No. 634 **December 2020**

Increased space-parallelism via time-simultaneous Newton-multigrid methods for nonstationary nonlinear PDE problems

J. Dünnebacke, S. Turek, C. Lohmann, A. Sokolov, P. Zajac

ISSN: 2190-1767

Jonas Dünnebacke*[1], Stefan Turek[1], Christoph Lohmann[1], Andriy Sokolov[1], and Peter Zajac[1]

[1]Institute of Applied Mathematics (LS III), TU Dortmund University, Vogelpothsweg 87, D-44227 Dortmund, Germany

Abstract

We discuss how 'parallel-in-space & simultaneous-in-time' Newton-multigrid approaches can be designed which improve the scaling behavior of the spatial parallelism by reducing the latency costs. The idea is to solve many time steps at once and therefore solving fewer but larger systems. These large systems are reordered and interpreted as a space-only problem leading to multigrid algorithm with semi-coarsening in space and line smoothing in time direction. The smoother is further improved by embedding it as a preconditioner in a Krylov subspace method. As a prototypical application, we concentrate on scalar partial differential equations (PDEs) with up to many thousands of time steps which are discretized in time, resp., space by finite difference, resp., finite element methods.

For linear PDEs, the resulting method is closely related to multigrid waveform relaxation and its theoretical framework. In our parabolic test problems the numerical behavior of this multigrid approach is robust w.r.t. the spatial and temporal grid size and the number of simultaneously treated time steps. Moreover, we illustrate how corresponding time-simultaneous fixed-point and Newton-type solvers can be derived for nonlinear nonstationary problems that require the described solution of linearized problems in each outer nonlinear step. As the main result, we are able to generate much larger problem sizes to be treated by a large number of cores so that the combination of the robustly scaling multigrid solvers together with a larger degree of parallelism allows a faster solution procedure for nonstationary problems.

1 Motivation

Modern High Performance Computing platforms feature a continuously growing number of cores, together with accelerator hardware, e.g. GPUs, while the performance of each core does barely increase or even stagnates: To efficiently use such systems the underlying numerical algorithms have to be more and more parallel. However, when dealing with time-dependent partial differential equations (PDEs) in the framework of initial value problems (IVP), the usual time-stepping approach is inherently sequential and does only allow spatial parallelization in each time step.

To be more precise: If we simulate such PDEs with a relatively low number of spatial degrees of freedoms (DOFs), but a very high number of time steps due to a long time

*jonas.duennebacke@math.tu-dortmund.de

horizon and/or small time steps, we can only exploit a certain degree of parallelism in each time step due to the small spatial problem size.

As a consequence, we cannot significantly speed up the complete simulation by parallel computing: If we utilize more and more cores, the run-time of the simulation will stagnate or even begin to increase when there are too few (spatial) unknowns left for each processor, because the communication time between the processors will outweigh the time that is needed for the actual computations in each process (see Fig. 1 for illustration).

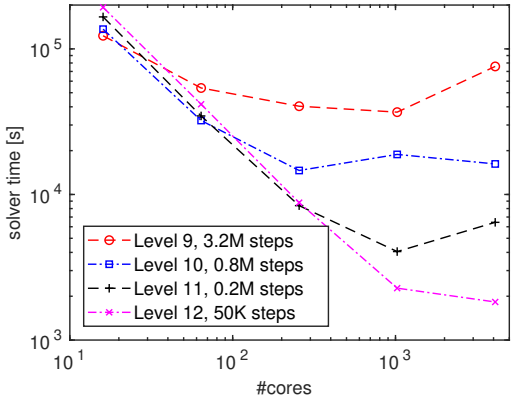


Figure 1: Strong scaling test: Typical solution time (y-axis) for an increasing number of cores (x-axis) with different mesh levels in space vs. number of time steps for nonstationary 2D problems discretized with bilinear FEM (Level i corresponds to mesh size $h = 2^{-i}$); the total number of unknowns (in the space-time domain) is the same in all 4 tests.

It is important to note that the main cost of communication is due to latency and not due to limited bandwidth. Vice versa, if the number of communication operations is reduced by communicating more data at once, the limited scaling behavior could be improved, so that more cores could be used efficiently in such simulations. To achieve this, we therefore have to abandon the usual sequential time-stepping which treats complete spatial problems, but one after the other.

One class of corresponding approaches is based on 'time-parallel' solvers in which case many methods are based on integrating ordinary differential equations (ODEs) parallel in time. The most prominent example of this group is Parareal [13] and its variants. Other notable examples are PFASST [2, 20] and MGRIT [3, 4, 8]. Another class of time-parallel methods is based on solving a global discrete system with multigrid methods. Since a simple geometric multigrid method as used for elliptic problems does not work for parabolic equations, there exist several different approaches. The first parabolic multigrid was developed by Hackbusch [7], while other forms of such schemes are the ones developed by Horton and Vandewalle [10] as well as the more recent variants [6, 9, 17, 27]. The method that is the closest to our approach (which is better described as time-simultaneous instead of time-parallel) is multigrid waveform relaxation. It was first published by Lubich and Ostermann [16] and has been studied in detail for different problems [21, 23, 24, 25]. For a more complete overview of parallel-in-time methods, we refer to [5].

2 Time-simultaneous multigrid solvers

In the following, we describe a geometric multigrid scheme that computes many time steps simultaneously but relies solely on spatial parallelization [1]. For illustration, we start with

a second order parabolic evolution equation

$$\partial_t u(x,t) - L(t)u(x,t) = f(x,t) \quad (x,t) \in \Omega \times (0,T) \tag{1}$$

where $L(t)$ is a linear elliptic operator in space for every $t \in (0,T)$ with some initial and boundary conditions.

To simplify the notation, we only consider linear one-step methods such as implicit Euler or Crank-Nicolson schemes. Moreover, we require a fixed spatial discretization by finite differences (FD) or finite elements (FE) for all time steps, so that the discrete system at each time level k can be written as

$$A_k \boldsymbol{u}_k = \boldsymbol{f}_k - B_k \boldsymbol{u}_{k-1} \quad , k = 1, \dots, K \tag{2}$$

with matrices $A_k, B_k \in \mathbb{R}^{N \times N}$ for $1 \le k \le K$ and a given discrete initial value $\boldsymbol{u}_0 \in \mathbb{R}^N$; $\boldsymbol{u}_k$ represents the vector of spatial DOFs at time level k. Here, $N \in \mathbb{N}$ is the number of spatial degrees of freedom and $K \in \mathbb{N}$ denotes the number of (simultaneously computed) time steps. This method can also be applied in combination with linear multistep methods: The only additional requirement is that the related systems in each time step, that means $A_k \boldsymbol{u}_k = r.h.s.$, can be efficiently solved by a spatial (geometric) multigrid method.

Now, we can gather the individual equations in (2) in an all-at-once system of the form

$$\underbrace{\begin{bmatrix} A_1 & & & \\ B_2 & A_2 & & \\ & \ddots & \ddots & \\ & & B_K & A_K \end{bmatrix}}_{=:\bar{A} \in \mathbb{R}^{NK \times NK}} \underbrace{\begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_K \end{bmatrix}}_{=:\bar{\boldsymbol{u}} \in \mathbb{R}^{NK}} = \underbrace{\begin{bmatrix} \boldsymbol{f}_1 - B_1 \boldsymbol{u}_0 \\ \boldsymbol{f}_2 \\ \vdots \\ \boldsymbol{f}_K \end{bmatrix}}_{=:\bar{\boldsymbol{f}} \in \mathbb{R}^{NK}} . \tag{3}$$

Then, the central idea is to reorder this system from a *space-major* ordering

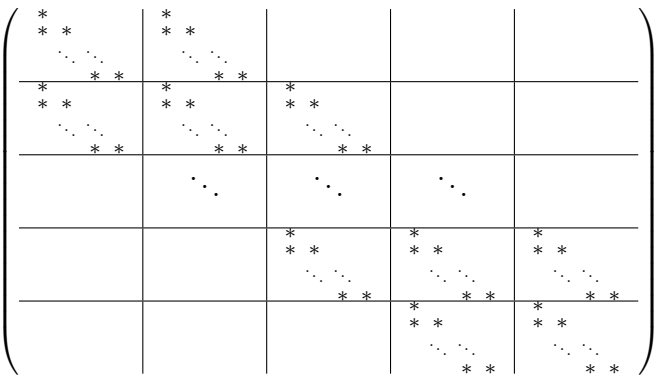
$$\bar{\boldsymbol{u}} = [u_{1,1}, \dots, u_{1,N}, u_{2,1}, \dots, u_{2,N}, \dots, u_{K,1}, \dots, u_{K,N}]^\top$$

to a *time-major* ordering

$$\boldsymbol{u} = [u_{1,1}, \dots, u_{K,1}, u_{1,2}, \dots, u_{K,2}, \dots, u_{1,N}, \dots, u_{K,N}]^\top$$

where $u_{k,i} = (\boldsymbol{u}_k)_i$ denotes the i-th spatial DOF at the k-th time step. The same reordering is applied to the vector $\bar{\boldsymbol{f}}$ and to the rows and columns of the matrix $\bar{A}$.

