9}$ is satisfied. With this methodology, the solver tolerance and the spatial and temporal accuracies are approximately the same[6].

Example 5.1. (Strong scaling). Let us consider the continuous problem (3.1) with parameters $X = T = 1$ and u_0 from (5.1). We use the discretization parameters $N_x = 1024$, $M = \{1, ..., 5\}$ and N_t varying according to the results of Figure 5.1, to avoid over-resolving in time, except for $M = 1$ where we use $N_t = 1024$, with a corresponding accuracy of approximately 10^{-6}.

We show, in Tables 5.1–5.4, run-times and iterations of the three multilevel approaches described in Section 5.2. The most relevant results are illustrated in Figure 5.2.

[6]For $M = 1$, we use $N_t = 1024$ with a corresponding temporal accuracy of $\sim 10^{-6}$, since $N_t \approx 10^7$ would be required to reach a temporal accuracy of 10^{-9}.

	SMG_3^7				
Cores	$M = 1$, $N_t = 1024$	$M = 2$, $N_t = 256$	$M = 3$, $N_t = 32$	$M = 4$, $N_t = 16$	$M = 5$, $N_t = 8$
1	1.59 [2]	2.03 [4]	0.49 [4]	0.41 [5]	0.32 [5]
2	1.24 [2]	1.27 [4]	0.29 [4]	0.30 [5]	0.22 [5]
4	0.62 [2]	0.80 [4]	0.18 [4]	0.17 [5]	0.13 [5]
8	0.38 [2]	0.37 [4]	0.10 [4]	0.11 [5]	0.08 [5]
16	0.29 [3]	0.23 [4]	0.08 [4]	0.07 [5]	
32	0.18 [3]	0.15 [4]	0.07 [4]		
64	0.12 [3]	0.14 [4]			
128	0.11 [3]	0.12 [4]			
256	0.11 [3]	0.12 [4]			
512	0.14 [3]				
1024	0.17 [3]				

Table 5.1: Seven level space-time multigrid run-times and iterations to convergence with no temporal coarsening.

	STMG_3^5				
Cores	$M = 1$, $N_t = 1024$	$M = 2$, $N_t = 256$	$M = 3$, $N_t = 32$	$M = 4$, $N_t = 16$	$M = 5$, $N_t = 8$
1	2.66 [6]	6.57 [18]	2.12 [30]	1.75 [30]	1.23 [30]
2	1.53 [6]	3.71 [18]	1.21 [29]	0.98 [30]	0.71 [31]
4	0.80 [6]	1.82 [18]	0.64 [32]	0.64 [33]	0.43 [32]
8	0.50 [6]	1.05 [18]	0.39 [34]	0.52 [37]	0.27 [34]
16	0.35 [6]	0.67 [18]	0.29 [37]	0.26 [37]	
32	0.22 [6]	0.53 [18]	0.22 [38]		
64	0.17 [6]	0.45 [17]			
128	0.17 [6]	0.37 [17]			
256	0.17 [6]	0.37 [17]			
512	0.18 [7]				
1024	0.29 [9]				

Table 5.2: Five level space-time multigrid run-times and iterations to convergence, with full space-time coarsening.

	SMMG_3^7			
Cores	$M = 2$, $N_t = 256$	$M = 3$, $N_t = 32$	$M = 4$, $N_t = 16$	$M = 5$, $N_t = 8$
1	3.01 [12]	0.99 [17]	0.79 [15]	0.75 [18]
2	1.77 [12]	0.62 [17]	0.44 [15]	0.50 [18]
4	0.86 [12]	0.28 [17]	0.23 [15]	0.28 [18]
8	0.47 [12]	0.17 [17]	0.15 [15]	0.15 [18]
16	0.24 [12]	0.12 [17]	0.11 [15]	
32	0.15 [12]	0.09 [17]		
64	0.12 [12]			
128	0.11 [12]			
256	0.11 [12]			

Table 5.3: Seven level space-time multigrid run-times and iterations to convergence, with $M-$coarsening in time. The column for $M = 1$ is not present as it would be equivalent to the one of Table 5.1.

