No. 650 March 2022

Numerical studies of a multigrid version of the parareal algorithm

L. Wambach, S. Turek

ISSN: 2190-1767

Numerical studies of a multigrid version of the parareal algorithm

Lydia Wambach and Stefan Turek*

Institut für Angewandte Mathematik, Technische Universität Dortmund, Vogelpothsweg 87, 44227 Dortmund, Germany

E-mail: lydia.wambach@math.tu-dortmund.de

Abstract

In this work, a parallel-in-time method is combined with a multigrid algorithm and further on with a spatial coarsening strategy. The most famous parallel-in-time method is the parareal algorithm. Depending on two different operators, it enables the parallelism of time-dependent problems. The operator with huge effort is carried out in parallel. But despite parallelization this can lead to long run times for long-term problems. Since the parareal algorithm has a two-level structure and the time-parallel multigrid methods are also widespread in the area of parallel time integration, we combine these approaches. We use the parareal algorithm as a smoothing operator in the basic framework of a geometrical multigrid method, where we apply a coarsening strategy in time. So we get a multigrid in time method which is strongly parallelizable. For partial differential equations we add an extra spatial coarsening strategy to our multigrid parareal version. All in all we get a method, which has a high parallel efficiency and converges fast due to the multigrid framework, which is shown in the numerical studies of this work. So we will get a highly accurate solution and can greatly reduce the parallel complexity, which is especially important for long-term problems with a limited number of processors.

1 Introduction

The presentation of the parareal algorithm was in 2001 by Lions, Maday and Turinici [8]. This new approach allows parallelization in time of ordinary or time-dependent partial differential equations. The parareal algorithm was developed to calculate initial value problems in real time by the parallelization in time direction. Therefore, it is called parareal, which is a combination of parallel and real-time.

The problem with the parallelization of ordinary and time-dependent partial differential equations and their numerical solution is the time dependence. Starting from one time point, the solution at the next time point is calculated. Accordingly, the solution of the previous time point must be given as a necessity in order to calculate the solution of the next time point. For this reason, initial value problems are approximated by numerical methods serially in time, whereas parallelization in space is already widespread. The decomposition of the space generates independent problems, which are suitable for parallelization.

The problem of parallelization in time was recognized early and we can divide these methods into four different types, which is shown in the work of Gander [3] where an detailed overview of already existing time-parallel algorithms is given. In the first group there is a variety of direct solution methods. We consider the other three types representing iterative procedures more detailed. The first type of iterative methods is called *Multiple Shooting Method* and was first examined by Nievergelt in 1964. The inaccurate initial value approximations result in more accurate approximations, which are calculated in parallel on the individual time steps. Therefore the time interval is decomposed. A combination of these approximations is then carried out by interpolation, which is done serially. This concept was used several times, always based on an iterative solution of a fixed point equation using the Newton method. The parareal algorithm by Lions et. al. [8] can be understood as a special modification of these methods, using the difference of two solutions on the macro grid to approximate the Jacobian matrix in the Newton method as shown in [4]. The *Domain Decomposition Methods in Space and Time* describe the second group of time-parallel algorithms. The concept behind this is to split the problem into individual spatial and time sub-problems, so that a sub-problem consists of a spatial and time sub-problem. These sub-problems can be calculated in parallel. This is followed by an iterative coupling of the approximations. A special modification is the *Waveform Relaxation* algorithm, which was initially developed for the simulation of integrated circuits. Its main feature is to use many processors in the space dimension. The fourth category, which is shortly discussed in this work, consists of the time-parallel multigrid methods, *Multigrid Methods in Space-Time.* These algorithms are not naturally parallel, but the execution of the individual components can be carried out in parallel at space-time intervals. There are a lot of further developments, however Hackbusch [7] was the first who published a *Multigrid only in Time* procedure in 1984. The best-known continuations include the PFASST algorithm and the MGRIT algorithm. The PFASST algorithm, *Parallel Full Approximation Scheme in Space-Time*, was published by Minion in 2012 [11] and is subject to a parallel *Deferred Correction Method* which works as a smoother in a fully approximated multigrid method in space-time dimension. In 2014 the MGRIT algorithm, *Multigrid Reduction in Time Method*, was developed by Falgout and Friedhoff et. al. [1], which creates multiple levels by different smoothing. There are also connections between the MGRIT algorithm and the parareal algorithm, since the parareal algorithm can be viewed as a two-level multigrid reduction method in time. There are several other interpretations of the parareal algorithm as a multigrid method, which is shown in detail in [6] by Gander, Kwok and Zhang in 2018. But the most important difference to our multigrid version is, that we do not view the parareal algorithm due to its level structure as a multilevel or multigrid method, rather we use it to get a parallel access in a geometrical multigrid in time method. To compare our variant with a similar one, we will examine a first approach of the *multigrid-in-time* algorithm, which has been published as a technical report in 2013 by Falgout and Friedhoff in their work [2] as the MGIT algorithm in more detail in the numerical studies in chapter 5.

This paper is organized as follows: In Section 2, the parareal algorithm, its main features and a matrix presentation are introduced. This is followed by our multigrid version of the parareal algorithm in Section 3. We show the geometrical framework of a multigrid algorithm and its combination with the parareal algorithm to get a parallel approach. In Section 4 we consider the heat equation and a spatial coarsening strategy in combination with the parareal algorithm and our multigrid parareal version. Section 5 is devoted to the numerical studies on an ordinary and a partial differential equation, which were implemented in Matlab. There we compare our multigrid version with another multigrid method of the parareal algorithm known from the literature.

2 Parareal algorithm

As a model problem we consider the following ODE system:

$$\frac{du}{dt} = au \quad \in \ (0,T), \ \ u(0) = u_0 \tag{1}$$

In order to parallelize differential equations in time, the parareal algorithm decomposes the time interval $[0, T_N]$ into N sub-intervals

$$T_0 < T_1 < \cdots < T_n = n \cdot \Delta T < T_{n+1} < \cdots < T_N.$$

Each processor is assigned to a time interval $I_n = [T_{n-1}, T_n]$, for $n = 1, \ldots, N$ with the number of required processors N. The global time step size between the global time points T_{n-1} and T_n is denoted by ΔT. Each of these N sub-intervals solves independently an initial value problem with the initial condition $U_{n-1}, n = 1, 2, \ldots, N$. Therefore, we need a macro and a micro time step size δT and δt, which belongs to the macro operator $\mathcal{G}$ and the micro operator $\mathcal{F}$. The micro and the macro time step sizes are defined such that $\delta t \ll \delta T \leq \Delta T$, like in Figure 1.

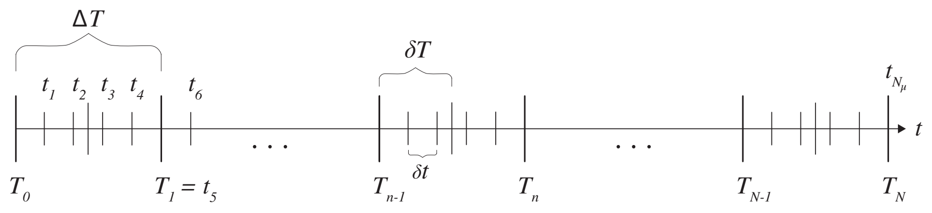


Figure 1: Decomposing of the time interval

The micro operator $\mathcal{F}$ with the time step size δt calculates with given initial values U_{n-1} at time point T_{n-1} the solution $\tilde{U}_n$ for the next time point T_n

$$\tilde{U}_n = u_n(T_n) = \mathcal{F}(U_{n-1}, T_n, T_{n-1}).$$

The micro operator $\mathcal{F}$ is due to the smaller time step size an accurate, but also more expensive solving method. To be more specific, it is a single step method which calculates an approximation on the next time point. This is done in parallel, because the micro operator is executed in the interval $[T_{n-1}, T_n]$ with the micro step size δt for $n = 1, \ldots, N$. The number of micro intervals per global interval $[T_{n-1}, T_n]$ is denoted as n_μ. It is calculated by the number of global intervals and the used micro time step size $n_\mu = \frac{\Delta T}{\delta t}$.

In order to obtain the initial values $U^0_{n-1}, n = 1, \dots, N$, for the N initial value problems, a fast but inaccurate single-step method is applied, the macro operator $\mathcal{G}(U_{n-1}, T_{n-1}, T_n)$ with the macro step size δT. It is executed serially over the entire interval $[0, T_N]$ to approximate solutions on the global time grid with grid points T_n for $n = 1, \dots, N$. The number of macro intervals per global interval is denoted by $n_\mathcal{M}$ and is calculated by $n_\mathcal{M} = \frac{\Delta T}{\delta T}$.

As soon as we have generated initial values for the N processors by executing the serial macro operator, the micro operator is executed in parallel on each sub-interval $[T_{n-1}, T_n]$. The difference between the micro solution and the macro solution $\delta^0_{n-1} = \mathcal{F}(U^0_{n-1}) - \mathcal{G}(U^0_{n-1})$ is calculated at the time point T_n to correct the old solution U^0_n. The correction for the time point T_n is carried out by reapplying the macro operator to the already corrected solution U^1_{n-1}. This happens sequentially and we get the new initial values $U^1_{n-1}, n = 1, \dots, N$ per processor.

The parareal algorithm can be understood as a predictor-corrector method by applying the macro operator to the already corrected solution U^1_{n-1} as a predictor. Calculating the difference δ^0_{n-1} from the application of the micro and macro operator to the solution U^0_{n-1} is understood as a corrector, see for more details Maday [9]. Accordingly, parareal proceeds iteratively and corrects the serially calculated inaccurate approximations of the macro operator through the more accurate micro approximations, which are calculated in parallel using an iteration formula for $k = 0, \dots, K$, where K is the maximum number of iterations and $n = 1, \dots, N$:

$$U^{k+1}_n = \underbrace{\mathcal{G}(U^{k+1}_{n-1}, T_{n-1}, T_n)}_{\hat{U}^{k+1}_n} + \underbrace{\mathcal{F}(U^k_{n-1}, T_{n-1}, T_n)}_{\tilde{U}^k_n} - \underbrace{\mathcal{G}(U^k_{n-1}, T_{n-1}, T_n)}_{U^k_n}, \tag{2}$$

$$U^0_n = \mathcal{G}(U^0_{n-1}, T_{n-1}, T_n), \text{ with } U^0_0 = u_0 \tag{3}$$

In Figure 2 one iteration of the parareal algorithm is shown, which runs over the complete time interval. The solution U^k_n is the solution on the macro time grid with the time step size

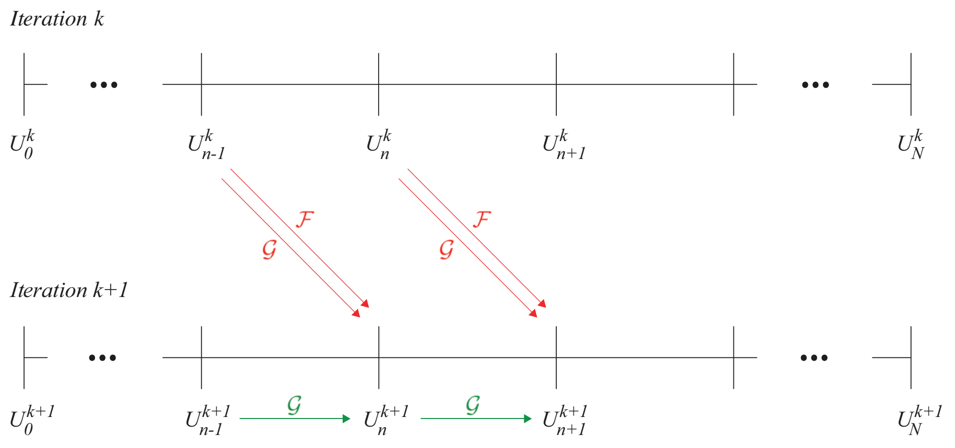


Figure 2: Iteration of the parareal algorithm

δT in iteration k. The color red shows the parallel execution per processor, while the color green stands for the serial execution. The serial green macro operator is executed in the correction step and is the predictor to the already corrected solution U_{n-1}^{k+1}. The red arrows represent the corrections $\delta_{n-1}^k = \mathcal{F}(U_{n-1}^k) - \mathcal{G}(U_{n-1}^k)$. The performance of the red macro operator can be disregarded, because it represents the execution of the macro operator per processor, which can be neglected in terms of effort. Thus, in this work we state that in each iteration k of the parareal algorithm, once the micro operator is executed in parallel and once the macro operator is executed serially in order to update the initial condition.