The reordered matrix $A \in \mathbb{R}^{NK \times NK}$ can be written as a block matrix whose block structure is the same as the structure of the matrices A_k and B_k in each time step. However, each (block) matrix entry is not a scalar value but a lower bi-diagonal $K \times K$ matrix. In the more general case of linear multi-step methods, each block is a lower triangular submatrix with a small bandwidth. For example, in the case of linear 1D finite elements, the stiffness and mass matrices are tridiagonal which means that the reordered global matrix A for one-step methods has the following block-tridiagonal structure:

$A =$ 

Summarizing, this new system is just based on grouping multiple equations into one large system and reordering. Therefore, it has the same solution as the initial time-stepping equations in (2), resp., in (3).

To solve this new system we simply apply a geometric multigrid method in space to the global system by treating each inner $(K \times K)$-block as a single matrix entry which is directly related to one corresponding spatial unknown, resp., grid point now containing all K time instances. Then, we may use a damped Jacobi smoother given by the iteration

$$\boldsymbol{u}^{m+1} = \boldsymbol{u}^m + \omega D^{-1}(\boldsymbol{f} - A\boldsymbol{u}^m)\,, \tag{4}$$

where $\omega > 0$ is a damping parameter and D is the (block) diagonal part of the reordered matrix A, to achieve a high level of parallelism as usual for Jacobi smoothing. Since we are formally treating the matrix A as a matrix of blocks, resp., submatrices, we have to use the complete diagonal blocks in the matrix

$$D = \begin{pmatrix} \# & & \\ & \# & \\ & & \ddots \\ & & & \# \end{pmatrix} \in \mathbb{R}^{NK\times NK} \quad \text{with} \quad \# = \begin{bmatrix} * & & & \\ * & * & & \\ & \ddots & \ddots & \\ & & * & * \end{bmatrix} \in \mathbb{R}^{K\times K} \tag{5}$$

resulting in a block-Jacobi smoother with block size K. Additionally, different smoothers which can be written in the form (4) are applicable in the same manner leading to block-smoothing procedures, too.

The corresponding grid transfer operators for performing multigrid iterations are constructed in an analogous way leading to *semi-coarsening in space* which means that the transfers in space can be applied to each time step independently so that the temporal grid remains the same across all grid levels. Consequently, we can write the prolongation matrix $P \in \mathbb{R}^{NK\times N_cK}$ and the restriction matrix $R \in \mathbb{R}^{N_cK\times NK}$ as the Kronecker products

$$P = P_s \otimes I_k \quad \text{and} \quad R = R_s \otimes I_k \tag{6}$$

with the identity matrix $I_K \in \mathbb{R}^{K\times K}$ and the spatial transfer matrices $P_s \in \mathbb{R}^{N\times N_c}$ and $R_s \in \mathbb{R}^{N_c\times N}$, which correspond to the usual grid transfer operators in a standard (space-only) geometric multigrid method. Here, N_c denotes the number of spatial DOFs on the coarser level.

With these modified smoothing and transfer operators we can apply the same multigrid algorithm as in each time step in (2), but now to solve the all-at-once system. As a consequence, we can treat K time steps simultaneously, while at the same time the dimension of the all-at-once system is K times greater than the dimension of the original spatial systems A_k in each time step which is the key to exploit a larger number of cores related to the given number of spatial grid points.

In our numerical experiments in the following sections, we also demonstrate the efficiency of different smoothing operators. For instance, instead of using a plain damped (block) Jacobi smoother, we may use BiCGSTAB (or GMRES) iterations for smoothing with matrix D in (4) as preconditioner. The reasoning behind this idea will be explained in the next subsection. Moreover, the total number of time steps $M = \frac{T}{\tau}$ that have to be computed to solve a problem in a given time interval $(0, T]$ and for a given time step size τ can be very large. In such cases it is not beneficial or even not possible to compute all M time steps simultaneously. Instead we will only solve $K < M$ steps simultaneously and perform such a 'macro time stepping' $\left\lceil \frac{M}{K} \right\rceil$ times, but in a sequential manner where the solution of the previous time slot can be used as initial value of the next one. To make this point clear: This does explicitly not expose any parallelism in time.

2.1 Intuitive explanation for small and large time steps

In the following, we provide a short intuitive understanding of two special cases that can help to tweak the algorithm in practice. To do this we consider the 1D heat equation with finite elements (and mass lumping) or finite differences as spatial discretization. In the most simplistic case of central differences as space discretization and the implicit Euler time discretization with equidistant spatial and temporal grids, the discrete scheme reads

$$\frac{1}{\tau}(u_{k,i} - u_{k-1,i}) - \frac{1}{h^2}(u_{k,i+1} - 2u_{k,i} + u_{k,i-1}) = f_{k,i} \tag{7}$$

with the spatial grid size h and the time step size τ. This shows that matrix entries belonging to the spatial derivatives are of the size $\mathcal{O}(h^{-2})$ and matrix entries belonging to the time derivative are of the size $\mathcal{O}(\tau^{-1})$. To describe the ratio between these values we introduce the 'anisotropy factor' $\lambda = \frac{\tau}{h^2}$ that is widely used in the convergence analysis of space-time multigrid methods [6, 10, 24]. For different discretizations or a different number of spatial dimensions, the same ratio can be used without losing its meaning as long as $L(t)$ is a second order elliptic differential operator. As this parameter depends on the temporal and spatial grids, it changes on different levels of the multigrid scheme. Furthermore, it can change locally on each level, if local grid refinement or space- and time-dependent coefficients are used. Consequently, the resulting method should yield convergence rates that are stable with respect to a wide range of values for λ.

In the case of $\lambda \nearrow \infty$, the matrix entries belonging to the spatial discretizations prevail and we obtain a discretized Poisson operator. This means that the prototype system has the global structure

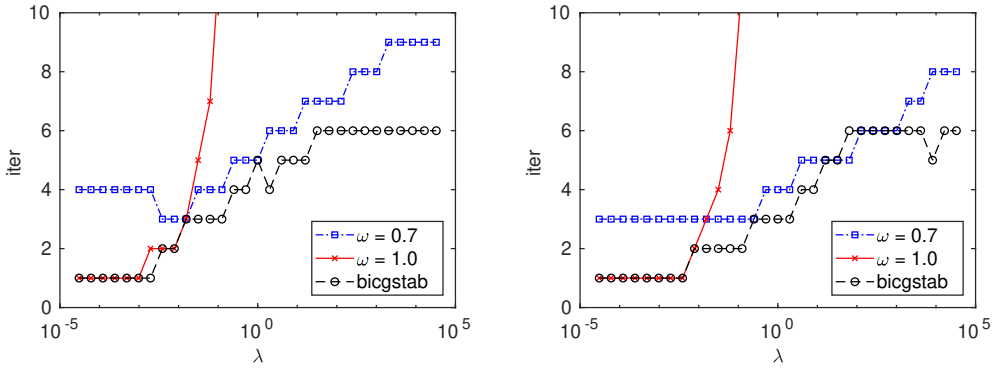
$$A = \tau^{-1} \left(\begin{array}{ccc|ccc|cc|cc} 1+2\lambda & & & -\lambda & & & & & & \\ -1 & 1+2\lambda & & & -\lambda & & & & & \\ & \ddots & \ddots & & & \ddots & & & & \\ & & -1 & 1+2\lambda & & & -\lambda & & & \\ \hline -\lambda & & & 1+2\lambda & & & & & & \\ & -\lambda & & -1 & 1+2\lambda & & & & & \\ & & \ddots & & \ddots & \ddots & & & & \\ & & & & & -1 & 1+2\lambda & & & \\ \hline & & & & & & & -\lambda & & \\ & & & & & & & & -\lambda & \\ & & & & & & & & & \ddots \\ & & & & & & & & & -\lambda \\ \hline & & & & & & -\lambda & & 1+2\lambda & \\ & & & & & & & -\lambda & -1 & 1+2\lambda \\ & & & & & & & \ddots & \ddots & \ddots \\ & & & & & & & & -1 & 1+2\lambda \end{array} \right)$$

where the absolute values of red entries are much larger than the black ones. If we ignore the significantly smaller values, each block of the global matrix becomes diagonal, so that the global system can be seen as K independent $N \times N$ systems for each time step. Then, using the time-simultaneous multigrid is equivalent to solving each independent problem with a (spatial) multigrid scheme on its own and it should not behave worse than the corresponding multigrid algorithm for the stationary problem with a convergence behavior independent of the spatial mesh size h (see Fig. 2). This consideration holds true also for all BDF-like time discretizations; with other linear one- or multistep methods the subdiagonal entries do not vanish, nevertheless the convergence behavior is similar.

In the opposite case $\lambda \searrow 0$, the (red) values of order $\mathcal{O}(\tau^{-1})$ dominate in the matrix:

$$A = \tau^{-1} \begin{pmatrix} 1+2\lambda & -\lambda & & \\ -1 & \ddots & \ddots & \\ & \ddots & -\lambda & \\ & -1 & 1+2\lambda & \end{pmatrix}$$

Ignoring all small values leads to a block diagonal global system if the mass matrix is diagonal. Such a diagonal mass matrix arises naturally in FD discretizations or can be obtained by using mass lumping in the FE case. However, for such block diagonal matrices, the described block-Jacobi preconditioner ($\omega = 1.0$) becomes exact and the corresponding multigrid solver converges in one step (see Fig. 2 for illustration).