	PFASST_1^3				
Cores	$M = 1$, $N_t = 1024$	$M = 2$, $N_t = 256$	$M = 3$, $N_t = 32$	$M = 4$, $N_t = 16$	$M = 5$, $N_t = 8$
1	1.49 [2]	0.74 [3]	0.23 [11]	0.17 [12]	0.12 [12]
2	1.26 [3]	0.71 [11]	0.17 [14]	0.12 [14]	0.09 [15]
4	0.85 [4]	0.50 [13]	0.11 [16]	0.08 [17]	0.06 [18]
8	0.55 [6]	0.40 [17]	0.09 [20]	0.07 [21]	0.05 [22]
16	0.37 [7]	0.31 [24]	0.08 [28]	0.06 [29]	
32	0.23 [7]	0.21 [29]	0.06 [35]		
64	0.18 [7]	0.15 [30]			
128	0.15 [8]	0.11 [30]			
256	0.15 [7]	0.11 [30]			
512	0.16 [8]				
1024	0.20 [7]				

Table 5.4: Three level PFASST run-times and iterations to convergence.

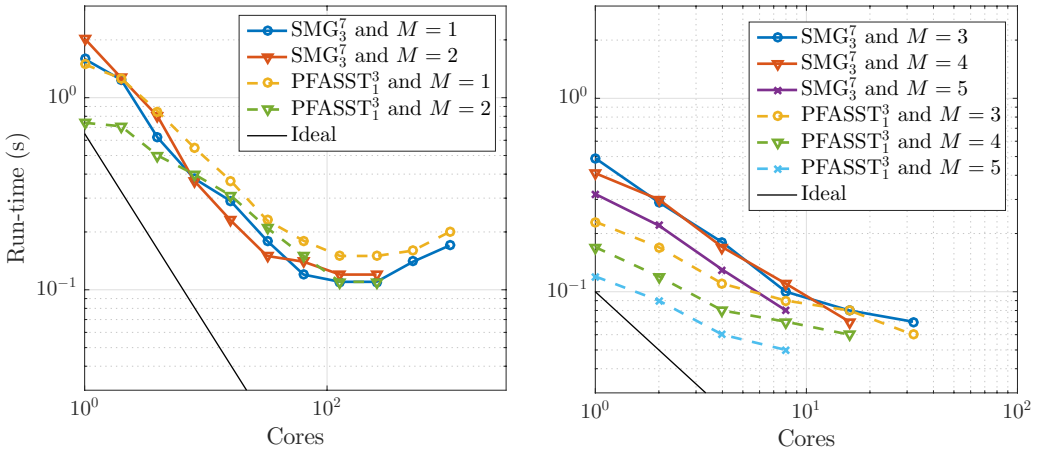

Figure 5.2: Strong scaling timing results of SMG (solid lines) and PFASST (dashed lines) from Example 5.1; run-times of STMG and SMMG are not included since they are not competitive. We report run-times for $M \in \{1, 2\}$ in the left plot and for $M \in \{3, 4, 5\}$ in the right one.

The results of Example 5.1 suggest several observations. First, for the space-time multigrid methods, using spatial coarsening only (SMG) obtains the fastest space-time multigrid convergence and run-times. We can explain this behavior using the discretization parameter

$$\mu = \frac{\Delta t}{\Delta x^2} = \frac{N_x^2}{N_t}.$$

In all cases considered $\mu \gg 1$; from space-time multigrid literature (e.g. [7],[51]) we can expect time coarsening to be not effective in this regime, as we observe by the larger iteration counts in Table 5.2. The case of $\mu \leq 1$ would not be meaningful in this setting, as it would result in a unnecessary over-resolved time discretization, with $N_t \geq N_x^2 > 10^6$.

Let us remark that differences in the convergence behavior for various coarsening strategies (cf. Tables 5.1–5.4) are not due to different sizes of coarse problems but are related to their conditioning. In fact, avoiding time coarsening results in better conditioned coarse problems. For example, the condition number of the coarsest matrices for SMG$_3^7$ and STMG$_3^4$ (that have the same size) is, respectively, $1.7 \cdot 10^3$ and $1.5 \cdot 10^4$ and time coarsening results in a slower convergence (six iterations) w.r.t. the SMG setting (two

iterations). The results collected in Table 5.3 suggest that $M-$coarsening is the most convenient way to coarsen in time in the space-time multigrid case, since run-times are close to the case of spatial coarsening only. Note also that the $M-$coarsening in time for SMMG and PFASST are the same. Comparing SMG with PFASST, cf. Figure 5.2, we observe an overall similar scaling and run-times, especially for $M \in \{1, 2\}$. As expected from the space-time paradigm, SMG is characterized by a slower sequential run-time compensated by a better scaling. As as results, both algorithm achieve a similar stagnation run-time. For higher values of M, PFASST is somewhat faster than SMG, presumably due to the reduced cost of the coarsest level. Finally, it is important to note that for the same accuracy, the higher-order methods have lower run-times than the lower-order methods, often with fewer processors. Hence, although the scaling in terms of iterations is better for the lower-order methods, they are more expensive in practice.