For the sake of simplicity the number of macro intervals per global interval is set to one so that the macro and global step sizes $\delta T = \Delta T$ coincide in this work.

2.1 Parareal in a matrix presentation

Regarding our following multigrid version, we present here the matrix form of the parareal algorithm. Maday and Turinici introduced it for the heat equation in their work [10]. A closer look can be found, for example, in the work [6] by Gander, Kwok and Zhang.

We consider the linear model problem in (1). We decompose the time interval into micro time points $t_0 < t_1 < \cdots < t_{\hat{n}_\mu} < t_{N_\mu} = T_N$, $\hat{n}_\mu = 1, \ldots, N_\mu = Nn_\mu$ with the micro step size $\delta t = t_{\hat{n}_\mu} - t_{\hat{n}_\mu - 1}$ where n_μ is the number of micro intervals per global and here also per macro grid interval. We denote with N_μ the total number of micro grid points. We get a bi-diagonal system

$$A_\mu u_\mu := \begin{pmatrix} 1 & & & \\ -\Phi_{\delta t} & 1 & & \\ & \ddots & \ddots & \\ & & -\Phi_{\delta t} & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N_\mu} \end{pmatrix} = \begin{pmatrix} u_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} =: b_\mu, \tag{4}$$

where $\Phi_{\delta t}$ is any one-step method with step size δt. The system of equations (4) thus corresponds to the serial solution of the problem (1) by the selected one-step method with a micro time step size δt. By eliminating unknowns, we can get a system of equations that only exists on the macro grid points $T_n = t_{n_\mu \cdot n}$. This is done by replacing the n_μ micro time steps by one macro step $\delta T := n_\mu \cdot \delta t$ and using the approximation $(\Phi_{\delta t})^{n_\mu} \approx \Phi_{\delta T}$. So we get the following preconditioned iteration

$$U^{k+1} = U^k + M_{\mathcal{M}}^{-1}(b_{\mathcal{M}} - A_{\mathcal{M}} U^k), \tag{5}$$

with the following matrices

$$A_{\mathcal{M}} := \begin{pmatrix} 1 & & & \\ -(\Phi_{\delta t})^{n_\mu} & 1 & & \\ & \ddots & \ddots & \\ & & -(\Phi_{\delta t})^{n_\mu} & 1 \end{pmatrix} \approx \begin{pmatrix} 1 & & & \\ -\Phi_{\delta T} & 1 & & \\ & \ddots & \ddots & \\ & & -\Phi_{\delta T} & 1 \end{pmatrix} =: M_{\mathcal{M}}. \tag{6}$$

Thus the matrix $A_{\mathcal{M}}$ presents the calculation of the micro solution at the macro time points, when doing the micro calculation serial and the matrix $M_{\mathcal{M}}$ presents the calculation of the macro solution at the macro time points. It is shown in [6] that this corresponds to the correction formula of the parareal algorithm. So if we solve the system $M_{\mathcal{M}}U^k = b_{\mathcal{M}}$, we get the same solution as if we execute the macro operator, $\mathcal{G}(v, T_{n-1}, T_n) := \Phi_{\delta T} v$. For the micro operator it is $\mathcal{F}(v, T_{n-1}, T_n) := (\Phi_{\delta t})^{n_\mu} v$, so the n-th line of the system $A_{\mathcal{M}}U^k = b_{\mathcal{M}}$ can be assigned to the micro operator with start point T_{n-1} and endpoint T_n.

Assuming U^k is known, it is obvious that the matrix-vector multiplications can be performed in parallel, while the other operations must be performed serially since they are calculated from the previous temporal solution.

2.2 Convergence

By calculating the iteration matrix $(I - M_{\mathcal{M}}{}^{-1}A_{\mathcal{M}})$ from equation (5) we recognize that it is nil-potent and therefore has a spectral radius of zero. This basically fulfills the property of a direct procedure which confirms that the parareal algorithm with solution U_n^k converges to the micro solution $\tilde{U}_n^k$ of the micro operator for $k \to N$ at the latest after N iterations. With $k = N$ we get the exact solution at the end time T_N, which we also obtain if we run the micro operator serially. Even if the parareal algorithm converges to machine accuracy, it cannot be more accurate than the serially calculated micro solution u_n. All this is explained in detail in Ruprecht [13] and Staff [14] where also a good overview of the convergence properties is given. The convergence for the model problem (1) was mainly examined in the literature. In the work of Gander and Vandewalle in [4] and [5] it was shown that the parareal algorithm converges superlinear if the following requirements are made: $T_N < \infty$, the macro operator $\mathcal{G}$ is a method with order p and the number of parareal iterations k is fixed. The error on the macro grid is known to be limited

$$\max_{n=0,1,\ldots,N} |u_n(T_n) - U_n^k| \leq C_k \Delta T^{p(k+1)},$$

with the micro serial solution on the macro grid points u_n and the parareal solution U_n^k. Additionally, it can be shown that the constant C_k is growing with k. The parareal algorithm changes the macro operator with order p to a scheme with order $p(k+1)$. To sum up, the macro operator is primarily responsible for the number of needed iterations whereas the micro operator just takes a small part in this. The micro operator is primarily responsible for the accuracy.

3 Variant of a multigrid parareal algorithm

In this chapter we consider our variant of a multigrid parareal algorithm. The main feature of the parareal algorithm is the use of a micro and macro grid. The application of different grids to solve differential equations reminds us of the classic multigrid method. In addition, Gander and Vandewalle have already shown in [4] that the parareal algorithm can be seen as a two-level multigrid method in time for a particular choice of smoothing, restriction and prolongation operator. Friedhoff et. al. [2] also used this approach and developed a multigrid variant by interpreting the parareal algorithm as a two-level reduction method and extended towards a multigrid process. This was done, among other things, by coarsening the micro and macro grids. This and further interpretations of the parareal algorithm as a multigrid method can be found in [6]. There, additionally to the MGRIT algorithm, which views the parareal algorithm as an algebraic multigrid method, the parareal as a geometric multigrid method and a multilevel parareal algorithm is shown. Our multigrid version of the parareal algorithm is not interpreted as a geometrical multigrid in time algorithm, rather we use a framework of a geometrical multigrid in time method which is combined with the parareal algorithm to achieve parallelism, which we now show in detail.

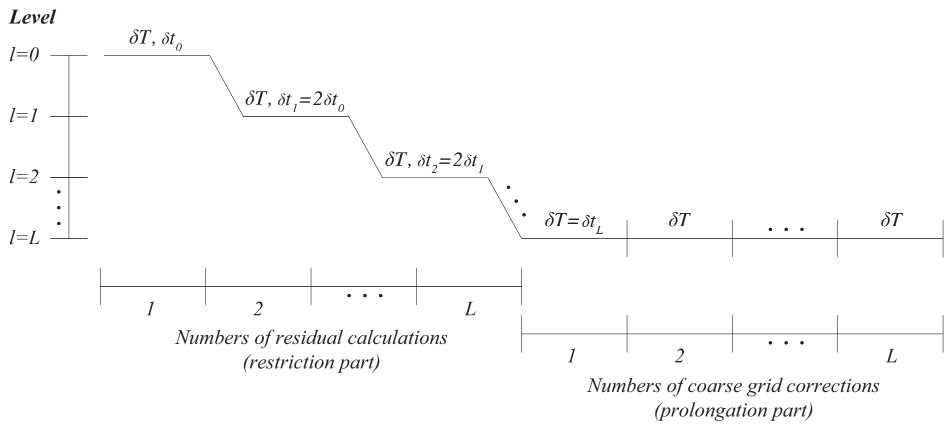
We consider the linear inhomogeneous initial value problem

$$u_t = f - au \quad \in \ (0, T) \text{ with } u(0) = 1, \tag{7}$$

which is discretized by a one-step method, so that we consider a system of equations of the form

$$A_{\mu,l} u_{\mu,l} = b_{\mu,l}, \tag{8}$$

which is defined on the micro grid, like presented in chapter 2.1 in equation (4). From now on our variables, which depend on the used coarsening level, obtain a level index l.



Level
l=0
l=1
l=2
l=L
δT, δt0
δT, δt1=2δt0
δT, δt2=2δt1
δT=δtL
δT
δT
1
2
L
Numbers of residual calculations (restriction part)
1
2
L
Numbers of coarse grid corrections (prolongation part)

Figure 3: Adjusted ‘half’ V-cycle of the MG-parareal algorithm

In our multigrid version, *the MG-parareal*, the original parareal algorithm was not seen as a two-level reduction method, but rather as a smoothing operator built into the framework of a V-cycle of the well-known multigrid method. To get a rough idea of the process, consider Figure 3. We apply a multigrid approach on the time grid, where we are only coarsening the micro time grid with step size δt_l. One V-cycle or here actually one ‘half’ V-cycle presents one iteration of the multigrid algorithm. Up to the coarsest multigrid level $L = log_2(n_\mu)$, where $\delta t_L = \delta T$, it is a classic V-cycle. We start on the finest micro grid with time step size δt_0 and go down to the coarsest level, here the macro grid with δT. This left part of the V-cycle calculations, which are especially the restriction part and the residual calculations, are executed on the micro grid with time points $t_{\hat{n}_{\mu,l}}$, $\hat{n}_{\mu,l} = 1, \ldots, N_{\mu,l}$, where the total number of micro grid points on the level is denoted with $N_{\mu,l}$. The macro grid remains the same in all levels and is just used in the parareal algorithm. It follows that the number of used processors on each level is also the same. The prolongation part of the V-cycle is not required in our algorithm in the classical form, since we calculate the solution on the coarsest grid with step size δT, so on our macro grid, due to the parareal algorithm. Thus, level L will not be changed. This is explained in more detail in the following paragraphs. We start with the smoothing operator and then describe the coarse grid correction.

Smoother Since we showed that the parareal algorithm is an iteration process, we assign i smoothing steps to i iterations of the parareal algorithm. In equation (2) it was shown that one iteration is built up of a micro calculation $\tilde{U}_n^{k-1} = \mathcal{F}(U_{n-1}^{k-1})$ in parallel and then a macro calculation $\hat{U}_n^k = \mathcal{G}(U_{n-1}^k)$ serially in the correction step. Due to the parareal correction step, solutions only exist on the macro grid and not on the micro grid. Before the first smoothing step, we need initial values. Thus, at the beginning of the MG-parareal algorithm, we use the known initialization by the macro operator like in equation (3). This leads to a multigrid method in time, whereby a time coarsening is done instead of a spatial coarsening.