(a) $h = 1/256$ (b) $h = 1/1024$

Figure 2: MG iterations for the heat equation with different smoothers, $K = 100$

Since we do not want to choose the damping parameter ω based on λ manually, a well-known remedy is to use modified smoothers like the Krylov subspace methods BiCGSTAB [22] and GMRES [19] with the described block diagonal matrix D as preconditioner. These methods yield convergence rates similar to the Jacobi smoothing with comparable effort (also w.r.t. parallel treatment), while they can recover the fast convergence in the case of $\lambda \searrow 0$ with a diagonal mass matrix. This is demonstrated in Fig. 2, too, where this prototype system is solved for different time step sizes resulting in different anisotropy factors λ. Here, the standard block-Jacobi smoother uses 2 smoothing steps while the BiCGSTAB smoother uses only 1 step to have comparable cost.

2.2 Relation to multigrid waveform relaxation

Although we developed the previously described algorithm independently, the resulting schemes can be interpreted as multigrid waveform relaxation methods [16]. These schemes

are based on discretizing the evolution equation in space and applying a multigrid splitting to the stiffness matrix of the semi-discrete ODE system. For finite elements, a similar splitting has to be applied to the mass matrix to be able to independently solve the ODEs that arise in every step of the multigrid algorithm. The resulting method is equivalent to our time-simultaneous method, if the same multigrid splitting with a smoother of type (4) is used for the mass and stiffness matrix and if the same linear multi-step method is used to solve each ODE problem in the multigrid approach. Therefore, we do not provide a more detailed convergence analysis, but rather refer to the literature on multigrid waveform relaxation in [11, 24] for further details.

Remark 1. *As was shown by Janssen and Vandewalle [11] for the time discrete multigrid waveform algorithm for finite elements and a time constant operator L, the algorithm converges and yields the same asymptotic convergence rates as the traditional multigrid algorithm in the time-stepping case, if the coarse grid system matrix and the preconditioning matrices D_l on each level l are regular. Since our algorithm for the plain block-Jacobi smoother is equivalent, this result holds true.*

Consequently, the asymptotic convergence rate (i.e. the spectral radius of the iteration matrix) is bounded, but that does not imply that the residual reduction w.r.t. an arbitrary norm is bounded in each iteration. Therefore, this result does not show that the number of iterations to achieve a given residual reduction is independent of the number of simultaneously treated time steps K. Nevertheless, the monotone convergence behavior of the block-Jacobi smoothing can be proven, too, and will be part of a forthcoming paper. Moreover, the described BiCGSTAB smoother does not fit into the traditional multigrid waveform relaxation framework since this smoother itself and the resulting multigrid iteration are not linear so that an analysis based on the iteration matrix is not possible anymore and hence the convergence result does not hold in that case. However, the numerical examples demonstrate the expected numerical convergence behavior.

2.3 Computational characteristics

The number of necessary floating point operations (FLOPs) in each iteration of the described time-simultaneous multigrid approach with K blocked time steps is still of linear complexity w.r.t. the total number of unknowns $N \cdot K$. Compared with the sequential time-stepping case where an $N \times N$ system is solved by a multigrid method in each of the K time steps, the cost of the grid transfer per iteration and time step is the same. The cost of the (space-time) residual calculation is slightly higher, because the global matrix has a moderately higher bandwidth in the time-simultaneous case: The bandwidth increases by a factor of $L+1$ at most where L is the width of the multistep method. For the smoothing procedure, the residual needs to be calculated, and the preconditioner has to be applied. That means we have to solve N_l block systems of size $K \times K$ on level l, but each block is a lower triangular matrix of bandwidth $L+1$ and can be treated independently from each other. Thus, we can solve these systems by forward substitution with linear complexity $O(K)$, too.

Summarizing, the computational cost per iteration of the time-simultaneous multigrid approach and the cost of the sequential time-stepping scheme with the multigrid solver in each time step only differ by a moderate constant (multiplicative) factor. And to make this clear: The described multigrid approach does not exploit any temporal parallelization. Nevertheless, the time-simultaneous method is beneficial because the number of required data communications per MG iteration and time step is reduced by a factor of K^{-1} since one multigrid step yields the solution of K time steps. Consequently, the latency induced

communication time can be lowered and better scaling of the spatial parallelization is possible. If the number of necessary multigrid iterations does not increase significantly with increasing K the total computational cost is only slightly higher and the improved scalability enables a speedup compared to sequential time-stepping.

Remark 2. *Even though parallelization in the time direction is not trivial, it is possible to apply parallel triangular solvers. This has been shown for multigrid waveform relaxation (c.f. [26]), but will not be considered in this paper.*

2.4 Extension to nonlinear problems

In the previous sections only linear evolution equations have been discussed, but in real world applications one often has to deal with nonlinear problems. In the following, we consider nonlinear time-dependent equations of the type

$$\partial_t u(x,t) - L(u(x,t))u(x,t) = f(x,t) \quad (x,t) \in \Omega \times (0,T)$$

with suitable initial and boundary conditions, where $L(v(x,t))$ is a linear differential operator for every $v(x,t)$ and every $(x,t) \in \Omega \times (0,T)$ (see the subsequent examples). In the traditional time-sequential approach we typically discretize the equation in time and then we only need to solve a nonlinear PDE to obtain the solution $u(x,t_i)$ at the new time step t_i when the solution $u(x,t_j)$ is already computed in the last time steps t_j with $j < i$. However, in the time-simultaneous approach this is not possible since we do not know the solution at the previous time steps.

Instead, we have to perform an outer nonlinear iteration where we solve a corresponding linearized PDE problem over the complete space-time domain in each step. First of all, we consider the (standard) fixed-point method where the new approximation $u^{(j+1)} = u^{(j+1)}(x,t)$ is calculated by solving

$$\partial_t u^{(j+1)} - L(u^{(j)})u^{(j+1)} = f \quad (x,t) \in \Omega \times (0,T)$$

where $u^{(j+1)}$ satisfies the same initial and boundary conditions as the solution u.

Another nonlinear iteration is the Newton method. Here, we solve

$$\partial_t v^{(j+1)} - K(u^{(j)})v^{(j+1)} = F^{(j)} \quad (x,t) \in \Omega \times (0,T)$$

where $K(u^{(j)})v^{(j+1)}$ corresponds to the Fréchet derivative of $L(u)u$ evaluated in $u^{(j)}$ and applied to $v^{(j+1)}$, while $F^{(j)}$ denotes the nonlinear residual:

$$F^{(j)} = \partial_t u^{(j)} - L(u^{(j)})u^{(j)} - f \tag{8}$$

The correction $v = v^{(j+1)}$ has to satisfy homogeneous boundary conditions so that the new approximation given by $u^{(j+1)} = u^{(j)} - v$ satisfies the boundary conditions if the initial guess u_0 also satisfies them.

Summarizing these two nonlinear approaches, we have to calculate a residual vector in each nonlinear iteration step and nonstationary linearized PDE problems have to be handled which correspond to the actual fixed-point, resp., Jacobian matrix. To solve these nonstationary, but linear problems, we can apply the described time-simultaneous multigrid approaches from the previous sections. From a theoretical point of view, the standard convergence results for classical fixed-point and Newton methods, but now applied onto the discretized space-time problem, can be obtained. However, how this is applied in practice will be demonstrated in the next section where we analyze numerically the behavior of such

time-simultaneous nonlinear iteration methods for several prototypical examples. Moreover, we shortly discuss additional aspects like adaptive step-length control techniques and also strategies for generating appropriate starting values in the time-simultaneous framework which can significantly influence the resulting numerical convergence behavior, too.

3 Numerical results

In the remaining part we provide numerical results for different linear and nonlinear test problems to illustrate the potential of this new numerical approach. For illustration purposes, all tests are performed on uniform meshes where each element has the same size. Furthermore, for simplicity, we use constant time step sizes.

All tests, that focus on the numerical behavior, are done in MATLAB while the solver for the 2D heat equation is implemented using the *C++* based *Finite Element Analysis Toolbox 3* [1] to test the performance and parallel scaling capabilities of this method. The strong scaling tests were executed on the LiDO3 cluster [2] which features 316 nodes with 2 Intel Xeon E5-2640v4 and 64 GB RAM on each node (see Fig. 4).