Example 5.2. (Weak time scaling in N_t) Let us consider the continuous problem (3.1) with parameters $X = T = 1$ and u_0 from (5.1). We use the discretization parameters $N_x = 1024$, $M = \{1, ..., 5\}$ and $N_t = C_M \cdot$ Cores. The parameter C_M depends on M and is chosen according to the parallel saturation from Example 5.1; for example, for $M = 1$ and $N_t = 1024$, according to Tables 5.1–5.4, we have maximum speedup with 256 cores and therefore $C_1 = 4$. Similarly $C_2 = 2$ and $C_M = 1$ for $M \geq 3$. We point out that the accuracy of the solution as a function of N_t is varying in the tables according to Figure 5.1: doubling N_t corresponds to a higher accuracy as M increases. We report, in Tables 5.5–5.6 and Figure 5.3 run-times and iterations of the multilevel approaches described in Section 5.2.

$M = 1$

Cores	N_t	L	time [its.]	R
256	1024	7	0.11 [3]	1
512	2048	8	0.15 [4]	1.4
1024	4096	8	0.33 [5]	3.0
2048	8192	8	0.72 [6]	6.5

$M = 2$

Cores	N_t	L	time [its.]	R
128	256	7	0.11 [4]	1
256	512	7	0.17 [4]	1.5
512	1024	8	0.18 [4]	1.6
1024	2048	8	0.43 [5]	3.9

$M = 3$

Cores	N_t	L	time [its.]	R
32	32	7	0.07 [4]	1
64	64	7	0.08 [5]	1.1
128	128	8	0.10 [5]	1.4
256	256	8	0.14 [5]	2.0

$M = 4$

Cores	N_t	L	time [its.]	R
16	16	7	0.10 [5]	1
32	32	7	0.10 [5]	1.0
64	64	7	0.12 [5]	1.2
128	128	8	0.16 [6]	1.6

$M = 5$

Cores	N_t	L	time [its.]	R
8	8	7	0.12 [5]	1
16	16	7	0.13 [6]	1.1
32	32	7	0.14 [7]	1.2
64	64	7	0.18 [7]	1.5

Table 5.5: Weak scaling in time of space-time multigrid SMG^3_L, with no temporal coarsening. The ratio R is computed dividing the current run-time by the base one (in the first line for each table) and $R = 1$ denotes an ideal weak scaling. Since no temporal coarsening is present, the weak scaling is poor for $M \in \{1, 2\}$.

$M = 1$

Cores	N_t	time [its.]	R
256	1024	0.16 [7]	1.0
512	2048	0.26 [8]	1.6
1024	4096	0.49 [10]	3.1
2048	8192	0.96 [10]	6.0

$M = 2$

Cores	N_t	time [its.]	R
128	256	0.09 [30]	1.0
256	512	0.11 [30]	1.2
512	1024	0.16 [29]	1.8
1024	2048	0.25 [30]	2.8

$M = 3$

Cores	N_t	time [its.]	R
32	32	0.06 [35]	1.0
64	64	0.11 [38]	1.8
128	128	0.12 [38]	2.0
256	256	0.13 [36]	2.2

$M = 4$

Cores	N_t	time [its.]	R
16	16	0.07 [29]	1.0
32	32	0.10 [38]	1.4
64	64	0.12 [42]	1.4
128	128	0.12 [42]	1.7

$M = 5$

Cores	N_t	time [its.]	R
8	8	0.05 [22]	1.0
16	16	0.08 [30]	1.6
32	32	0.10 [39]	2.0
64	64	0.11 [42]	2.2

Table 5.6: Weak scaling in time of PFASST$^3_1$. The ratio R is computed dividing the current run-time by the base one (in the first line for each table) and $R = 1$ denotes an ideal weak scaling. We can notice that the weak scaling for $M = 1$ is poor, since no temporal coarsening is present in this case.