Residual calculation To calculate the residual on the micro grid points, we have to compute for $n = 0, \dots, N-1$

$$r_{\mu,l} = b_{\mu,l} - A_{\mu,l} u_{\mu,l}^k. \tag{9}$$

We know the matrix $A_{\mu,l}$ and the right hand side $b_{\mu,l}$ from equation (8), which represents the serially execution of the micro operator. Further this is always solved on the actual micro grid and varies due to the multigrid approach. To achieve the parallel calculated micro grid solution $u_{\mu,l}^k$, the micro operator $\mathcal{F}(U_{n,l}^k)$ with the initial values $U_{n,l}^k$, which we get respectively from the smoothing step, is executed for $n = 0, \dots, N-1$. This can be described by solving the following system of equation

$$\tilde{A}_{\mu,l} u_{\mu,l}^k = \tilde{b}_{\mu,l} \quad \text{with } \tilde{A}_{\mu,l} \in \mathbb{R}^{(N_{\mu,l}+1)(N_{\mu,l}+1)} \text{ and } \tilde{b}_{\mu,l} \in \mathbb{R}^{(N_{\mu,l}+1)}.$$

The entry of the r-th row and the s-th column of the matrix $\tilde{A}_{\mu,l}$ is denoted with $\tilde{A}_{\mu,l}[r,s]$ and the r-th item of the right hand side with $\tilde{b}_{\mu,l}[r]$. To include the parallelism of the micro operator in the system, we add to the vector of the right hand side $b_{\mu,l}[r]$ one step with the micro operator on the macro grid solution U_{n-1}^{k-1}. In the serial case, the micro operator would be executed on the solution u_{n-1}^k which is done in the system $A_{\mu,l}[r,s] u_{\mu,l}^{k,[r]} = -\Phi_{\delta t} u_r^k = b_{\mu,l}[r]$. With $r, s = 1, \dots, n n_{\mu,l} + j, \dots, N_{\mu,l} + 1$ with $j = 0, \dots, n_{\mu,l} - 1$ for each global interval $n = 1, \dots, N-1$, it follows

$$\begin{aligned} \tilde{A}_{\mu,l}[r,s] &= A_{\mu,l}[r,s], & \tilde{b}_{\mu,l}[r] &= b_{\mu,l}[r], & &\text{with } r \neq n n_{\mu,l} + 2,\ s \neq n n_{\mu,l} + 1, \\ \tilde{A}_{\mu,l}[r,s] &= 0, & \tilde{b}_{\mu,l}[r] &= b_{\mu,l}[r] + \Phi_{\delta t}(U_{n-1}^{k-1}), & &\text{with } r = n n_{\mu,l} + 2,\ s = n n_{\mu,l} + 1. \end{aligned}$$

Accordingly, the residual is only different from zero at the first micro grid point after a

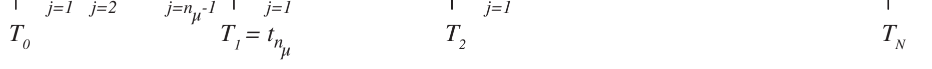


Figure 4: Micro grid points for the resuidual calculation

macro grid point $u_{n \cdot n_{\mu,l}+1}$, which is shown in Figure 4 by the red micro grid points. So we just have to determine the residual at the micro grid points $t_{n \cdot n_{\mu,l}+1}$ which correspond in our vectors to the entries in the $n \cdot n_{\mu,l} + 2$ row for $n = 0, \ldots, N-1$

$$\begin{aligned} r_{\mu,l}[nn_{\mu,l}+2] =& b_{\mu,l}[nn_{\mu,l}+2] - \\ & A_{\mu,l}[nn_{\mu,l}+2,\, nn_{\mu,l}+1 : nn_{\mu,l}+2](U_n^{k-1}, u_{\mu,l}^k[nn_{\mu,l}+2])^T. \end{aligned} \tag{10}$$

This calculation runs via the N processors, is independent of each other and can therefore be performed in parallel again. Thus, the residual corresponds to the error that arises from the parallel execution of the micro operator. If the micro operator would be executed serially over the entire time interval T, the residual would be zero everywhere.

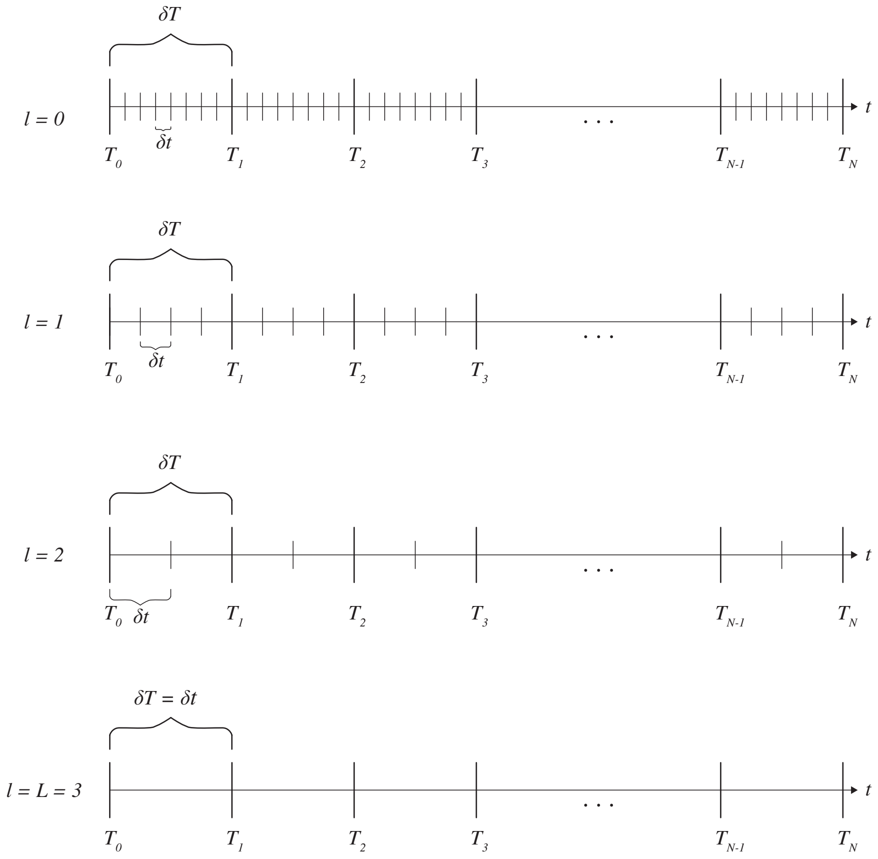


Figure 5: Coarsening strategy of the MG-parareal algorithm

Restriction As we already know from the multigrid method, the central element of the coarse grid correction is the relaxation on the coarser grid. We relax directly on the error in order to be able to eliminate the oscillating components. We carry out a micro grid coarsening until the micro grid corresponds to the macro grid, whereby the macro grid remains unchanged in our variant. So solving on the macro mesh is assumed to be exact on the lowest level. Let us consider Figure 5 with $L = 3$. We see that the macro grid with step size δT remains the same at every level l. The micro step size δt is doubled from level to level

$$\delta T_l = \delta T, \text{ and } \delta t_{l+1} = 2 \cdot \delta t_l, \ \forall\, l.$$

This coarsening strategy allows us to use a large difference between micro and macro grid and thus already at the starting level in order to keep the serial execution component low.

The calculation of the residual shown in equation (10) also applies to the auxiliary problems. We therefore do not need a restriction operator to transfer the residuals from one level to the next level as in the classic multigrid method, since every second micro grid point can be omitted. Thus $n_{\mu,l}$ denotes the number of fine intervals per global interval on level l which is here given by $n_{\mu,l+1} = \frac{n_{\mu,l}}{2}$. We only take the residuals $r_{\mu,l}[n \cdot n_\mu + 2]$ on the grid point $t^l_{nn_{\mu_l}+1}$on level l, so that the residuals $r_{\mu,l+1}[n \cdot n_{\mu,l+1} + 2]$ on the grid point $t^{l+1}_{nn_{\mu,l+1}+1}$ on level $l+1$ are given as

$$\begin{aligned} r_{\mu,l+1}[nn_{\mu,l+1} + 2] &= r_{\mu,l}[nn_{\mu,l} + 2] && \forall\, n, \\ r_{\mu,l+1}[nn_{\mu,l+1} + j] &= 0 && \text{for } j \neq 2, \ \forall\, n. \end{aligned} \tag{11}$$

A relaxation on a coarser grid can also be parallelized, since we consider $r_{\mu,l+1}$ per macro grid point T_n independently.

This is followed by the recursive call of our multigrid method for the next level $l+1$, which uses the previously relaxed residual $r_{\mu,l+1}$ as the right side of the system of equations

$$A_{\mu,l+1}\hat{e}_{\mu,l+1} = r_{\mu,l+1}. \tag{12}$$

The matrix $A_{\mu,l+1}$ results from the same type of approximation that was used in (6) for $A_{\mathcal{M}}$. Every second micro grid point of the matrix $A_{\mu,l}$ has now been omitted, so that we use a one-step method $\Phi^{l+1}_{\delta t2}$ with time step size $\delta t \cdot 2$ in matrix $A_{\mu,l+1}$, which is approximated by $\Phi^{l+1}_{\delta t2} \approx (\Phi^l_{\delta t})^2$. In the auxiliary problems we set $\underline{e}_{\mathcal{M},l+1} = \underline{e}_{\mathcal{M}} = 0$ for each processor as initial values on the fixed macro grid. Through the smoother or respectively when solving by the micro operator, we get the error solution $\hat{e}_{\mu,l+1}$ on the micro grid. This is done recursively until we have reached the coarsest micro grid, level L.

Solve the residual equation At the lowest level L, the residual equation (12) is then exactly solved by the macro operator, so it holds $\hat{e}_{\mu=\mathcal{M},L} = \mathcal{G}(\underline{e}^L_{\mu=\mathcal{M}})$. The solution only refers to the macro grid and is relaxed to the next finer micro grid.

Prolongation We do not need a classical prolongation because the correction of the approximation is done by $\tilde{e}_{\mathcal{M},l-1} = \tilde{e}_{\mathcal{M},l}$, where we approximate the error on the macro grid.

Correction of the approximation The solution $\tilde{U}^k_{\mathcal{M},l-1} = u^k_{\mu,l}(T_n), \ n = 0, \ldots, N$ is the micro solution on the macro grid from equation (9), on the level $l-1$, which is now corrected by the calculated error of the coarser micro grid level $l-1$ by the correction $U^k_{\mathcal{M},l-1} = \tilde{U}^k_{\mathcal{M},l-1} + \tilde{e}_{\mathcal{M},l-1}$. This continues until we reach the initial level 0. Starting from this new solution in the macro grid points $U^k_n = U^k_{\{\mathcal{M},l=0\},n}$, j post-smoothing steps are then carried out, which are again j steps of the parareal algorithm, with the new solution of the

MG-parareal algorithm U_n^k as initial values. To sum up, the pseudocode for the smoothing algorithm 1, which is identical to the iteration step of the original parareal algorithm, and the pseudocode of the MG-parareal algorithm 2 are presented.