3.1 Linear problems

As first linear test problem, the heat equation with the following setting

$$\begin{aligned} \partial_t u(x,t) - \Delta u(x,t) &= 1 + 0.1\sin(t) && (x,t) \in \Omega \times (0,T)\,, \\ u(x,t) &= 0 && (x,t) \in \partial\Omega \times (0,T)\,, \\ u(x,0) &= 0 && x \in \Omega\,, \end{aligned}$$

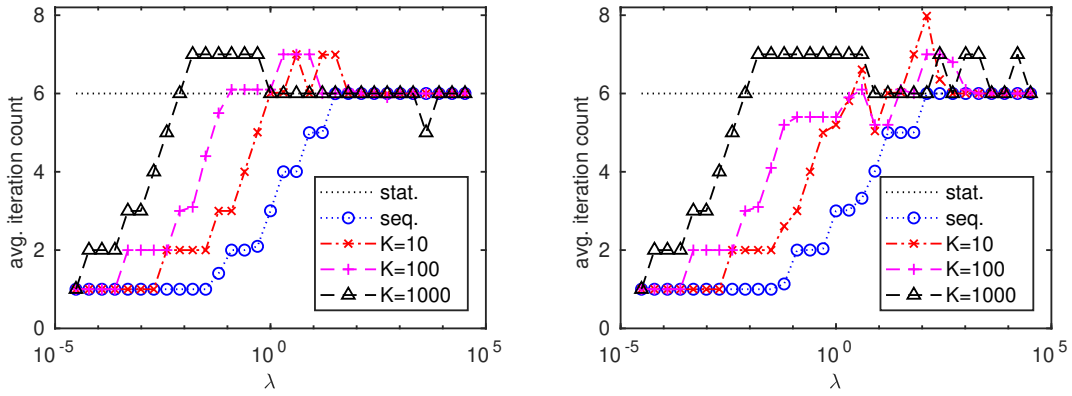
on the 2D unit-square domain $\Omega = (0,1)^2 \subset \mathbb{R}^2$ is used which is discretized with bilinear finite elements and mass lumping in space and with Crank-Nicolson in time. The introduced time-simultaneous multigrid algorithm (here: F-cycle) uses one BiCGSTAB iteration (preconditioned with the block diagonal matrix) as pre- and post-smoothing procedure. For each test the total number of time steps is $M = 1000$, while the number of simultaneously computed time steps K changes. For example, in the case of $K = 100$, 10 sequentially executed multigrid solves are needed to compute the solution on the full time domain. This ensures that for a fixed $\lambda = \frac{\tau}{h^2}$ the same problem is solved regardless of the number of simultaneous time steps K. Then, the shown number of multigrid iterations is the average over those 10 solver steps. Additional simulations using the sequential time-stepping scheme and solving the corresponding stationary equation ('stat.') are performed to obtain reference results (see Fig. 3). Obviously, changing λ in combination with a fixed spatial grid size and a fixed number of total time steps implies that the time step size τ and the time interval $(0,T)$ are changed.

The number of multigrid iterations for very small and large time steps behaves as expected: For $\lambda \gg 1$ the number of iterations needed to reduce the norm of the global residual vector by a factor of 10^{-8} is independent of the block size K and corresponds to the number of iterations that are needed in the stationary test (which means the solution of one Poisson problem on the given spatial mesh). In the case of $\lambda \ll 1$, the multigrid algorithm converges in one step and in-between the number of iterations is at most slightly higher than in the case of large time steps. The only major difference between different

[1] `http://www.featflow.de/en/software/feat3.html`
[2] `https://www.lido.tu-dortmund.de/`

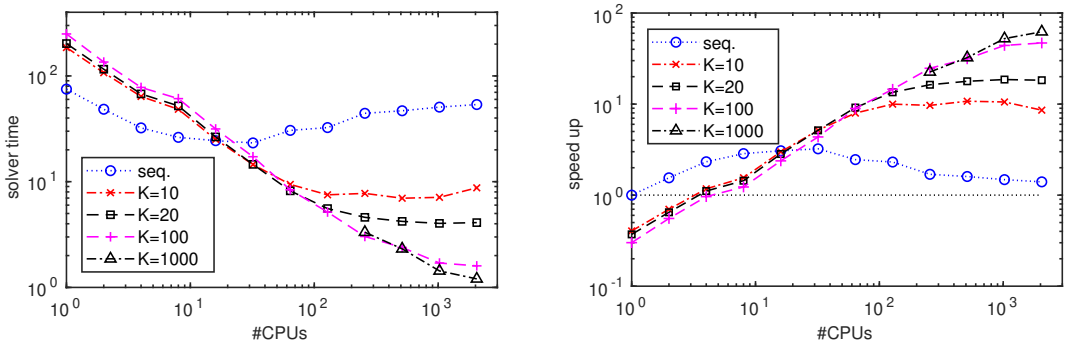
block sizes is that the transition area between 'small' and 'large' time steps shifts to smaller time steps if the block size K, and hence the number of simultaneously treated time steps increases.

Comparing the results of different spatial grids shows that the grid size only affects the convergence speed due to its influence on λ. Other linear multistep methods, higher order finite elements and different test cases show a similar qualitative behavior.



(a) $h = 1/32$ (b) $h = 1/128$

Figure 3: Number of multigrid iterations in the heat equation test case with different time step sizes and block dimensions



(a) solver times [s] (b) speedup vs. sequential time-stepping without parallelization

Figure 4: Strong scaling test: Solver time and speed up for an increasing number of cores, $h = 1/256$ (leading to appr. 66.000 grid points), $\tau = 0.001$, $T = 1$

The strong scaling tests in Fig. 4 use the same multigrid setup and the same test problem with 256×256 quadratic elements ($h = 1/256$) and time step size $\tau = 0.001$. In the sequential time-stepping case the lowest time of 23.3 seconds to solve the complete system, that means for 1000 time steps, can be achieved using 32 cores, while more yield no benefit for such (in space) small-scale problems. Due to the computational overhead, the time-simultaneous approach needs approximately twice the time for low core counts but provides much better scaling. For a small block size of $K = 20$ time steps the simulation time can be already significantly improved, while greater block sizes may even further improve the scaling behavior. With 2048 cores and $K = 1000$ simultaneous steps we achieve a speedup by a factor of 27 per multigrid iteration, but we need 7 iterations,

whereas the sequential time-stepping solver only needs appr. 5 multigrid iterations on average. This results in a overall speedup factor of 19 leading to a solver time of 1.2 seconds.

3.1.1 Convection-Diffusion equation

The second linear test case is the (1D) nonstationary convection-diffusion equation

$$\partial_t u(x,t) - \varepsilon \partial_{xx} u(x,t) + c(x,t)\partial_x u(x,t) = f(x,t) \qquad (x,t) \in (0,1) \times (0,T)\,,$$

with homogeneous Dirichlet boundary conditions, the initial condition

$$u(x,0) = \exp(4(x-x^2)) - 1\,,$$

the space- and time-dependent velocity field

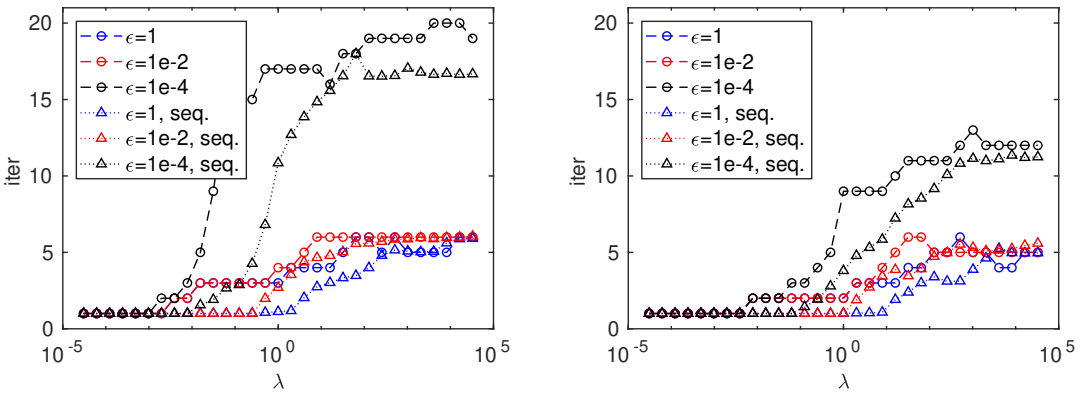
$$c(x,t) = \sin(\pi x)\sin(2t)$$

and the forcing function

$$f(x,t) = \sin(\pi t)\,.$$

The PDE is discretized using finite differences in space with a first-order upwind discretization for the convection term and the implicit Euler discretization in time. The time-simultaneous multigrid solver uses one BiCGSTAB smoothing step (with the described (block) Jacobi preconditioner). The following results are computed using a W-cycle. Other cycles lead to a larger increase of iterations for both, the time-stepping and the time simultaneous solver, when the diffusion coefficient is small. In contrast to the previous tests, the number of simultaneous time-steps is fixed and equal to the total number of time steps $K = M = 100$.

Each simulation is executed with the time-simultaneous multigrid method and with the corresponding sequential time-stepping where a standard (geometric) multigrid solver with the same setup is used as linear solver in each time step. The results of the sequential time-stepping approach are provided as reference (labeled with 'seq.' in Fig. 5).



(a) $h = 1/256$ (b) $h = 1/1024$

Figure 5: Convection-diffusion equation with $M = K = 100$ time steps for the sequential ('seq.') and the time-simultaneous approach

The results show again a similar behavior for both methods (see Fig. 5): The number of necessary multigrid steps to reduce the residual by a factor of 10^{-8} is bounded for large

time steps, and for small step sizes only one iteration is needed. Decreasing the diffusion parameter ε leads (as can be expected) to a slower convergence speed but this is also true for the sequential time-stepping case and is likely caused by the Jacobi smoother which is not perfectly suitable for convection-dominated problems. Therefore, this effect is less dominant on finer spatial grids. The biggest difference between both methods is that the transition range between higher and lower values for λ shifts to smaller time steps in comparison with the sequential time-stepping approach. When the diffusion parameter ε is small, this behavior is more noticeable.