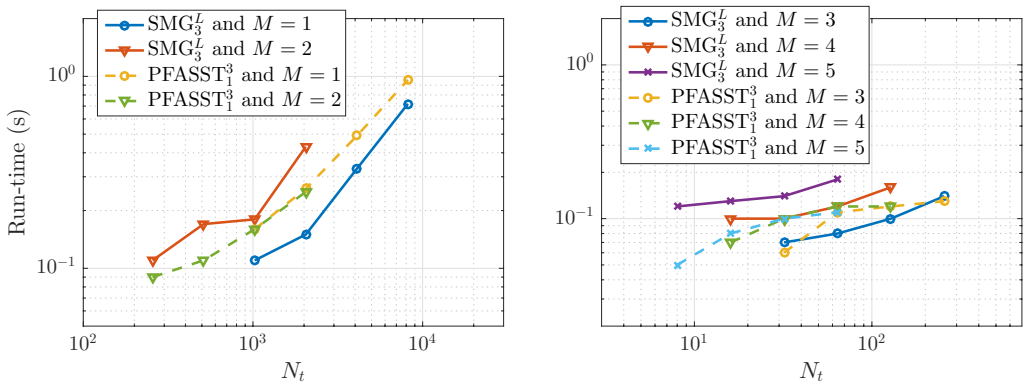

Figure 5.3: Weak scaling timing results of SMG (solid lines) and PFASST (dashed lines) from Tables 5.5–5.6; run-times of STMG and SMMG are not included since they are not competitive. We report run-times for $M \in \{1, 2\}$ in the left plot and for $M \in \{3, 4, 5\}$ in the right one.

It is clear from Figure 5.3 that higher-order methods display better weak scaling than the lower-order ones. This is partly due to the absence of time-coarsening w.r.t. N_t for both methods, that becomes relevant as N_t grows. It should be noted that for $M > 1$, the error w.r.t. the analytical solution is already saturated in the first rows of Tables 5.5–5.6 (according to Figure 5.1) and increasing N_t does not produce a more accurate solution. In this scaling regime, there is no consistent winner between SMG and PFASST for all M and N_t.

5.4. Non-linear example: the monodomain equation.

In this section we consider the full reaction-diffusion model, i.e. $\gamma > 0$ in (3.1). Note that for the space-time discretization in (3.4), a non-linear solver is required due to the cubic reaction term. For the implementation in PFASST an IMEX or semi-implicit method is employed for each correction substep [52] treating the nonlinear reaction terms explicitly. Hence the cost per iteration of the PFASST method is essentially the same as for the linear case since the implicit part is much more expensive. One could instead use a fully implicit substepping method such as backward-Euler in PFASST, which would result in a more expensive method, but one with a larger stability region (see for example [53]). Alternatively, a multi-implicit approach could

also be chosen as in [54]. A careful comparison of each approach is beyond the scope of the current study, but it is important to emphasize that the flexibility in the PFASST methods is one important difference between PFASST and STMG.

To model a traveling wave in an excitable media we consider reaction dominated examples. In this case we set a narrow initial stimulus in the centre of the domain and we chose T such that the final solution is stationary, i.e. for all x we have $u(T, x) \simeq 1$ and $\partial_t u(T, x) \simeq 0$.

Example 5.3. (Strong scaling) Let us consider the continuous problem (3.1) with model parameters $X = 10, T = 2, \gamma = 5$ and the initial condition

$$u_0 = 2 \exp\left(\frac{x - X}{0.1}\right)^2.$$

We use the discretization parameters $N_x = N_t = 1024$ and $M = \{1, ..., 5\}$. We show, in Table 5.7, run-times and Newton iterations of the space-time strategy, using SMG_3^7 as linear solver and, in Table 5.8, the PFASST data.