Algorithm 1: Pseudocode of the smoothing operation

```
Function: Smoothing
Input: $U^0$                                                   // initial values
Output: $U^J$                               // solution of the parareal algorithm
for $j = 1$ to $i$ do                                       // smoothing iteration
    for $n = 1$ to $N$ do                                             // parallel
        $\tilde{U}_n^{j-1} \leftarrow \mathcal{F}(U_{n-1}^{j-1})$                   // micro operator
    end
    for $n = 1$ to $N$ do                                               // serial
        $\hat{U}_n^j \leftarrow \mathcal{G}(U_{n-1}^j)$                          // macro operator
        $U_n^j \leftarrow \hat{U}_n^j + \tilde{U}_n^{j-1} - U_n^{j-1}$                // correction
    end
end
```

Algorithm 2: Pseudocode of the MG-parareal algorithm

```
Function: MGparareal
Input: $U^0$                                                   // initial values
Output: $U^k$                            // solution of the MG-parareal algorithm
if $(k == 0 \,\&\, l == 0) \,||\, l == L$ then     // initialization and solving on level L
    for $n = 1$ to $N$ do                                               // serial
        $U_n^0 \leftarrow \mathcal{G}(U_{n-1}^0)$                              // iteration 0
    end
else
    $l = l + 1$                                               // increase the level
    for $k = 1$ to $K$ do                                 // MG-parareal iteration
        $U^{k-1} = Smoothing(U^{k-1})$                                 // smoothing
        for $n = 1$ to $N$ do                                         // parallel
            $u_n^k \leftarrow \mathcal{F}_{\delta t}(U_{n-1}^{k-1})$             // micro operator on micro grid
        end
        for $n = 1$ to $N$ do                                         // parallel
            $r(n) = \tilde{b}_n - \tilde{A}_n \cdot [U_{n-1}^{k-1}, u_n^k]$                    // residual
        end
        $\tilde{r} = r(1 : \frac{n_\mu}{2} : \frac{N_\mu}{2})$              // relaxation of a coarser grid
        $e = MGparareal(\tilde{r})$                // recursive request MG-parareal
        $U_n^k = u_n^k(T_n) + e$                                  // grid correction
        $U^k = Smoothing(U^k)$                                         // smoothing
        if $|U_n^k - U_n^{k-1}| < TOL \;\; \forall\, n$ then
            BREAK                                           // breaking condition
        end
    end
end
```

4 Spatial coarsening for partial differential equations

Here we show a variant of the parareal and MG-parareal algorithm, applied to time-dependent partial differential equations. The heat equation is considered and $\Omega \subset \mathbb{R}$ is a spatial domain with Dirichlet boundary conditions on $\partial\Omega$: Find $u(x,t) : \Omega \times I \to \mathbb{R}$ such that

$$\begin{aligned} \partial_t u - \Delta u &= f && \text{in} \quad \Omega \times (0,T), \\ u &= 0 && \text{on} \quad \partial\Omega \times I, \\ u(x,0) &= u_0(x) && \text{in} \quad x \in \Omega, \end{aligned} \tag{13}$$

with the initial condition u_0. By spatial discretizing using a standard difference scheme, we get at a system of ordinary differential equations

$$\dot{U}_h(t) + A_h U(t) = f_h(t), \qquad U_h(0) = U^0,$$

which we can solve using the well-known iteration formula of the parareal algorithm (2).

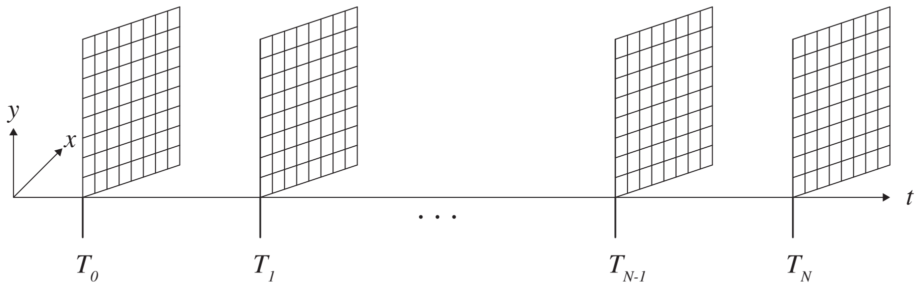


Figure 6: Time interval for the two-dimensional heat equation with a given spatial grid

In Figure 6 the decomposition of the time interval with a two-dimensional spatial mesh, which is presented instead of a one-dimensional spatial mesh for reasons of illustration, is shown. We solve on each macro grid point T_n a system depending on the spatial grid, which is pictured as a rectangle. For each time grid point all spatial grid points are considered.

4.1 Parareal with spatial coarsening

When considering partial differential equations, the parareal algorithm enables a coarsening strategy in the spatial domain. The convergence behavior of the parareal algorithm with a spatial coarsening strategy was published by Ruprecht in [12]. This coarsening in space is subsequently used in combination with the parareal algorithm, as shown in his work. It is done using different spatial grids for the micro and macro operator. The spatial step size of the macro operator is chosen to be larger than the spatial step size of the micro operator. The spatial grid of the micro operator is used for the solution U_n^k of the parareal algorithm. Therefore, we need spatial transfer operators to perform the parareal correction between the macro and micro solution. To transfer the macro grid solution, which now uses a coarser spatial grid, to the fine spatial grid, we use the interpolation operator P. In order to transfer the parareal solution U_n^k from the fine spatial grid to the coarse spatial grid, we rely on the restriction operator R. The correction formula of the parareal algorithm (2) thus changes to

$$\begin{aligned} U_n^{k+1} &= \mathbf{P}\mathcal{G}(\mathbf{R}U_{n-1}^{k+1}, T_{n-1}, T_n) + \mathcal{F}(U_{n-1}^k, T_{n-1}, T_n) - \mathbf{P}\mathcal{G}(\mathbf{R}U_{n-1}^k, T_{n-1}, T_n), \\ U_0^0 &= u_0, \qquad\qquad n = 1, \dots, N, \quad k = 0, \dots, K-1. \end{aligned} \tag{14}$$

The time grid used by the parareal algorithm remains unchanged. Only the coarse spatial step size has been changed. In addition, the solution of the parareal algorithm is approximated with coarsening in space on the fine spatial grid. In particular with regard to the complexity of the parareal algorithm, the coarsening of the space results in a shortening of the serially required execution time. This was also the basic idea of Ruprecht in [12].

Here this approach is called *SC-parareal*. For reasons of simplification, the coarse spatial step size h_c was taken as the double of the fine spatial step size h_f in the numerical studies

$$h_c = 2 \cdot h_f. \tag{15}$$

4.2 MG-parareal with spatial coarsening

Since we use the basic framework of a geometrical multigrid method, we can easily apply the coarsening strategy in time also in space. So in this section we consider the possibilities of spatial coarsening in the MG-parareal algorithm. Two variants are presented, whereby in the first approach two fixed spatial grids are used. In the second one the fine spatial grid is doubled in the individual levels. Due to the fact, that spatial coarsening only affects the spatial grid, all of our variants of MG-parareal algorithm with spatial coarsening have the same time coarsening strategy.

4.2.1 MG-SC-parareal

The first variant is called *MG-parareal with spatial coarsening* and is shortened by *MG-SC-parareal*. This approach results from the assignment of a finer spatial grid h_f to the micro operator, regardless of the micro time step size used and a coarser spatial grid h_c to the macro operator, which remains the same in the lower levels, when solving the auxiliary problems. For reasons of simplification, the coarsening strategy in time is graphically neglected and only the macro time grid with the grid points T_n is shown in Figure 7. The coarsening strategy in time remains unchanged in all of our multigrid variants and leads to the individual level l. The rectangles form the fine spatial grid and the red dots in the rectangle show the coarse spatial grid points. We see that the coarse and fine spatial grid remains the same in all auxiliary problems. So from now on whenever we run the macro operator, we use a coarse spatial grid and analogously a fine spatial grid for the micro operator. Consequently, as smoothing operator, we take the SC-parareal algorithm shown in equation (14).

The solution of the parareal algorithm U_n^k corresponds to the macro time points and the fine spatial points. The coarse grid correction in the multigrid framework is on the fine spatial grid. So the only difference between the MG-parareal and the MG-SC-parareal algorithm is the behavior on the lowest level L. There, the solution is calculated on the macro time points and coarse spatial grid points by the macro operator. This is followed by a single

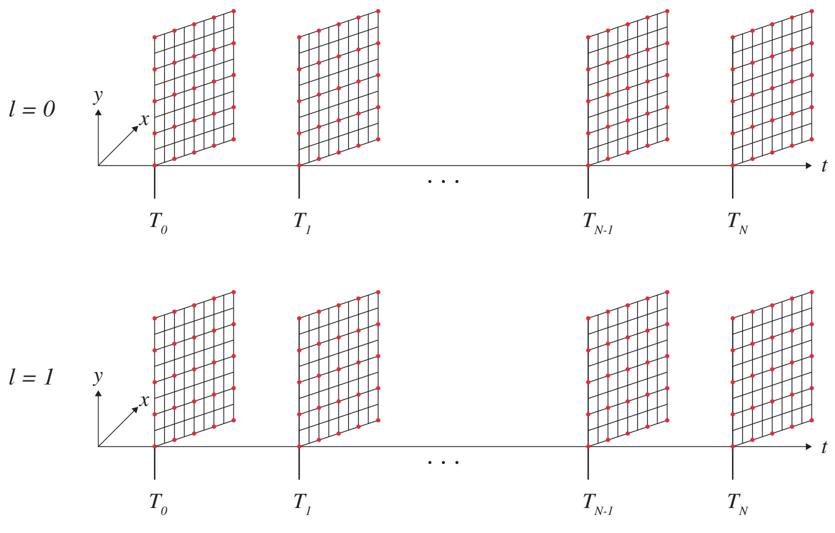


Figure 7: Spatial coarsening of MG-SC-parareal algorithm

prolongation to move from the coarse spatial grid to the fine spatial grid. The residual calculation, the relaxation on a coarser micro time grid and the coarse grid correction are always carried out on the fine spatial grid.

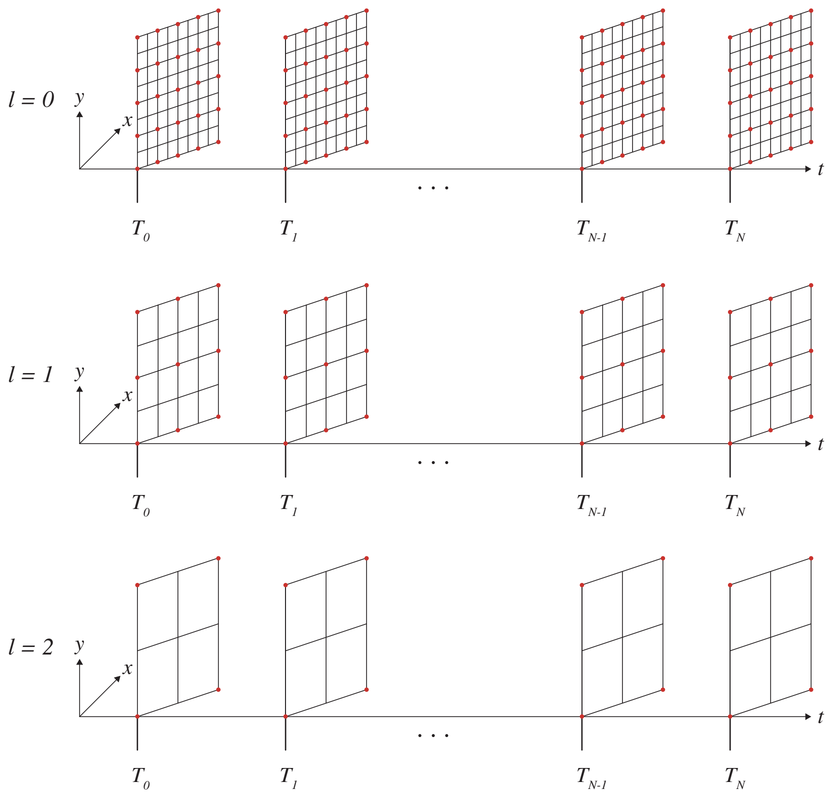


Figure 8: Spatial coarsening of the MG-LSC-parareal algorithm

4.2.2 MG-LSC-parareal

The second and more interesting variant is called *MG-parareal with spatial coarsening in level* and is shorted with *MG-LSC-parareal*. With this approach, the coarsening strategy of the micro time step size is also carried out for the spatial step sizes. Thus, we now use different and increasingly coarser spatial step sizes in the lower levels, when calculating the auxiliary problems. The fine and the coarse spatial step sizes are now doubled per level and with equation (15) it follows with an added level index l

$$h_{f,l+1} = 2 \cdot h_{f,l} = h_{c,l} \text{ and } h_{c,l+1} = 2 \cdot h_{f,l+1} = 2 \cdot h_{c,l}.$$

This is shown in Figure 8, where the red coarse spatial grid points on the finer level correspond to the fine grid points on the coarser level. These time and spatial coarsening strategies lead to the *MG-LSC-parareal* algorithm with the *SC-parareal* algorithm as a smoother and so two small adjustments of the MG-parareal algorithm arise. Since we coarsen the fine and coarse spatial grid from level to level, we also need restriction and prolongation operators when transferring residuals and errors between the different levels.