3.2 Nonlinear problems

To demonstrate that this new time-simultaneous approach can also be used to solve nonlinear PDEs while exploiting the reduced latency costs in the linear subproblems, we show numerical results for problems of three different types.

3.2.1 Viscous Burgers' equation

The first nonlinear test problem is the viscous Burgers' equation

$$\partial_t u(x,t) - \varepsilon \partial_{xx} u(x,t) + u(x,t)\, \partial_x u(x,t) = f(x,t) \qquad (x,t) \in (0,1) \times (0,T) \qquad (9)$$

in one spatial dimension with a given initial condition $u(x,0) = u_0(x) = \max\{1 - 5x, 0\}$, Dirichlet boundary conditions $u(0,t) = g(t) = 1$ and $u(1,t) = h(t) = 0$, with $t \in (0,T)$. In the fixed-point iteration the new approximation $u^{(j+1)} = u^{(j+1)}(x,t)$ is determined as solution of the linear convection-diffusion problem

$$\begin{aligned} \partial_t u^{(j+1)} - \varepsilon \partial_{xx} u^{(j+1)} + u^{(j)} \partial_x u^{(j+1)} &= f(x,t) && (x,t) \in (0,1) \times (0,T)\,, \\ u^{(j+1)}(x,0) &= u_0(x) && x \in (0,1)\,, \\ u^{(j+1)}(0,t) = g(t)\,,\ u^{(j+1)}(1,t) &= h(t) && t \in (0,T)\,, \end{aligned}$$

whereas the update $u^{(j+1)} = u^{(j)} - v^{(j+1)}$ in the Newton method is given by the linear convection-diffusion-reaction problem in each outer iteration step

$$\begin{aligned} \partial_t v^{(j+1)} - \varepsilon \partial_{xx} v^{(j+1)} + v^{(j+1)} \partial_x u^{(j)} + u^{(j)} \partial_x v^{(j+1)} &= F^{(j)} && (x,t) \in (0,1) \times (0,T)\,, \\ v^{(j+1)}(x,0) &= 0 && x \in (0,1)\,, \\ v^{(j+1)}(0,t) = v^{(j+1)}(1,t) &= 0 && t \in (0,T)\,, \end{aligned}$$

with

$$F^{(j)} = \partial_t u^{(j)} - \varepsilon \partial_{xx} u^{(j)} + u^{(j)} \partial_x u^{(j)} - f\,.$$

Here, we use a FD discretization with upwinding in space and the implicit Euler scheme in time. As linear solver we use the same multigrid configuration as for the convection-diffusion equation in section 3.1.1. The time-simultaneous solvers treat all time steps as once, so the number of simultaneously computed time steps is equal to the total number of time steps $M = K = \frac{T}{\tau}$.

It is obvious that the number of necessary fixed-point iterations, denoted by it, to achieve a global residual reduction by 10^{-6} depends on the simulated time horizon in the case of small diffusion coefficients ε which render the problem more nonlinear (see Tab. 1). For example, in the case of $T = 0.1$, $\varepsilon = 10^{-3}$ and $\tau = 0.005$, the fixed-point iteration achieves the desired residual reduction after 9 steps, but 37 iterations are needed for $T = 0.4$. If we extend the time interval even further to $T = 1.0$, the time-simultaneous

		$\epsilon = 1$				$\epsilon = 10^{-2}$				$\epsilon = 10^{-3}$			
T	τ	it	it_{ref}	mg	mg_{ref}	it	it_{ref}	mg	mg_{ref}	it	it_{ref}	mg	mg_{ref}
0.1	0.050	4	4.00	6.50	6.38	10	9.00	6.70	6.50	13	11.50	7.15	7.04
0.1	0.005	4	3.00	6.25	6.33	8	4.15	6.75	6.67	9	4.45	7.33	6.93
0.1	0.001	4	2.99	6.25	6.46	7	3.00	6.86	6.38	8	3.00	8.38	6.01
0.4	0.050	4	4.00	6.25	6.47	16	9.75	6.81	6.65	26	13.75	8.38	7.94
0.4	0.005	4	3.00	6.25	6.36	15	4.79	6.53	6.44	37	6.85	8.59	7.24
0.4	0.001	4	2.54	6.25	6.72	15	3.00	6.53	6.26	41	4.18	9.46	6.38
1.0	0.050	5	4.00	6.40	6.49	23	9.30	6.52	6.60	45	14.10	8.58	8.16
1.0	0.005	5	3.00	6.40	6.54	25	4.92	6.40	6.05	>50	7.63	–	7.49
1.0	0.001	5	2.22	6.40	6.87	25	3.00	6.40	5.76	>50	4.67	–	6.65

(a) Fixed-point method

		$\epsilon = 1$				$\epsilon = 10^{-2}$				$\epsilon = 10^{-3}$			
T	τ	it	it_{ref}	mg	mg_{ref}	it	it_{ref}	mg	mg_{ref}	it	it_{ref}	mg	mg_{ref}
0.1	0.050	2	2.00	6.00	7.00	3	3.00	6.33	7.00	3	3.50	7.67	7.57
0.1	0.005	2	2.00	6.50	7.00	3	2.55	6.67	7.06	3	2.80	8.33	8.27
0.1	0.001	2	2.00	6.50	7.00	3	2.00	6.67	7.00	3	2.00	10.00	6.76
0.4	0.050	2	2.75	6.00	6.73	3	3.75	6.00	7.00	3	4.50	8.33	8.56
0.4	0.005	2	2.00	6.50	7.00	3	2.89	6.67	7.01	4	3.45	10.25	8.40
0.4	0.001	2	2.00	6.50	7.00	3	2.00	6.67	7.26	4	2.63	12.00	7.06
1.0	0.050	2	2.90	6.00	6.69	3	3.90	6.67	7.01	3	4.80	9.00	8.86
1.0	0.005	2	2.00	6.50	7.00	3	2.96	6.33	6.80	4	3.78	12.00	8.62
1.0	0.001	2	2.00	6.50	7.00	4	2.00	6.50	7.40	5	2.85	14.00	7.23

(b) Newton method

Table 1: Burgers' equation: number of nonlinear iterations, $h = \frac{1}{2048}$

fixed-point iteration needs 45 iterations in the case of $\tau = 0.05$ and for smaller time steps the nonlinear solver does not reach the convergence criterion in 50 iteration. On the other hand, the averaged number of fixed-point iterations per time step it_{ref} is only 14.1 in the sequential time-stepping approach with time step size $\tau = 0.05$ and decreases for smaller time steps. Therefore, a time-simultaneous fixed-point iteration is not suitable for the Burgers' equation in the case of small viscosity since the number of nonlinear iterations clearly depends on the final simulation time T which is not surprising due to the physical information transport in this Burgers' problem. However, the number of time-simultaneous multigrid steps in each nonlinear fixed-point iterations remains stable as expected.

In contrast, the Newton scheme can provide much faster convergence, especially if the initial starting values are close enough to the solution. Then, the typical quadratic convergence behavior can be observed. Thus, we compute the solution for the same problem in a nested manner with $2h$, τ and 2ε and use the result as initial guess for the simulation with the grid sizes τ, h and the viscosity parameter ε. Increasing the viscosity leads to a more diffusive profile that can be appropriately resolved on the coarser meshes. It also improves the linear multigrid solver as the problems on the coarser meshes are less convection-dominant. Using this starting value procedure, the number of iterations shows only a slight increase if a longer time horizon is calculated simultaneously since the quadratic convergence behavior of Newton methods can be exploited. In these tests at most 5 iterations are necessary to achieve the desired residual reduction.

The averaged number of multigrid steps in each nonlinear iteration, denoted by mg,

increases with a smaller viscosity, but is still more or less independent of the size K of simultaneously treated time steps. In the most difficult test case with $T = 1.0$, $\tau = 0.001$ and $\varepsilon = 10^{-3}$, 5 Newton iterations with an averaged number of 14 (time-simultaneous) multigrid iterations are needed to solve the complete space-time problem using the nested starting strategy. This means that a total number of $14 \cdot 5 = 70$ multigrid iterations are needed to solve the fully nonlinear time-dependent problem. In the sequential time-stepping case we only need 2.85 Newton iterations on average with an averaged number of $mg_{ref} = 7.23$ multigrid iterations resulting in roughly $7.23 \cdot 2.85 = 20.6$ multigrid iterations per time step. Overall the time-simultaneous approach for this nonstationary nonlinear problem needs appr. $\frac{70}{20.6} = 3.4$ times the number of multigrid iterations in comparison with the sequential time-stepping solver but the previous scaling results (see Fig. 4) show that with 1000 simultaneously treated time steps a bigger speedup per multigrid iteration is possible.