	SMG_3^7				
Cores	$M = 1$, $N_t = 1024$	$M = 2$, $N_t = 256$	$M = 3$, $N_t = 32$	$M = 4$, $N_t = 16$	$M = 5$, $N_t = 8$
1	46.4 [16]	31.1 [16]	6.47 [16]	5.18 [16]	301 [58]
2	29.7 [16]	19.5 [16]	4.50 [16]	3.75 [16]	123 [45]
4	16.6 [16]	11.0 [16]	2.61 [16]	2.17 [16]	125 [47]
8	10.6 [16]	7.45 [16]	1.81 [16]	1.60 [16]	34.4 [18]
16	8.51 [16]	5.81 [16]	1.36 [16].	1.16 [16]	
32	7.22 [16]	4.92 [16]	1.07 [16]		
64	6.11 [16]	4.54 [16]			
128	5.10 [16]	3.90 [16]			
256	5.50 [16]	5.48 [16]			
512	5.64 [16]				
1024	6.20 [16]				

Table 5.7: Run-time of a seven level space-time multigrid, with no temporal coarsening and corresponding Newton iterations.

	PFASST$_1^3$				
Cores	$M = 1$, $N_t = 1024$	$M = 2$, $N_t = 256$	$M = 3$, $N_t = 32$	$M = 4$, $N_t = 16$	$M = 5$, $N_t = 8$
1	1.51 [2]	0.54 [5]	0.23 [93]	n.c.	n.c.
2	1.16 [3]	0.42 [6]	0.20 [94]	n.c.	n.c.
4	0.67 [4]	0.25 [8]	n.c.	n.c.	n.c.
8	0.40 [5]	0.16 [8]	n.c.	n.c.	n.c.
16	0.27 [6]	0.11 [8]	n.c.	n.c.	
32	0.18 [6]	0.07 [8]	n.c.		
64	0.14 [6]	0.07 [8]			
128	0.13 [6]	0.07 [8]			
256	0.15 [6]	0.08 [8]			
512	0.16 [6]				
1024	0.20 [6]				

Table 5.8: Three level PFASST run-times and iterations, n.c. abbreviating "not converged".

Note that in the cases where PFASST converges, the run-time is significantly smaller than the corresponding SMG times. As mentioned above, this is due to the fact that the PFASST implementation is using a semi-implicit or IMEX time stepping method, while SMG method is fully implicit requiring Newton iterations. The failure of PFASST to converge for large time steps is also due to the IMEX stepping, which has a time step restriction due to the explicit treatment of the reaction term (see, e.g. [40]). The reaction term could also be handled implicitly in PFASST using a multi-implicit approach [55] as was done in [54] to increase the stability, but we defer this sort of comparison to future work.

Let us remark that using an IMEX approach is a standard choice for reaction-diffusion problems, but is not favorable in the space-time context. For space-time, both a fully explicit and an IMEX approach would lead to a non-linear system to be solved. An explicit/IMEX space-time discretization of the reaction term leads to approximately the same number of Newton iterations and run-times obtained for the implicit case.

6. Conclusions

In this paper we discussed the parallel performance of multilevel space-time solution strategies and of the algorithm PFASST for a (reaction) diffusion problem. From an implementation prospective, space-time multigrid approaches are convenient since the time parallelization boils down to the parallel solution of a system of equations (in (3.4)), for example using fast and parallel preconditioned Krylov methods as PGMRES. A tensor structure between space and time grids allows for a flexible choice of coarsening strategies, since transfer operators in (4.1) can be set independently. On the other hand, the assembly of system (3.4) comes at a cost, in terms of time[7] and, especially, memory footprint[8]. Such cost can be be reduced significantly when (3.4) is distributed among many processors and if highly parallel assembly routines are used, as parallel Kronecker products in (3.5). Let us mention that matrix-free implementations of (3.4) are also possible, avoiding assembling costs and reducing storage, but have more limitations in terms of solution strategy (e.g. ILU preconditioners are typically not available in a matrix-free context).

In Examples 5.1–5.2 we investigated the scalability of different parallel iterative strategies for a diffusion problem. We obtained similar performance from PFASST and the parallel space-time multigrid with no temporal coarsening (SMG). The use of high order methods in time, reducing the number of time steps N_t accordingly, is convenient for both approaches, in terms of overall performance and especially for the weak scaling in time.

As expected from the literature, full space-time coarsening or time coarsening are not effective in the settings we considered ($\mu \gg 1$). In the space-time multigrid framework $M-$coarsening in time can be advantageous w.r.t. coarsening in the number of time steps N_t, in terms of stability, but employ-

[7]With reference to Example 5.1, the serial assembly of the space-time system (3.4) takes 0.28 seconds for $M = 1$ and 0.04 for $M = 5$, corresponding to $\sim 15\%$ of the STMG solving time. Similarly, the assembly of space-time restriction operators (4.1) takes 0.26 seconds for $M = 1$ and 0.01 for $M = 5$. Assembly routines are fully scalable.