Starting from equation (10) and equation (11), where we calculate the residual and then restrict it to the next finer micro time grid, we now restrict this residual to the next coarser fine spatial grid and later on, the approximated error is prolongated to next finer fine spatial grid. Since we are only using the restriction and prolongation operators for the spatial coarsening, we can do this in parallel for each time interval.

The number of levels is still determined by the time step size and is therefore independent of the original spatial grid. We only coarsen the spatial grid until we either reach the macro time grid, level L, or the coarse spatial grid is discretized as a matrix of size one, so that a coarsening of the spatial grid is no longer possible and we are already using the largest possible spatial step size. In this case, the remaining level with the largest possible spatial step size is used in the macro operator and accordingly the half of it in the micro operator.

4.3 Numerical complexity

In this section we consider the computational complexity of the parareal algorithm and present a cost estimate for the one-dimensional heat equation. The key feature of the parareal algorithm is the parallelization so that the computing time is reduced. Without parallelization, it is clear that this iterative algorithm cannot be faster than the execution of the purely sequential micro operator. Therefore it has no application in a complete serial use.

We only consider the use of the macro and micro operator here, since the correction step can be disregarded in terms of effort. With a_G and a_F we denote the computational effort of executing the macro and micro operators per global interval and per spatial grid point. We denote the size of the fine spatial grid by m. Only the fine spatial step size is used in the original parareal algorithm. For the methods with coarsening of the spatial domain the cost of prolongation or restriction to move from one spatial grid to the other is added with $m \cdot a_T$ as the expense of transfer. The number of pre- and post-smoothing steps of the smoothing operator are called g. The effort for the residual calculation and relaxation per level is denoted by a_R.

Table 1: Comparison of numerical complexity

Method	Complexity
parareal	$mNa_G + mk(Na_G + a_F)$
SC-parareal	$\frac{m}{2}Na_G + mk(\frac{Na_G}{2} + a_F + 2a_T)$
MG-parareal	$mNa_G + mk((1 + Lg)Na_G + (g + 1)2a_F + 2a_R)$
MG-SC-parareal	$\frac{m}{2}Na_G + mk((1 + Lg)\frac{Na_G}{2} + 2(g + 1)a_F + 2a_R + 2L(1 + g)a_T)$
MG-LSC-parareal	$(\frac{m}{2} + k)Na_G + mk(gNa_G + \frac{4}{3}(g + 1)a_F + \frac{4}{3}a_R + 4(1 + g)a_T)$

We compare the methods in Table 1. For the parareal algorithm we get the effort Na_G and a_F for the macro and the micro operator over the complete time interval $[0, T]$. The number of intervals is disregarded for the micro operator, since it calculates in parallel. In the initialization step, the macro operator is executed once and then once in each iteration. The micro operator is executed once in each iteration, too.

Spatial coarsening leads to halving of the coarse effort, since the macro operator only uses $\frac{m}{2}$ grid points. We will see later in the numerical studies on the SC-parareal algorithm that the transfer costs are low compared to the serial costs of the macro operator due to parallelization of restriction and prolongation. This time coarsening strategy leads particular with a large serial effort to a reduction in the computing time. The serial execution time can be reduced by a maximum of half in comparison to the original parareal algorithm.

In the MG-parareal algorithm, the macro operator is carried out in the correction step per level and iteration and requires an effort of $kgLNa_G$. Due to the coarsening strategy of doubling the micro grid per level, the effort of the micro operator and residual calculation per level is halved

$$\sum_{i=0}^{L-1} \frac{a_F}{2^i} \approx 2a_F, \qquad \sum_{i=0}^{L-1} \frac{a_R}{2^i} \approx 2a_R.$$

In the coarse grid correction, the micro operator per level and iteration is performed again and the effort is halved. At the lowest level, the macro operator is executed per iteration which has an effort of kNa_G. Because of the term $kNa_G(1 + Lg)$, it can be assumed that the MG-parareal approach has a significantly higher effort than the parareal algorithm per iteration. But the MG-parareal algorithm has a smaller number of iterations until convergence because of its multigrid structure, since it has better convergence properties. We will see in the numerical investigations that the MG-parareal approach shows advantages especially for long-term problems and problems with very small micro step sizes. But the serial part is very large due to the macro operator in the smoothing steps in the levels. The spatial coarsening gives us the opportunity to solve exactly this problem.

To get an estimation of complexity for the MG-SC-parareal, we just halve the effort of the coarse operator and add the transfer cost per level and smoothing step. If we now additionally apply the spatial coarsening strategy in connection with the MG-parareal algorithm, we can see for the MG-LSC-parareal algorithm that the complexity of the micro operator and of the residual calculation are no longer multiplied by a factor of 2, due to reduction of spatial size per level and smoothing step

$$\sum_{i=0}^{L-1} \frac{a_F}{2^i}\frac{m}{2^i} \approx \frac{4}{3} m a_F, \qquad \sum_{i=0}^{L-1} \frac{a_R}{2^i}\frac{m}{2^i} \approx \frac{4}{3} m a_R.$$

Similarly, the transfer and macro costs are halved per level and iteration, so that we can approximate it with a factor of 2 and now it is independent of the number of levels. Furthermore on the level L it holds that $m = 1$. We significantly reduced the serial execution time. In addition, we were also able to reduce all other costs. After presenting and analyzing several multigrid variants, we now consider them in detail.

5 Numerical studies

In this work the presented methods are analyzed for an ordinary differential equation and a partial differential equation. In the last section, a comparison is made between our multigrid variants of the parareal algorithm and a multigrid variant from the work of Friedhoff [2].

Our methods are firstly investigated with the iteration error

$$\max_{\hat{n}_\mu}(||u^k_{\hat{n}_\mu} - u_{\hat{n}_\mu}||_{L^2(\Omega)}),$$

where $u^k_{\hat{n}_\mu}$ denotes the micro parareal solution on the micro grid points for the iteration k and $u_{\hat{n}_\mu}$ is the discrete micro solution, which results from a purely serial execution of the micro operator. We define the error with the analytic solution $u(t_{\hat{n}_\mu})$ on the micro grid points and the the defect as in equation (9)

$$\max_{\hat{n}_\mu}(||u^k_{\hat{n}_\mu} - u(t_{\hat{n}_\mu})||_{L^2(\Omega)}) \quad \text{and} \quad \max_{\hat{n}_\mu}(||b - A \cdot u^k_{\hat{n}_\mu}||_{L^2(\Omega)}).$$

We test our multigrid versions for different configurations of number of processors and this results in different macro step sizes, which coincides with the global time step size.

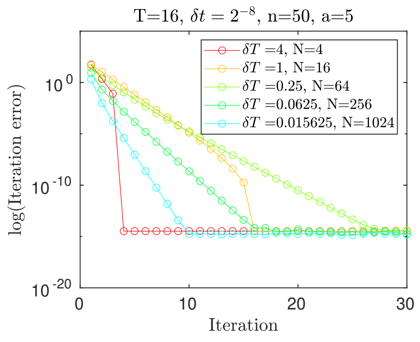
5.1 ODE

We consider the following ordinary differential equation (7)

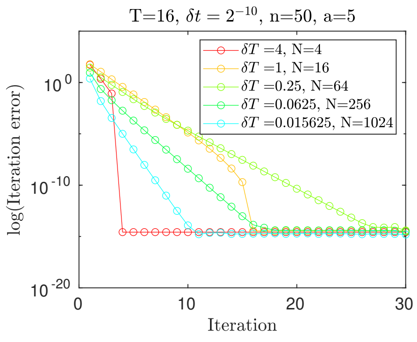
$$u_t = f - au = -ae^{(-at)} + \frac{n\pi(t+T)cos(n\pi t) + sin(n\pi t)}{T} \quad \in (0, T] \text{ with } u(0) = 1,$$

which is discretized by the backward Euler method in the time interval $(0, T]$. The analytic solution is given by $u(t) = sin(n\pi t)(1 + t/T) + exp(-at)$. We fix the parameters as $n = 50$ and $a = 5$. Furthermore the stopping criteria are removed in the following figures to analyze the convergence and stagnation behavior better.

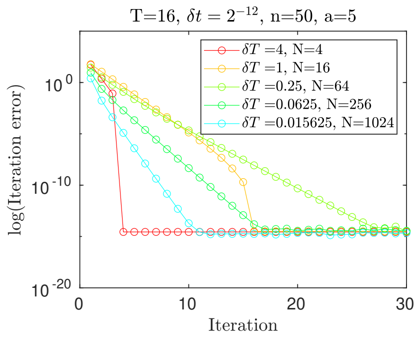
Parareal We start with a different number of processors $N = 4, 16, 64, 256, 1024$ and thus for different macro step sizes with a fixed end time of $T = 16$. In order to investigate the dependence of the convergence behavior with respect to the macro and micro step size, the iteration error, the error and the defect of the parareal algorithm for different micro time step sizes per column are shown in Figure 9. In a) we see for a number of processors of 4 and 16 that the parareal algorithm converges after 4 and 16 iterations corresponding due to the fact that parareal converges after N iterations at the latest. For a larger number of processors and the resulting smaller macro step size, the number of iterations decreases. If we look at the whole line, we see that the convergence behavior does not change for different micro step sizes. The iteration error and the defect are in the size range 10^{-15} for all macro