3.2.2 Allen-Cahn equation

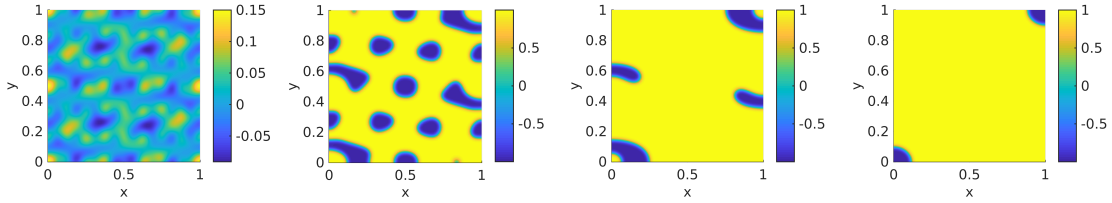
As second nonlinear problem we consider the Allen-Cahn equation for $u = u(x,t)$

$$\begin{aligned} \partial_t u - \Delta u + \frac{1}{\varepsilon^2} u(u^2 - 1) &= 0 && (x,t) \in \Omega \times (0,T), \\ \partial_n u &= 0 && (x,t) \in \partial\Omega \times (0,T), \\ u(x,0) &= u_0(x) && x \in \Omega, \end{aligned}$$

on the unit square domain $\Omega = (0,1)^2$ with $\varepsilon = 0.01$. Here, we use the initial condition

$$\begin{aligned} u_0(x) = 0.05\big(&\cos(6\pi x)\cos(4\pi y) + (\cos(8\pi x)\cos(6\pi y))^2 \\ &+ \cos(2\pi x - 10\pi y)\cos(4\pi x - 2\pi y)\big). \end{aligned}$$

The qualitative behavior of the solution at different times is shown in Fig. 6.



(a) $t = 0.0$ (b) $t = 0.001$ (c) $t = 0.004$ (d) $t = 0.016$

Figure 6: Qualitative behavior of the solution at different time steps

Here, Newton linearization leads to the following reaction-diffusion PDE for $v = v(x,t)$

$$\begin{aligned} \partial_t v - \Delta v + \frac{1}{\varepsilon^2}\Big(3{u^{(j)}}^2 - 1\Big)v &= \partial_t u^{(j)} - \Delta u^{(j)} + \frac{1}{\varepsilon^2} u^{(j)}\Big({u^{(j)}}^2 - 1\Big) && (x,t) \in \Omega \times (0,T), \\ \partial_n v(x,t) &= 0 && (x,t) \in \partial\Omega \times (0,T), \\ v(x,0) &= 0 && x \in \Omega, \end{aligned}$$

which has to be solved via the time-simultaneous (linear) multigrid approach in order to compute the Newton update $u^{(j+1)} = u^{(j)} - v$. For the discretization in time and space, we use bilinear finite elements in space and the Implicit Euler scheme.

For this test case, the results show a strong dependency of the number of nonlinear iterations from the starting values which can be expected for the Newton method. To

visualize the corresponding convergence behavior in time we calculate the global residual vector $\boldsymbol{r}_j$ after the j-th Newton iteration, which corresponds to the discretized version of the residual $F^{(j)}$ in equation (8). Then, we split that vector and compute the spatial norm in each time step, so that we can quantify how much each time step contributes to the global residual. So the residual at time step k after the i-th Newton iteration is given by

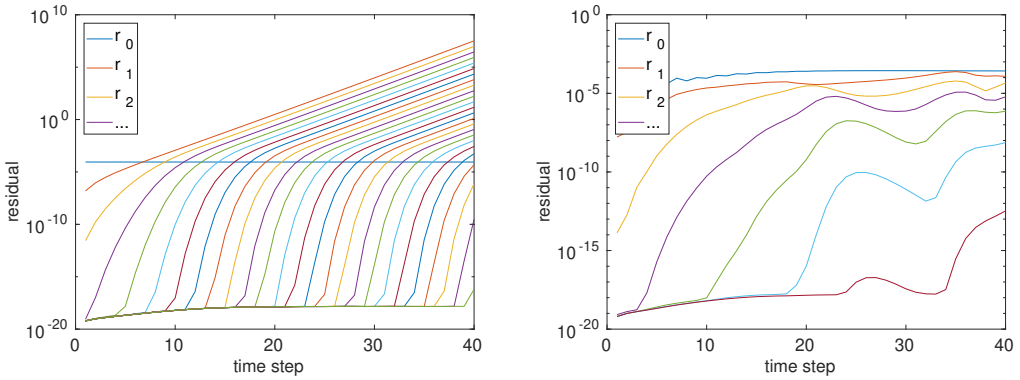
$$r_j(t_k) = \Big(\sum_{l=1}^{N} \boldsymbol{r}_j(x_l, t_k)^2 \Big)^{\frac{1}{2}} \quad .$$

It should be noted that a residual of (almost) 0 in a single time step does not imply that the solution is reached if the residual at a previous time is not 0.

If the prescribed initial condition is used as starting value at all time steps, the norm of the residual declines only in the very first of the simultaneously computed time steps, whereas the residual in the later time steps even increases significantly (see Fig. 7a). Then, the residual decreases in one time step after another. This implies that increasing the time horizon T and therefore increasing the number of simultaneously computed time steps leads to even more nonlinear iterations.

Better results can be achieved by using a nested procedure where the starting value is computed by solving (also in a time-simultaneous manner) the same problem, but with a coarser discretization. In Fig. 7b the solution for mesh size $h = 1/50$ and $\tau = 5 \times 10^{-5}$ is used as a starting value to solve the problem with refined grid sizes $h = 1/100$ and $\tau = 2.5 \times 10^{-5}$. In this case the discrete residual decreases in all time steps in all iterations which corresponds better to the expected quadratic convergence behavior of the Newton solver. Consequently, only 5 iterations are needed to achieve a global residual reduction by 10^{-8}.

In Tab. 2, corresponding results for different time step sizes τ and different end points T are shown. In each simulation all time steps are computed using the solution computed with the time step size 2τ as start values. The iteration counts of the Newton scheme are stable and nearly independent of the number of simultaneously computed time steps if the time step size is small enough. Otherwise the initial guess is not good enough, although it portrays the qualitative behavior of the solution. Moreover, if we choose over-refined meshes in space and time, it is possible to use the solution from a coarser discretization in space and time as initial guess (see Tab 3).



(a) initial condition as starting value

(b) nested start strategy

Figure 7: Global residuals per time step in each iteration, $h = 1/100$, $\tau = 2.5 \times 10^{-5}$, $T = 0.001$

h	τ	$T = 0.001$	$T = 0.004$	$T = 0.016$
1/100	1/10000	16	31	81
1/100	1/20000	12	14	18
1/100	1/40000	6	6	7
1/100	1/80000	5	5	5
1/100	1/160000	4	4	5

Table 2: Number of nonlinear iterations: Solution with h and 2τ used as initial guess

h	τ	$T = 0.001$	$T = 0.004$	$T = 0.016$
1/100	1/10000	16	31	81
1/200	1/20000	12	14	17
1/400	1/40000	6	6	9
1/800	1/80000	5	5	6

Table 3: Number of nonlinear iterations: Solution with $2h$ and 2τ used as initial guess

3.2.3 Nonlinear stabilization method for advection problems

In this final example, we focus on a nonlinear stabilization technique for the (pure) linear advection problem in one dimension with periodic boundary conditions

$$\begin{aligned} \partial_t u(x,t) + \partial_x u(x,t) &= 0 && (x,t) \in (0,1) \times (0,T), \\ u(0,t) &= u(1,t) && t \in (0,T), \\ u(x,0) &= \chi_{(0.1,0.5)}(x) && x \in (0,1). \end{aligned}$$

It is well known that the continuous and piecewise linear Galerkin approximation of this problem might be polluted by spurious oscillations especially if the exact solution possesses discontinuities. Therefore, stabilization techniques are commonly used to prevent the occurrence of unphysical overshoots and undershoots. For example, the so called algebraic flux correction (AFC) methodology introduced by Kuzmin and Turek [12] adds algebraically defined artificial diffusion operators to the Galerkin discretization to provably guarantee the validity of discrete maximum principles. In this case, the time-dependent degrees of freedom $(u_i)_{i=1}^N : [0,T] \to \mathbb{R}^N$ of the continuous and piecewise linear finite element function u_h solve the semi-discrete problem [15, Section 4.5.2]

$$(\sum_j m_{ij}) \mathrm{d}_t u_i - \sum_j k_{ij} u_j + \sum_{j \neq i} \dot{\alpha}_{ij} m_{ij} (\dot{u}_j - \dot{u}_i) - \sum_{j \neq i} (1 - \alpha_{ij}) d_{ij} (u_j - u_i) = 0 \qquad \forall i, \ (10)$$

where m_{ij} and k_{ij} denote the entries of the mass and convection matrices while $d_{ij} = \max(-k_{ij}, 0, -k_{ji})$. The auxiliary quantity $\dot{u} \in \mathbb{R}^N$ approximates the time derivative of the degrees of freedom $(\mathrm{d}_t u_i)_{i=1}^N$ as follows:

$$\dot{u}_i = \Big(\sum_j m_{ij}\Big)^{-1} \Big(\sum_j k_{ij} u_j + \sum_{j \neq i} d_{ij} (u_j - u_i)\Big) \quad \forall i$$