[8]The number of non-zero elements in the coefficient matrix of the space-time system (3.4), is given by $O(N_x N_t M^2)$. Memory footprint is quadratic w.r.t. M, but, for a fixed accuracy, as in the presented experiment, $N_t M^2$ is decreasing as M increases

ing just coarsening in space remains the best option for the discretizations considered.

In Example 5.3 we considered a non-linear reaction-diffusion problem. For such a problem the space-time approach is limited to a fully implicit treatment of the non-linearity and the corresponding use of a non-linear solver, such as Newton's method. In particular, we observe that the number of Newton iterations to convergence is not robust in terms of problem parameters and initial guess. On the other hand, in this respect PFASST is more flexible since it allows one to treat the non-linearity explicitly, through an IMEX approach. Such a strategy, even if less stable (especially for large Δt and high order M), can reduce dramatically the time-to-solution.

Acknowledgements

The authors acknowledge the Deutsche Forschungsgemeinschaft (DFG) as part of the "ExaSolvers" Project in the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) and the Swiss National Science Foundation (SNSF) under the lead agency grant agreement SNSF-162199.

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

The work of M. Minion was supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under contract number DE-AC02005CH11231. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC0205CH11231.

Declaration of interest: none.

References

[1] M. J. Gander, 50 years of time parallel time integration, in: Multiple Shooting and Time Domain Decomposition, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-319-23321-5_3

[2] M. Emmett, M. L. Minion, Toward an efficient parallel in time method for partial differential equations, Communications in Applied Mathematics and Computational Science 7 (2012) 105–132.
URL http://dx.doi.org/10.2140/camcos.2012.7.105

[3] W. Hackbusch, Parabolic multi-grid methods, Computing Methods in Applied Sciences and Engineering, VI (1984) 189–197.
URL http://dl.acm.org/citation.cfm?id=4673.4714

[4] G. Horton, S. Vandewalle, A Space-Time Multigrid Method for Parabolic Partial Differential Equations, SIAM Journal on Scientific Computing 16 (4) (1995) 848–864.
URL http://dx.doi.org/10.1137/0916050

[5] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, J. B. Schroder, Parallel time integration with multigrid, SIAM Journal on Scientific Computing 36 (2014) C635–C661.
URL http://dx.doi.org/10.1137/130944230

[6] M. J. Gander, M. Neumüller, Analysis of a new space-time parallel multigrid algorithm for parabolic problems, SIAM Journal on Scientific Computing 38 (4) (2016) A2173–A2208.

[7] S. R. Franco, F. J. Gaspar, M. A. V. Pinto, C. Rodrigo, Multigrid method based on a space-time approach with standard coarsening for parabolic problems, Applied Mathematics and Computation 317 (2018) 25–34.

[8] P. Benedusi, D. Hupp, P. Arbenz, R. Krause, A parallel multigrid solver for time-periodic incompressible Navier-Stokes equations in 3D, in: Numerical Mathematics and Advanced Applications ENUMATH 2015, Springer, 2016, pp. 265–273.

[9] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, J. B. Schroder, S. Vandewalle, Multigrid methods with space–time concurrency, Computing and Visualization in Science 18 (4-5) (2017) 123–143.

[10] M. J. Gander, F. Kwok, H. Zhang, Multigrid interpretations of the parareal algorithm leading to an overlapping variant and mgrit, Computing and Visualization in Science 19 (3-4) (2018) 59–74.

[11] M. Bolten, D. Moser, R. Speck, A multigrid perspective on the parallel full approximation scheme in space and time, Numerical Linear Algebra with Applications 24 (6) (2017) e2110.

[12] P. Lasaint, P. Raviart, On a finite element method for solving the neutron transport equation, in: Mathematical Aspects of Finite Elements in Partial Differential Equations, Proceedings of a Symposium Conducted by the Mathematics Research Center, the University of Wisconsin-Madison, Madison, WI, USA, 1974, pp. 1–3.