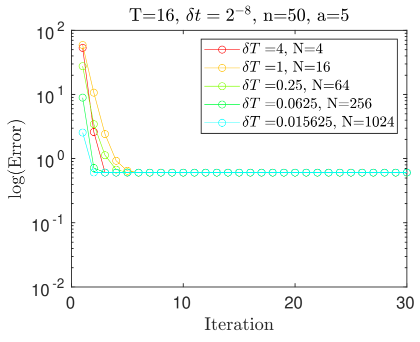
(a) Iteration error



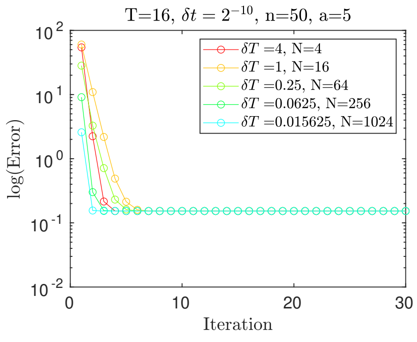
(b) Iteration error



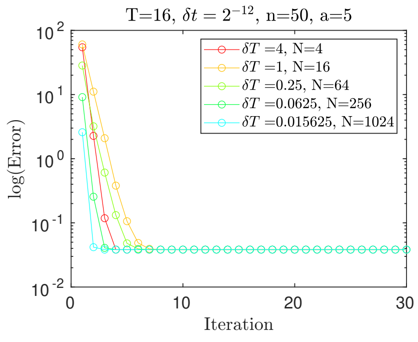
(c) Iteration error



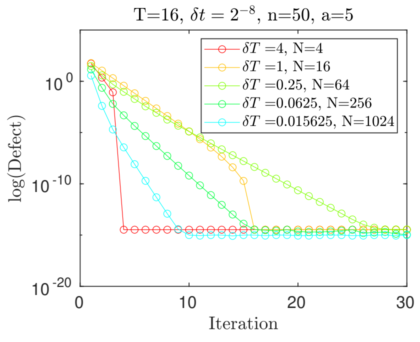
(d) Error



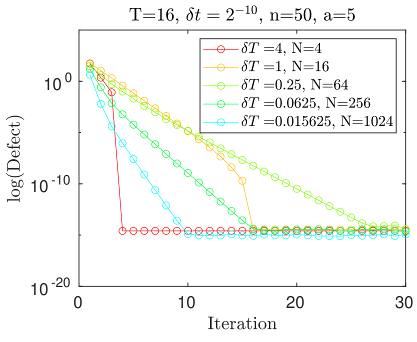
(e) Error



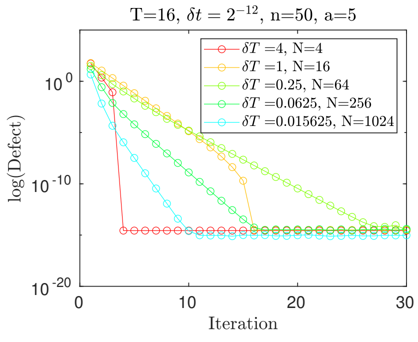
(f) Error



(g) Defect



(h) Defect

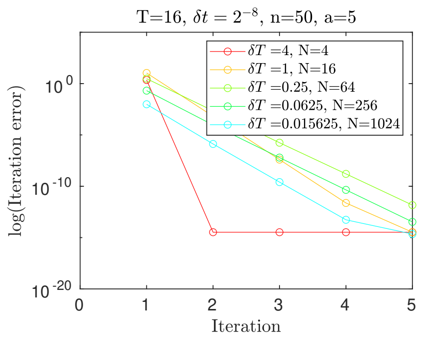


(i) Defect

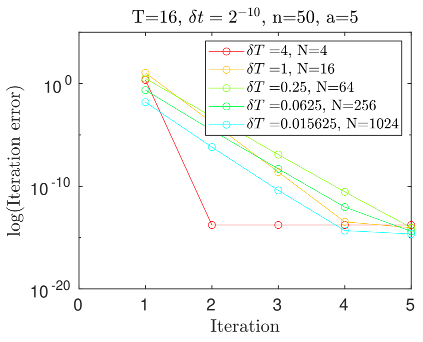
Figure 9: Iteration error, error and defect of the parareal algorithm for $\delta t = 2^{-8}, 2^{-10}, 2^{-12}$ per column with $T = 16$.

step sizes. If we illustrate the second line, we see that the smaller the micro step size, the smaller the error. The error does not change for smaller macro step sizes. The defect is shown in line three and it has an analogous behavior like the iteration error.

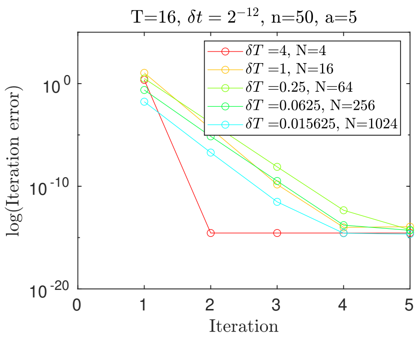
MG-parareal We consider the MG-parareal algorithm in Figure 10. One pre-smoothing step and no post-smoothing is used, because as we will see, the number of iterations required is already relatively small. The number of iterations required decreases as the number of processors increases. If we now consider the complete line and thus the behavior for smaller micro step sizes, we see that the number of iterations is reduced. This is justified by the number of levels used. The smaller the micro step size is, the higher the number of levels



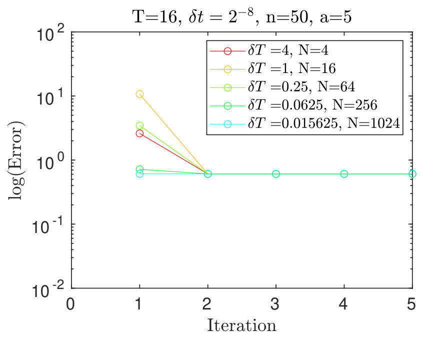
(a) Iteration error



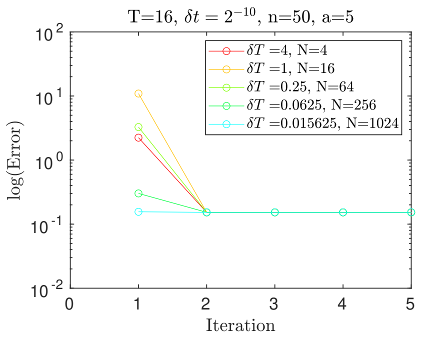
(b) Iteration error



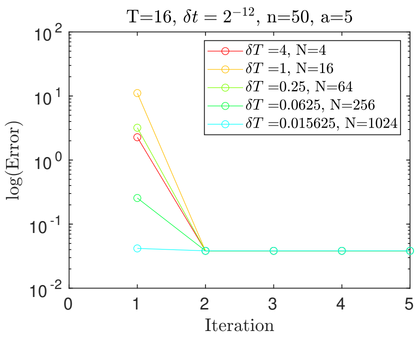
(c) Iteration error



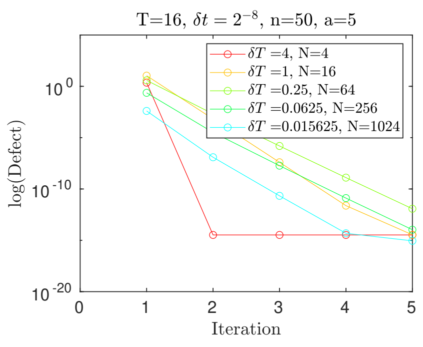
(d) Error



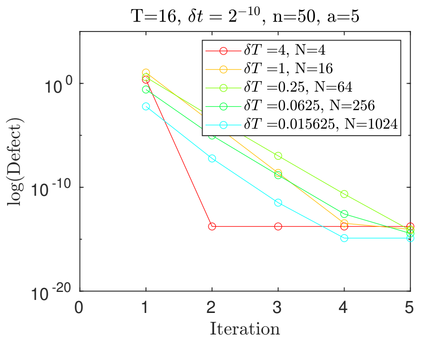
(e) Error



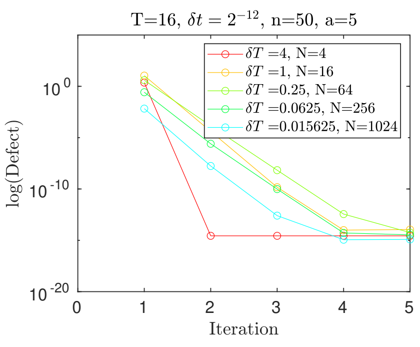
(f) Error



(g) Defect



(h) Defect



(i) Defect

Figure 10: Iteration error, error and defect of the MG-parareal algorithm for $\delta t = 2^{-8}, 2^{-10}, 2^{-12}$ per column with $T = 16$.

in which the residual is calculated, compare Table 2. The smaller the micro step size is, the smaller is the error too. Also here the error is independent of the macro step size.

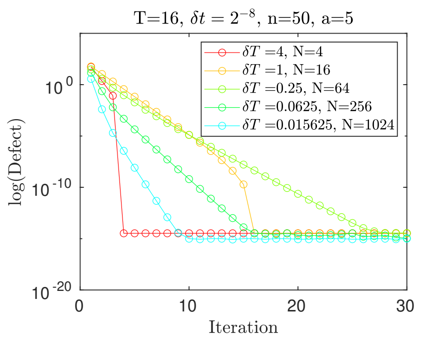
Table 2: Number of levels for $T = 16$

N \ δt	2^{-8}	2^{-10}	2^{-12}
4	10	12	14
16	8	10	12
64	6	8	10
256	4	6	8
1024	2	4	6

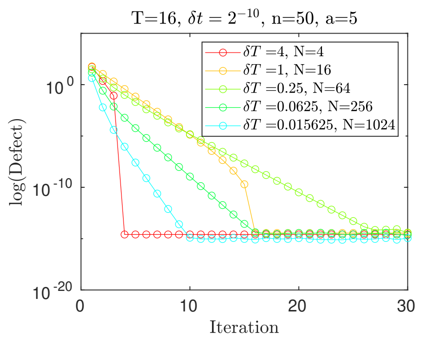
Parareal vs. MG-parareal In order to investigate the convergence behavior depending on the end time T, we now compare the defects of the parareal and MG-parareal algorithm for different end times $T = 16$ and $T = 64$ in Figure 11. Since the end time only has an effect on the macro step size and not on the micro step size, the error is not considered. We see in a) and d) for a large number of processors $N = 64, 256, 1024$ that with the same micro step size, the number of iterations increases strongly for a larger end time and thus for a larger macro step size. The MG-parareal algorithm in g) and j) has the same behavior, but the number of iterations increases only slightly. Accordingly, the convergence behavior depends on the end time. The second difference is that the behavior of the parareal algorithm remains the same for smaller micro step sizes, whereas the MG-parareal algorithm converges faster, because of the larger number of levels. Let us now consider a comparison of the number of parareal iterations. An iteration of the MG-parareal algorithm corresponds to two iterations of the parareal algorithm due to the number of pre-smoothing steps and the one execution of the micro operator based on the residual calculation. By illustrating the small macro step sizes with $N = 64, 256, 1024$ in the last column with the smallest step size, the parareal algorithm requires approximately 25, 27 and 15 iterations and the MG-parareal 4, 5 and 4, which are 8, 10 and 8 parareal iterations. The number of required iterations for the MG-parareal is significantly lower.

Numerical complexity We consider the computing time of the methods. The stopping tolerance was set to 10^{-9} and the CPU time is given in seconds. The required time for the macro operator is shown with τ_G and the total time with τ. Since the micro operator and the residual calculation can be executed in parallel the execution times τ_F and τ_R represent the required CPU time per processor. The number of required iterations is called k.

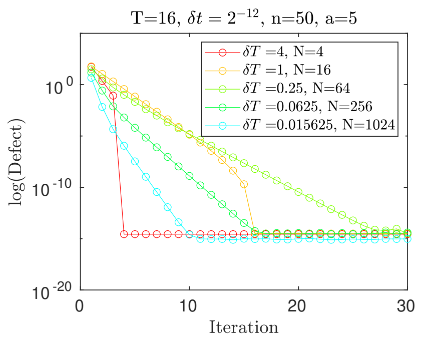
We start with a configuration with a small end time $T = 16$ and a relatively large micro step size $\delta t = 2^{-12}$ in Table 3. The higher the number of processors is, the lower is the CPU time of the micro operator and the higher is the macro time. The total execution time depends mostly on the macro operator for a larger number of processors, so that the total execution time of the MG-parareal is slightly longer than of the parareal algorithm. The MG-parareal algorithm runs through fewer iterations, but the use of the macro and micro operator in the individual levels and so in the auxiliary problems leads to an increase in effort. The computing time of the residual is very short, which is guaranteed by the parallelism.