Inspired by [14, Section 7.2], we define the correction factors $\alpha_{ij}, \dot{\alpha}_{ij} \in [0,1]$ by

$$\alpha_{ij} = \beta_i \beta_j, \quad \beta_i = 1 - \max\Big(0, 1 - q\frac{Q_i^+ Q_i^-}{(P_i+\epsilon)^2}\Big)^3,$$
$$\dot{\alpha}_{ij} = \dot{\beta}_i \dot{\beta}_j, \quad \dot{\beta}_i = 1 - \max\Big(0, 1 - \dot{q}\frac{Q_i^+ Q_i^-}{(\dot{P}_i+\epsilon)^2}\Big)^3,$$

$$Q_i^{\pm} = \sum_{j\neq i, m_{ij}>0} d_{ij} \frac{\max\big(0, \pm(u_j - u_i)\big)^3}{(u_j-u_i)^2+\epsilon} \approx \sum_{j\neq i, m_{ij}>0} d_{ij} \max\big(0, \pm(u_j-u_i)\big),$$
$$P_i = \sum_{j\neq i, m_{ij}>0} d_{ij} \sqrt{(u_j-u_i)^2+\epsilon} \qquad \approx \sum_{j\neq i, m_{ij}>0} d_{ij} |u_j - u_i|,$$
$$\dot{P}_i = \sum_{j\neq i, m_{ij}>0} d_{ij} \sqrt{(\dot{u}_j-\dot{u}_i)^2+\epsilon} \qquad \approx \sum_{j\neq i, m_{ij}>0} d_{ij} |\dot{u}_j - \dot{u}_i|,$$

where $\epsilon > 0$ is a small regularization parameter and $q, \dot{q} > 0$ can be increased to improve the accuracy of the AFC solution. In this work, we set $\epsilon = 10^{-8}$ and consider $q = 10$ as well as $\dot{q} = \frac{q}{\tau}$. Eventually, the semi-discrete problem (10) is discretized in time using the implicit Euler method to achieve a provably bound-preserving finite element scheme [15, Section 4.5.2].

Although the continuous problem under consideration is linear in u, the dependence of the correction factors on the unknown solution makes the resulting AFC residual highly nonlinear. To solve the time-blocked problem, we apply again the time-simultaneous (damped) Newton scheme. Here, $\omega \in (0,1]$ is the relaxation parameter in the Newton update $u^{(j+1)} = u^{(j)} - \omega v^{(j+1)}$ and is either chosen to be constant or is set to the largest $\omega \in \{2^{-k} | k \in \mathbb{N}_0\}$ so that the Armijo condition [18, eq. (3.4) with $c_1 = 10^{-4}$] is satisfied for the norm of the residual. The solution is accepted if the relative norm of the residual becomes smaller than 10^{-6} within a maximum of 500 nonlinear iterations. As an initial guess, we either consider the solution sequentially calculated using a single Newton iteration in each time step or the low order approximation produced by the AFC scheme for $q = \dot{q} = 0$. The former starting value is very accurate so that the time-simultaneous Newton scheme converges with a fixed number of iterations independently of the final time T if $\frac{\tau}{h} \leq 0.2$ (see Tab. 4). The same behavior can be observed for greater time increments if the solution update is damped, at least in some iterations. On the other hand, the low order solution is very diffusive and the use of a relaxation parameter smaller than 1 seems to be mandatory to achieve a global residual reduction which is independent of T (see Tab. 5).

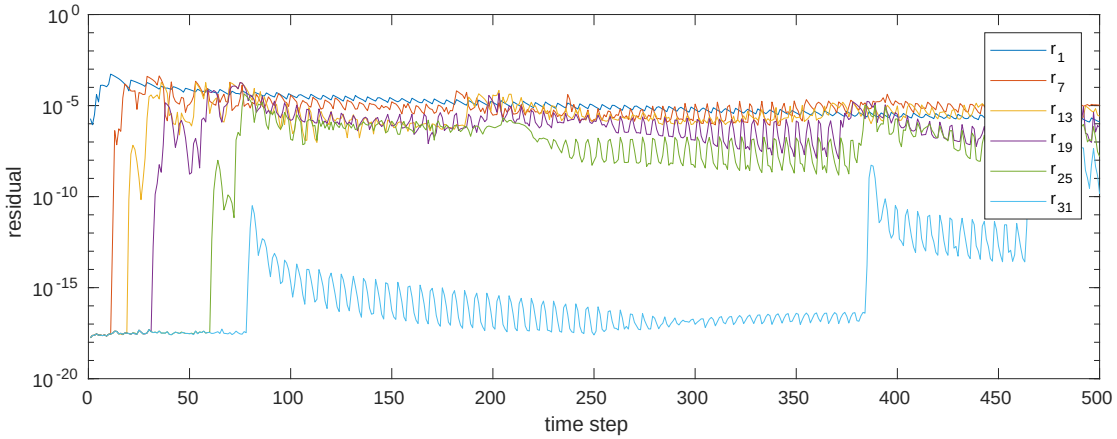
In Fig. 8, the behavior of the residual for the time-simultaneous Newton scheme using different damping parameters is illustrated for $h = 0.01$, $T = 1$, $\tau = 0.002$, and the low order initial guess. For the undamped Newton scheme, the residual decreases sequentially with respect to time and, hence, leads to a convergence behavior which depends on the final time T. If the damping parameter is smaller than one, the order of convergence is formally reduced, but the solution converges globally so that the total number of iterations seems to be independent of T. Therefore, the Armijo algorithm can be used to combine both benefits: During the first iterations, the residual of this time-simultaneous approach is large and a relaxation parameter smaller than 1 results in a global residual reduction. When the residual is sufficiently small, the relaxation parameter is adaptively increased and a quadratic convergence behavior can be observed.

T	τh^{-1}	$\omega = 1.0$			$\omega = 0.8$			$\omega = 0.5$			ω adapt.		
1	0.8	–	–	–	–	–	–	27	27	27	17	17	13
1	0.4	–	–	–	15	17	–	27	27	27	17	15	15
1	0.2	4	4	4	12	12	12	27	27	27	4	4	4
1	0.1	4	4	4	12	12	12	27	27	27	4	4	4
2	0.8	–	–	–	–	–	–	27	27	27	17	17	13
2	0.4	–	–	–	15	19	–	27	27	27	17	17	15
2	0.2	6	4	4	12	12	12	27	27	27	6	4	4
2	0.1	4	4	4	12	12	12	27	27	27	4	4	4
3	0.8	–	–	–	–	–	–	27	27	27	17	17	13
3	0.4	–	–	–	15	19	–	27	27	27	17	17	15
3	0.2	6	4	4	12	12	12	27	27	27	6	4	4
3	0.1	4	4	4	12	12	12	27	27	27	4	4	4
4	0.8	–	–	–	–	–	–	27	27	27	17	17	13
4	0.4	–	–	–	15	19	–	27	27	27	17	17	15
4	0.2	6	4	4	12	12	12	27	27	27	6	4	4
4	0.1	4	4	4	12	12	12	27	27	27	4	4	4

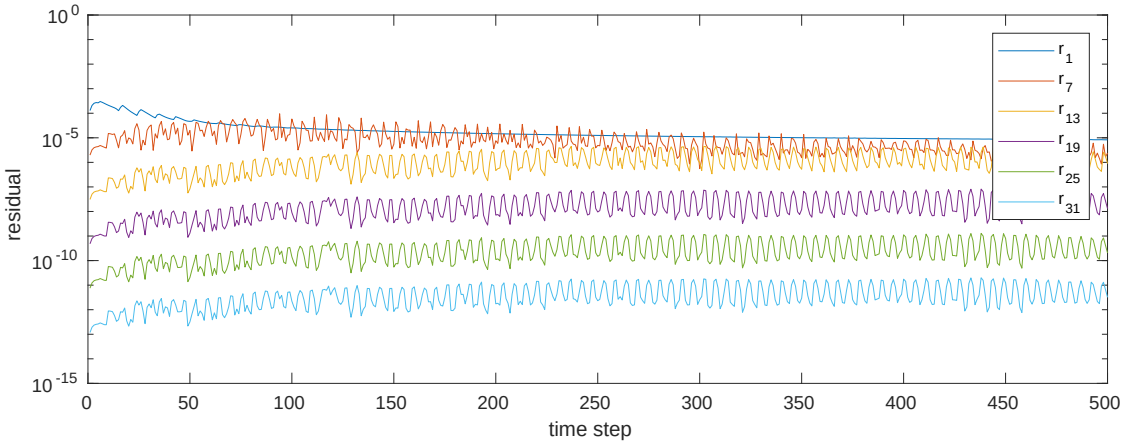
Table 4: Advection equation: Number of (damped) Newton iterations where the initial guess is given by the solution sequentially calculated using one Newton iteration in each time step (first column: $h = \frac{1}{50}$; second column: $h = \frac{1}{100}$; third column: $h = \frac{1}{200}$)