[13] J. Huang, J. Jia, M. Minion, Accelerating the convergence of spectral deferred correction methods, J. Comput. Phys. 214 (2) (2006) 633–656.

[14] A. Dutt, L. Greengard, V. Rokhlin, Spectral deferred correction methods for ordinary differential equations, BIT Numerical Mathematics 40 (2) (2000) 241–266.

[15] M. Weiser, Faster sdc convergence on non-equidistant grids by dirk sweeps, BIT Numerical Mathematics 55 (4) (2015) 1219–1241.

[16] E. Hairer, G. Wanner, Solving ordinary differential equations II : stiff and differential-algebraic problems, Springer Berlin Heidelberg, 1991.

[17] K. Eriksson, C. Johnson, Adaptive finite element methods for parabolic problems i: A linear model problem, SIAM Journal on Numerical Analysis 28 (1) (1991) 43–77.

[18] M. Schmich, B. Vexler, Adaptivity with dynamic meshes for space-time finite element discretizations of parabolic equations, SIAM Journal on Scientific Computing 30 (1) (2008) 369–393. doi:10.1137/060670468.

[19] P. Jamet, Galerkin-type approximations which are discontinuous in time for parabolic equations in a variable domain, SIAM Journal on Numerical Analysis 15 (5) (1978) 912–928.

[20] C. Klaij, J. van der Vegt, H. van der Ven, Space-time discontinuous Galerkin method for the compressible Navier-Stokes equations, Journal of Computational Physics 217 (2) (2006) 589 – 611.

[21] X. Li, N.-E. Wiberg, Implementation and adaptivity of a space-time finite element method for structural dynamics, Computer Methods in Applied Mechanics and Engineering 156 (1-4) (1998) 211–229.

[22] J. Sudirham, J. van der Vegt, R. van Damme, Space-time discontinuous Galerkin method for advection-diffusion problems on time-dependent domains, Applied Numerical Mathematics 56 (12) (2006) 1491 – 1518.

[23] M. Feistauer, V. Kučera, K. Najzar, J. Prokopová, Analysis of space–time discontinuous Galerkin method for nonlinear convection–diffusion problems, Numerische Mathematik 117 (2) (2011) 251–288.

[24] M. Besier, R. Rannacher, Goal-oriented space–time adaptivity in the finite element Galerkin method for the computation of nonstationary incompressible flow, International Journal for Numerical Methods in Fluids 70 (9) (2012) 1139–1166. doi:10.1002/fld.2735.

[25] P. Benedusi, C. Garoni, R. Krause, X. Li, S. Serra-Capizzano, Space-time FE-DG discretization of the anisotropic diffusion equation in any dimension: The spectral symbol, SIAM Journal on Matrix Analysis and Applications 39 (3) (2018) 1383–1420.

[26] M. Delfour, W. Hager, F. Trochu, Discontinuous Galerkin methods for ordinary differential equations, Mathematics of Computation 36 (154) (1981) 455–473.

[27] K. Eriksson, C. Johnson, V. Thomée, Time discretization of parabolic problems by the discontinuous Galerkin method, ESAIM: Mathematical Modelling and Numerical Analysis 19 (4) (1985) 611–643.

[28] F. Schieweck, A-stable discontinuous Galerkin–petrov time discretization of higher order, Journal of Numerical Mathematics 18 (1) (2010) 25–57.

[29] S. Zhao, G.-W. Wei, A unified discontinuous Galerkin framework for time integration, Mathematical methods in the applied sciences 37 (7) (2014) 1042–1071.

[30] V. Thomée, Galerkin finite element methods for parabolic problems, Vol. 1054, Springer, 1984.

[31] I. Smears, Robust and efficient preconditioners for the discontinuous Galerkin time-stepping method, IMA Journal of Numerical Analysis 37 (4) (2016) 1961–1985.

[32] T. Richter, A. Springer, B. Vexler, Efficient numerical realization of discontinuous Galerkin methods for temporal discretization of parabolic problems, Numerische Mathematik 124 (1) (2013) 151–182.

[33] S. Hussain, F. Schieweck, S. Turek, Higher order Galerkin time discretizations and fast multigrid solvers for the heat equation, Journal of Numerical Mathematics 19 (1) (2011) 41–61.