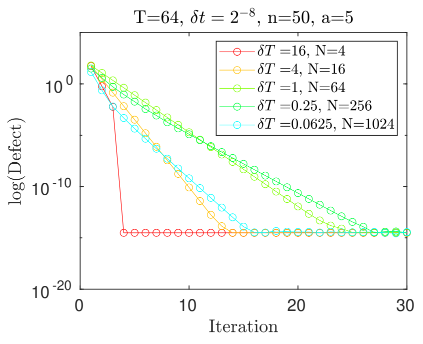
(a) Parareal $T = 16$



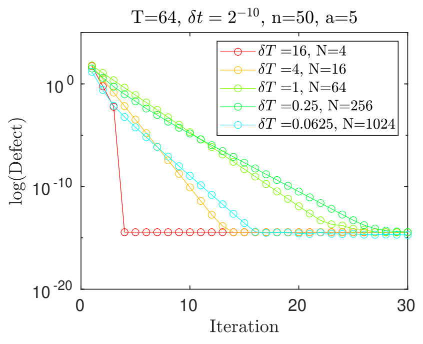
(b) Parareal $T = 16$



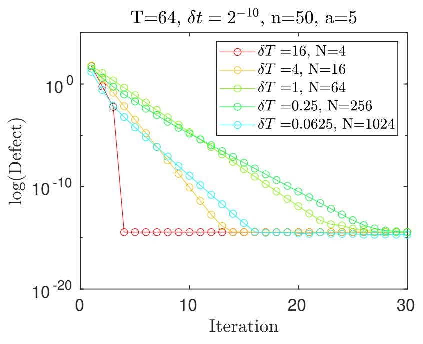
(c) Parareal $T = 16$



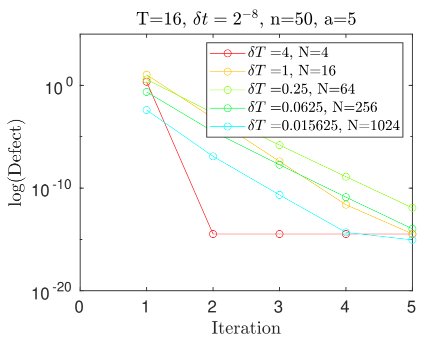
(d) Parareal $T = 64$



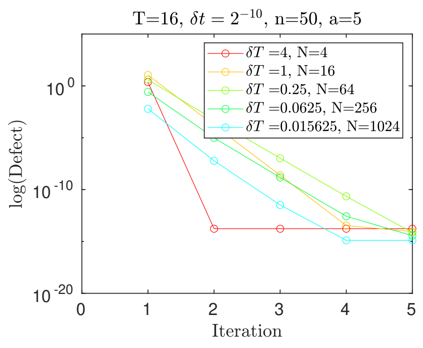
(e) Parareal $T = 64$



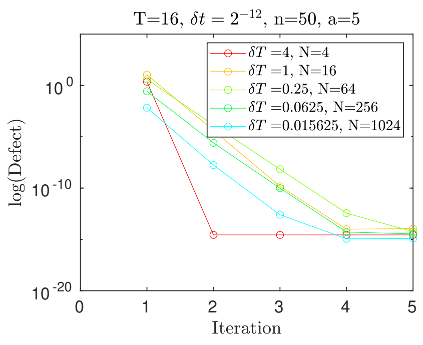
(f) Parareal $T = 64$



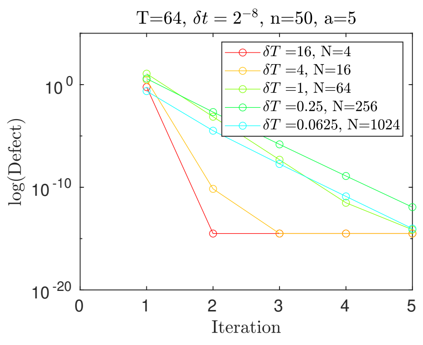
(g) MG-parareal $T = 16$



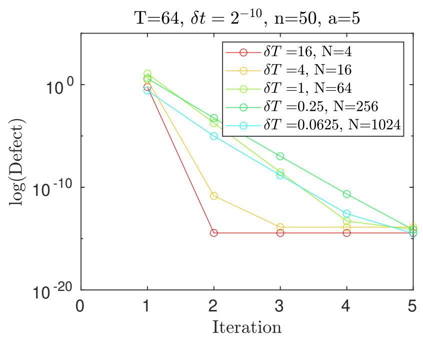
(h) MG-parareal $T = 16$



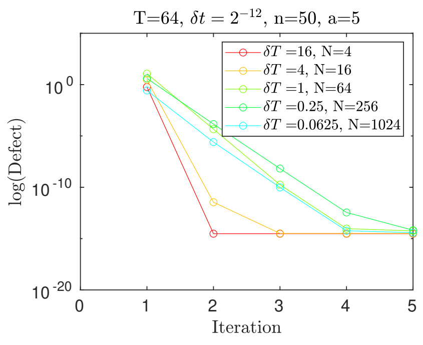
(i) MG-parareal $T = 16$



(j) MG-parareal $T = 64$



(k) MG-parareal $T = 64$



(l) MG-parareal $T = 64$

Figure 11: Defect of the parareal and MG-parareal algorithm for $\delta t = 2^{-8}, 2^{-10}, 2^{-12}$ per column and per line for $T = 16, 64$.

Table 3: CPU-time in seconds for $T = 16$ and $\delta t = 2^{-12}$

	N	4	16	64	256	1024
Parareal	k	5	15	18	11	7
	τ_G	0.0300	0.0060	0.0126	0.0206	0.0516
	τ_F	0.0717	0.0597	0.0159	0.0024	0.0005
	τ	0.1018	0.0657	0.0285	0.0230	0.0520
MG-parareal	k	2	3	4	4	3
	τ_G	0.0047	0.0081	0.0158	0.0208	0.0517
	τ_F	0.0724	0.0426	0.0130	0.0043	0.0059
	τ_R	0.0210	0.0020	0.0026	0.0019	0.0027
	τ	0.0981	0.0527	0.0314	0.0269	0.0603

Table 4: CPU-time in seconds with $T = 256$ and $\delta t = 2^{-18}$

	N	4	16	64	256	1024
Parareal	k	5	7	10	17	18
	τ_G	0.0158	0.0052	0.0125	0.0332	0.1272
	τ_F	81.4292	28.2369	10.2307	4.0663	1.0413
	τ	81.4450	28.2421	10.2432	4.0996	1.1685
MG-parareal	k	2	2	2	3	3
	τ_G	0.0037	0.0074	0.0109	0.0380	0.1465
	τ_F	68.5853	15.1890	4.0014	1.5279	0.3603
	τ_R	0.0128	0.0082	0.0113	0.0088	0.0108
	τ	68.6019	15.2047	4.0236	1.5747	0.5176

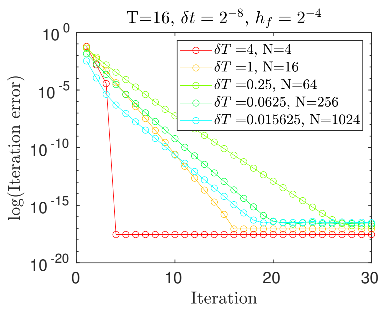
This configuration does not use the main advantage of the parareal algorithm, since such problems can easily be performed without parallelization. Accordingly, configurations of long-term simulations with very small micro step sizes, are more interesting. In addition, we saw in the previous numerical tests that the parareal algorithm produces worse results for a large end time and does not produce any change for small micro step sizes, in contrast to the MG-parareal algorithm, which gets better. We now consider in Table 4 an outrider of long-term simulation with a very small micro step size $\delta t = 2^{-18}$ and an end time of $T = 256$. Here the total execution time depends on the micro run time, which is for the MG-parareal algorithm approximately half the size of the parareal algorithm. The total run time is also significantly shorter.

5.2 Heat equation

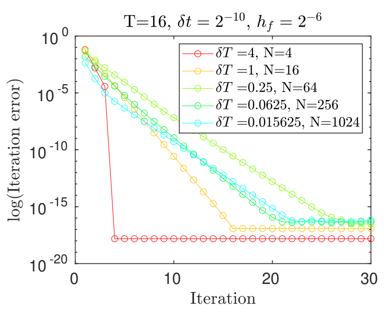
We consider the heat equation (13) in one dimension for a domain $\Omega \subset \mathbb{R}$ with Dirichlet boundary condition $\partial\Omega$. Find $u(x,t) : \Omega \times (0,T] \to \mathbb{R}$ such that

$$\partial_t u - \Delta u = f = -2(6x^2 - 6x + 1)sin(n\pi t) + \pi n cos(n\pi t)x^2(1-x)^2 \quad \text{in} \quad \Omega \times (0,T],$$
$$\text{with } u = 0 \text{ on } \partial\Omega \times I \text{ and } u(x,0) = u_0(x) \text{ in } x \in \Omega = [0,1],$$

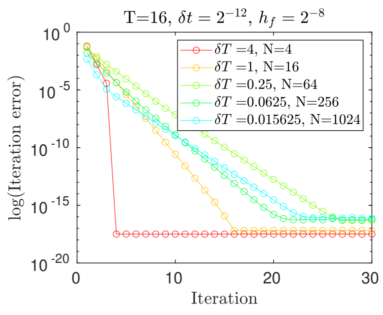
with $u(x,t) = x^2(1-x)^2 sin(n\pi t)$. We set $n = 5$. Finite Differences were used for the spatial discretization.



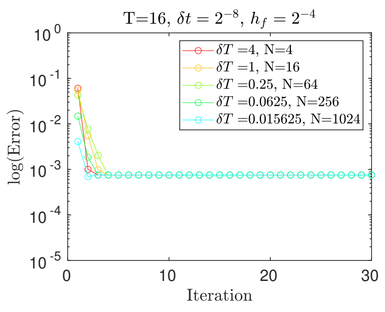
(a) Iteration error



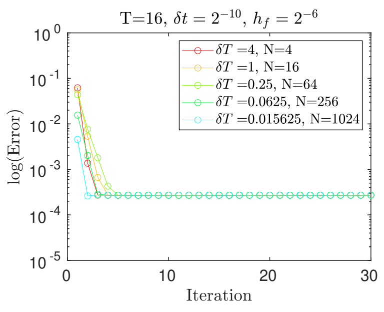
(b) Iteration error



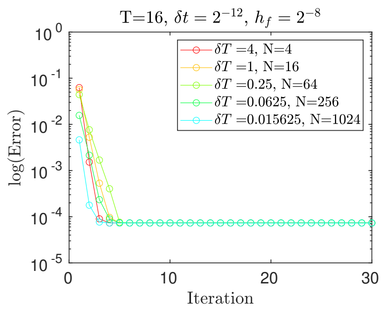
(c) Iteration error



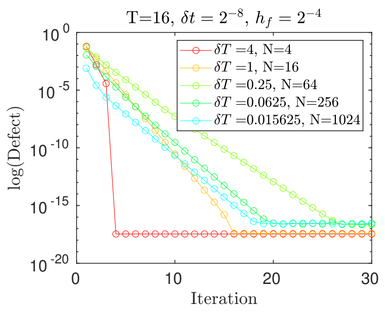
(d) Error



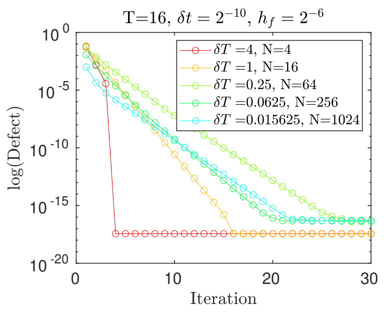
(e) Error



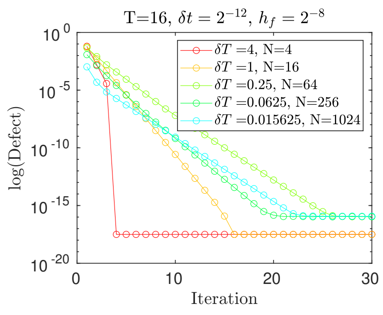
(f) Error



(g) Defect



(h) Defect

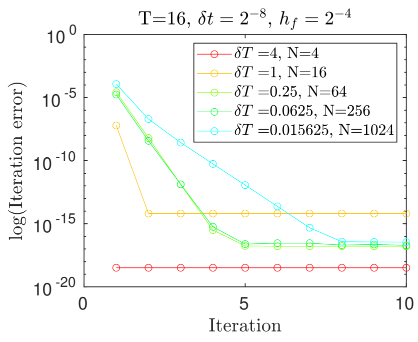


(i) Defect

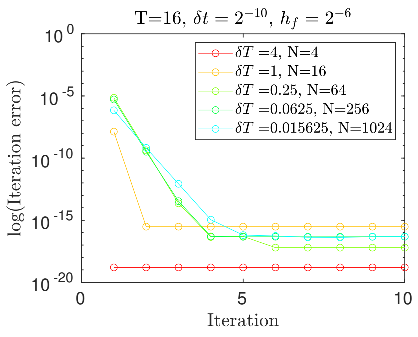
Figure 12: Iteration error, error and defect of the parareal algorithm for $\delta t = 2^{-8}, 2^{-10}, 2^{-12}$ and $h_f = 2^{-4}, 2^{-6}, 2^{-8}$ per column with $T = 16$.