T	τh^{-1}	$\omega = 1.0$			$\omega = 0.8$			$\omega = 0.5$			ω adapt.		
1	0.8	–	–	–	–	–	–	28	28	28	14	14	17
1	0.4	–	–	–	17	–	–	30	31	31	13	13	13
1	0.2	21	36	78	18	19	21	32	35	36	12	14	29
1	0.1	15	19	88	17	20	23	32	36	40	12	13	20
2	0.8	–	–	–	–	–	–	28	28	28	14	14	17
2	0.4	–	–	–	17	–	–	30	31	31	16	13	14
2	0.2	21	53	135	18	19	31	32	35	38	12	15	30
2	0.1	19	27	105	17	20	30	33	39	47	13	15	24
3	0.8	–	–	–	–	–	–	28	28	28	15	14	17
3	0.4	–	–	–	17	–	–	30	31	32	16	13	19
3	0.2	21	104	222	18	23	55	32	35	55	12	16	24
3	0.1	21	44	126	17	22	63	45	83	77	14	19	40
4	0.8	–	–	–	–	–	–	28	28	28	15	14	17
4	0.4	–	–	–	17	–	–	30	31	35	16	13	23
4	0.2	21	201	299	18	26	77	32	35	87	12	16	30
4	0.1	30	63	196	19	43	83	51	96	161	14	18	37

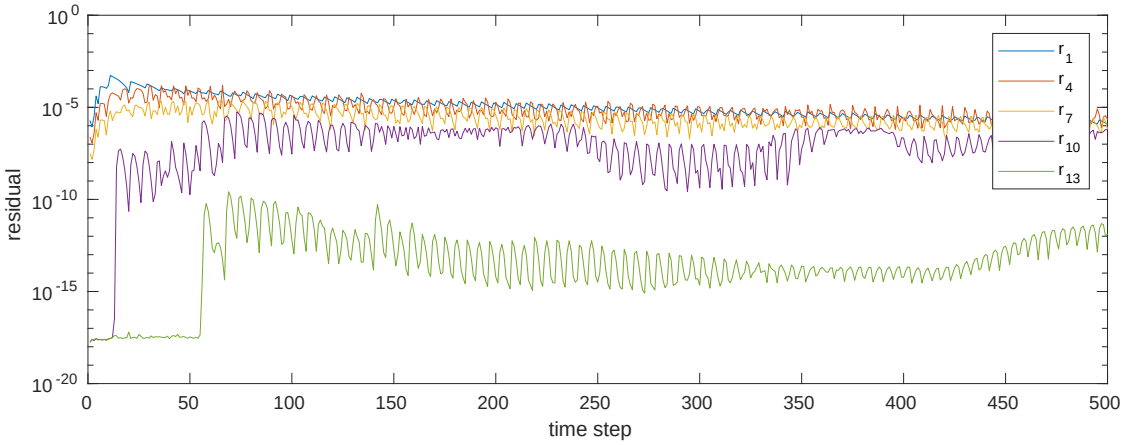
Table 5: Advection equation: Number of (damped) Newton iterations using the low order solution ($q = \dot{q} = 0$) as initial guess (first column: $h = \frac{1}{50}$; second column: $h = \frac{1}{100}$; third column: $h = \frac{1}{200}$)



(a) $\omega = 1.0$



(b) $\omega = 0.5$



(c) ω adaptively chosen

Figure 8: Advection equation: Discrete residual per time step in selected Newton iterations using the low order solution as initial guess, $h = 0.01$, $\tau = 0.002$, $T = 1$

4 Conclusion

We have presented an algebraically motivated approach for solving linear PDE problems leading to time-simultaneous geometric multigrid methods which are closely related to multigrid waveform relaxation techniques. In the shown numerical examples the proposed method yields a numerical convergence behavior that is robust w.r.t. the number of simultaneously treated time steps, the spatial mesh size and the time step size. In these cases, where the number of iterations does not grow significantly with increasing K, the computational cost is only slightly higher than in the corresponding sequential time-stepping case, but the time-simultaneous multigrid approach enhances the scalability of the spatial parallelization. Therefore, it is possible to use a higher degree of parallelism.

The application of this new approach to nonlinear nonstationary problems is also possible by using an adapted time-simultaneous Newton solver with suitable initial starting values and corresponding adaptive damping strategies while solving linearized subproblems, corresponding to the related Jacobian matrices, with the introduced time-simultaneous multigrid techniques in each Newton step. While this paper mainly aims to introduce and explain the underlying basic principles of this approach and explains the corresponding algorithmic and computational aspects to exploit an improved scaling behavior of spatial parallelism, the presented numerical tests of prototypical character illustrate the potential towards real life problems in the context of nonstationary nonlinear PDE problems for long time horizons.

Acknowledgements Calculations have been carried out on the LiDO3 cluster at TU Dortmund University. The support by the LiDO3 team at the ITMC at TU Dortmund University is gratefully acknowledged.

References

[1] J. Dünnebacke, S. Turek, P. Zajac, and A. Sokolov. A time–simultaneous multigrid method for parabolic evolution equations. Technical report, Fakultät für Mathematik, TU Dortmund, December 2019. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 619.

[2] M. Emmett and M. Minion. Toward an efficient parallel in time method for partial differential equations. *Commun. Appl. Math. Comput. Sci.*, 7(1):105–132, 2012.

[3] R. Falgout, S. Friedhoff, T. Kolev, S. MacLachlan, and J. Schroder. Parallel time integration with multigrid. *SIAM J. Sci. Comput.*, 36(6):C635–C661, 2014.

[4] R. Falgout, T. Manteuffel, B. O'Neill, and J. Schroder. Multigrid reduction in time for nonlinear parabolic problems: A case study. *SIAM J. Sci. Comput.*, 39(5):S298–S322, 2017.

[5] M. Gander. 50 years of time parallel time integration. In T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, editors, *Multiple Shooting and Time Domain Decomposition Methods*, pages 69–113, Cham, 2015. Springer International Publishing.

[6] M. Gander and M. Neumüller. Analysis of a new space-time parallel multigrid algorithm for parabolic problems. *SIAM J. Sci. Comput.*, 38(4):A2173–A2208, 2016.

[7] W. Hackbusch. Parabolic Multi-Grid Methods. In R. Glowinski and J.-L. Lions, editors, *Computing Methods in Applied Sciences and Engineering, VI*, pages 189–197. North-Holland, 06 1984.

[8] A. Hessenthaler, B. Southworth, D. Nordsletten, O. Röhrle, R. Falgout, and J. Schroder. Multilevel convergence analysis of multigrid-reduction-in-time. *SIAM J. Sci. Comput.*, 42(2):A771–A796, 2020.

[9] C. Hofer, U. Langer, M. Neumüller, and R. Schneckenleitner. Parallel and robust preconditioning for space-time isogeometric analysis of parabolic evolution problems. *SIAM J. Sci. Comput.*, 41(3):A1793–A1821, 2019.

[10] G. Horton and S. Vandewalle. A space-time multigrid method for parabolic partial differential equations. *SIAM J. Sci. Comput.*, 16(4):848–864, 1995.

[11] J. Janssen and S. Vandewalle. Multigrid waveform relaxation on spatial finite element meshes: The discrete-time case. *SIAM J. Sci. Comput.*, 17(1):133–155, 01 1996.

[12] D. Kuzmin and S. Turek. High-resolution fem-tvd schemes based on a fully multidimensional flux limiter. *Journal of Computational Physics*, 198(1):131–158, 2004.

[13] J. Lions, Y. Maday, and G. Turinici. Résolution d'EDP par un schéma en temps «pararéel ». *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 332(7):661 – 668, 2001.

[14] C. Lohmann. On the solvability and iterative solution of algebraic flux correction problems for convection–reaction equations. Technical report, Fakultät für Mathematik, TU Dortmund, August 2019. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 612.

[15] C. Lohmann. *Physics-Compatible Finite Element Methods for Scalar and Tensorial Advection Problems*. Springer Fachmedien Wiesbaden, Wiesbaden, 2019.

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

[17] M. Neumüller and A. Thalhammer. Combining space-time multigrid techniques with multilevel monte carlo methods for sdes. In *Domain Decomposition Methods in Science and Engineering XXIV*, pages 493–501. 01 2018.

[18] J. Nocedal and S. Wright. *Numerical Optimization*. Springer New York, New York, NY, 2006.

[19] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7(3):856–869, 1986.

[20] R. Speck, D. Ruprecht, R. Krause, M. Emmett, M. Minion, M. Winkel, and P. Gibbon. A massively space-time parallel N-body solver. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 92:1–92:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[21] S. Ta'asan and H. Zhang. On the multigrid waveform relaxation method. *SIAM J. Sci. Comput.*, 16(5):1092–1104, 1995.

[22] H. van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13(2):631–644, 1992.

[23] J. van Lent and S. Vandewalle. Multigrid waveform relaxation for anisotropic partial differential equations. *Numer. Algorithms*, 32:361–380, 2002.

[24] S. Vandewalle and G. Horton. Fourier mode analysis of the multigrid waveform relaxation and time-parallel multigrid methods. *Computing*, 54:317–330, 1995.

[25] S. Vandewalle and R. Piessens. Numerical experiments with nonlinear multigrid waveform relaxation on a parallel processor. *Appl. Numer. Math.*, 8(2):149–161, 1991.

[26] S. Vandewalle and E. Van de Velde. Space-time concurrent multigrid waveform relaxation. *Ann. Numer. Math.*, 1(1-4):347–363, 1994.

[27] T. Weinzierl and T. Köppl. A geometric space-time multigrid algorithm for the heat equation. *Numer. Math. Theor. Meth. Appl.*, 5:110–130, 02 2012.