[34] P. Benedusi, P. Ferrari, C. Garoni, R. Krause, S. Serra-Capizzano, Fast parallel solver for the space-time IgA-DG discretization of the diffusion equation, Journal of Scientific Computing (2021). doi:https://doi.org/10.1007/s10915-021-01567-z.

[35] K. Eriksson, C. Johnson, Adaptive finite element methods for parabolic problems ii: Optimal error estimates in $L_\infty L_2$ and $L_\infty L_\infty$, SIAM Journal on Numerical Analysis 32 (3) (1995) 706–740.

[36] D. Schötzau, T. P. Wihler, A posteriori error estimation for hp-version time-stepping methods for parabolic partial differential equations, Numerische Mathematik 115 (3) (2010) 475–509.

[37] C.-W. Shu, Discontinuous Galerkin method for time-dependent problems: survey and recent developments, Springer International Publishing, 2014, pp. 25–62.

[38] J. P. Keener, J. Sneyd, Mathematical physiology, Vol. 1, Springer, 1998.

[39] R. Speck, D. Ruprecht, M. Emmett, M. L. Minion, M. Bolten, R. Krause, A multi-level spectral deferred correction method, BIT Numerical Mathematics 55 (2015) 843–867.

[40] M. L. Minion, Semi-implicit projection methods for incompressible flow based on spectral deferred corrections, Appl. Numer. Math. 48 (3-4) (2004) 369–387.

[41] C. Hofer, U. Langer, M. Neumüller, R. Schneckenleitner, Parallel and robust preconditioning for space-time isogeometric analysis of parabolic evolution problems, SIAM Journal on Scientific Computing 41 (3) (2019) A1793–A1821.

[42] E. McDonald, A. Wathen, A simple proposal for parallel computation over time of an evolutionary process with implicit time stepping, in: Numerical Mathematics and Advanced Applications ENUMATH 2015, Springer, 2016, pp. 285–293.

[43] W. L. Briggs, V. E. Henson, S. F. McCormick, A Multigrid Tutorial, 2nd Edition, SIAM, Philadelphia, 2000.

[44] P. Benedusi, Parallel space-time multilevel methods with application to electrophysiology, Ph.D. thesis, Università della Svizzera italiana (2020).

[45] W. Pazner, P.-O. Persson, Approximate tensor-product preconditioners for very high order discontinuous Galerkin methods, Journal of Computational Physics 354 (2018) 344–369.

[46] S. Börm, R. Hiptmair, Analysis of tensor product multigrid, Numerical Algorithms 26 (3) (2001) 219–234.

[47] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc users manual, Tech. Rep. ANL-95/11 - Revision 3.12,

Argonne National Laboratory (2019).
URL https://www.mcs.anl.gov/petsc

[48] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc Web page, https://www.mcs.anl.gov/petsc (2019).
URL https://www.mcs.anl.gov/petsc

[49] P. Zulian, A. Kopaničáková, M. C. G. Nestola, A. Fink, N. Fadel, A. Rigazzi, V. Magri, T. Schneider, E. Botter, J. Mankau, R. Krause, Utopia: A C++ embedded domain specific language for scientific computing. Git repository, https://bitbucket.org/zulianp/utopia (2016).
URL https://bitbucket.org/zulianp/utopia

[50] Y. Notay, Rigorous convergence proof of space-time multigrid with coarsening in space, Numerical Algorithms (2021) 1–25.

[51] G. Horton, S. Vandewalle, A space-time multigrid method for parabolic partial differential equations, SIAM Journal on Sci. Comp. 16 (4) (1995) 848–864.

[52] M. L. Minion, Semi-implicit spectral deferred correction methods for ordinary differential equations, Commun. Math. Sci. 1 (3) (2003) 471–500.

[53] R. Schöbel, R. Speck, PFASST-ER: combining the parallel full approximation scheme in space and time with parallelization across the method, Computing and visualization in science 23 (1-4) (2020) 12. doi:10.1007/s00791-020-00330-5.
URL https://juser.fz-juelich.de/record/885633

[54] S. Götschel, M. L. Minion, An efficient Parallel-in-Time method for optimization with parabolic PDEs, SIAM J. Sci. Comput. 41 (6) (2019) C603–C626.

[55] A. Bourlioux, A. T. Layton, M. L. Minion, High-order multi-implicit spectral deferred correction methods for problems of reactive flow, J. Comput. Phys. 189 (2) (2003) 651–675.