Parareal Firstly we study the iteration error, the error and the defect in Figure 12 for an end time of $T = 16$. Different micro step sizes and spatial step sizes are shown per column. In line one and three, the number of required iterations decreases for smaller macro steps and thus for a larger number of processors. Further on small macro step sizes, like for $N = 1024$, require more iterations, the smaller the micro step size is. The error does not change for different macro step sizes, but decreases for smaller micro step sizes and spatial step sizes.

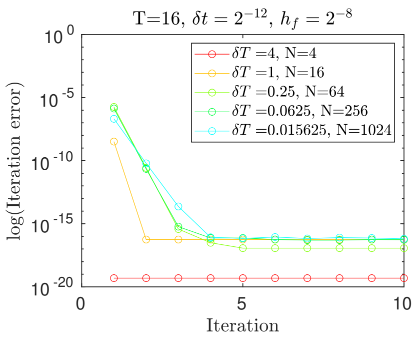
MG-parareal Figure 13 shows the MG-parareal algorithm with one pre- and one post-smoothing step, which is fixed in this study. We recognize that the smaller the macro step size is, the more iterations are required and the slightly higher the size of the iteration error and of the defect are by reaching stagnation. The smaller the micro step size is, the fewer iterations are required due to the arising number of levels. The error is independent of the macro step size, but it gets smaller for smaller micro time step sizes and spatial step sizes.



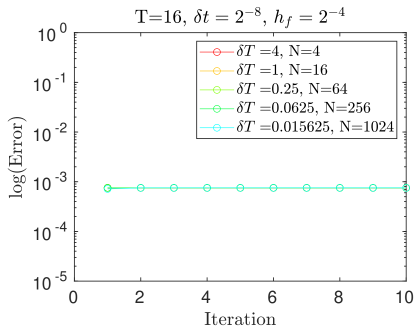
(a) Iteration error



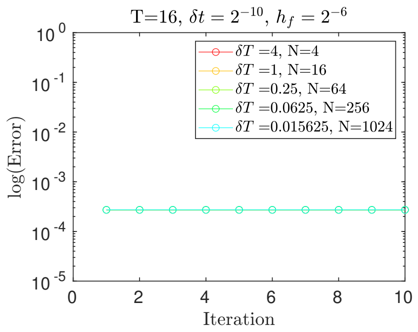
(b) Iteration error



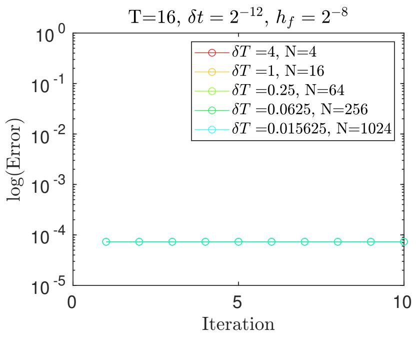
(c) Iteration error



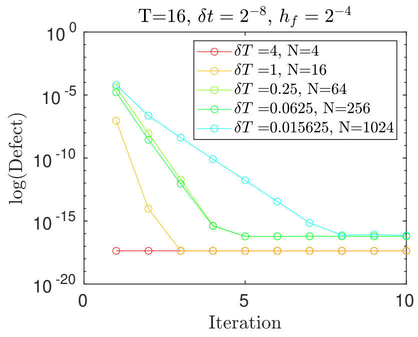
(d) Error



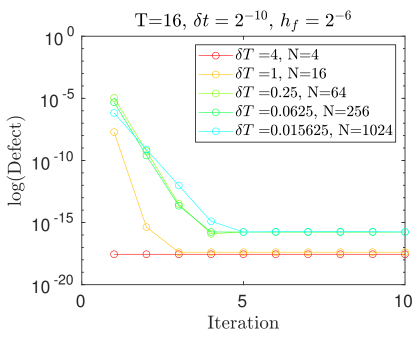
(e) Error



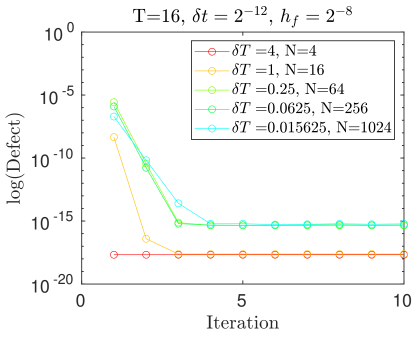
(f) Error



(g) Defect



(h) Defect

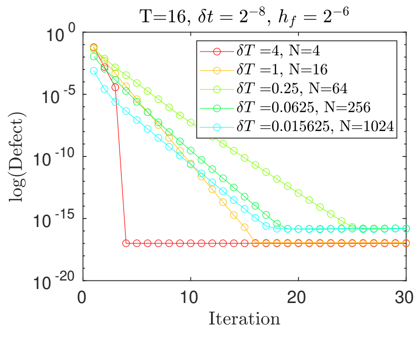


(i) Defect

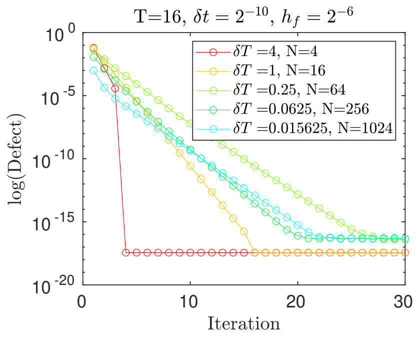
Figure 13: Iteration error, error and defect of the MG-parareal algorithm for $\delta t = 2^{-8}, 2^{-10}, 2^{-12}$ and $h_f = 2^{-4}, 2^{-6}, 2^{-8}$ per column with $T = 16$.

Parareal vs. MG-parareal We recognize in Figure 14 that for smaller macro steps, the number of iterations of the parareal algorithm increases and of the MG-parareal decreases, the larger the end time is. Also the smaller the micro time step size is and now with a constant spatial step size, the better the convergence behavior for the MG-parareal algorithm is. By using the parareal algorithm, the number of iterations either remains constant or increases.

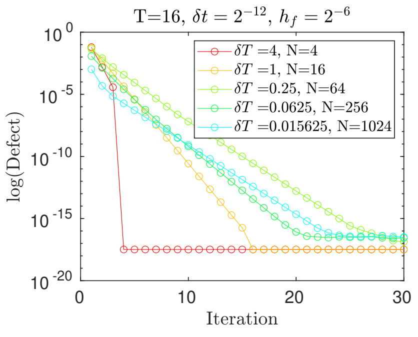
Parareal and SC-parareal We consider an overview of the parareal algorithm and the SC-parareal algorithm for different spatial and micro step sizes per column in Figure 15. In line one and two we see the known behavior of the defect. The number of iterations of both methods increases for smaller micro time step sizes and fine spatial step sizes. The coarsening in space therefore does not necessarily lead to a degradation of the speed of convergence.



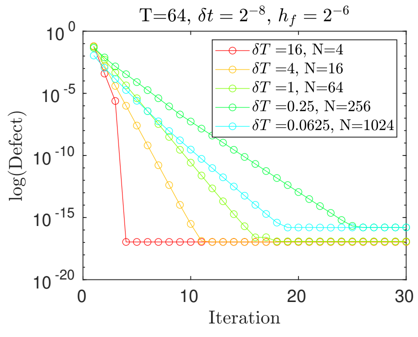
(a) Parareal $T = 16$



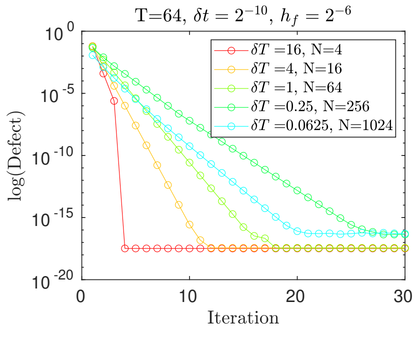
(b) Parareal $T = 16$



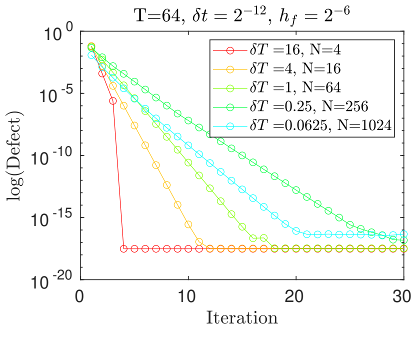
(c) Parareal $T = 16$



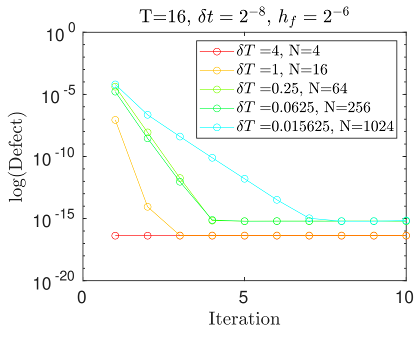
(d) Parareal $T = 64$



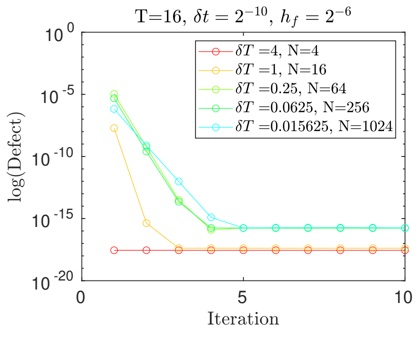
(e) Parareal $T = 64$



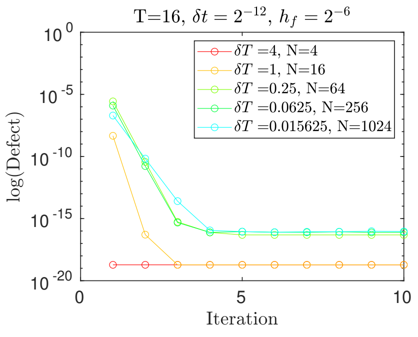
(f) Parareal $T = 64$



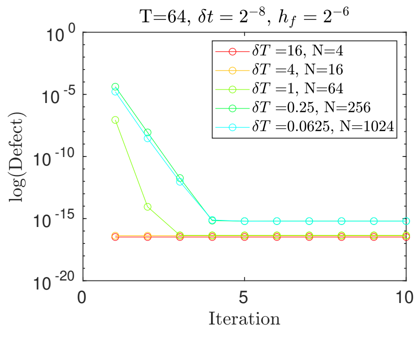
(g) MG-parareal $T = 16$



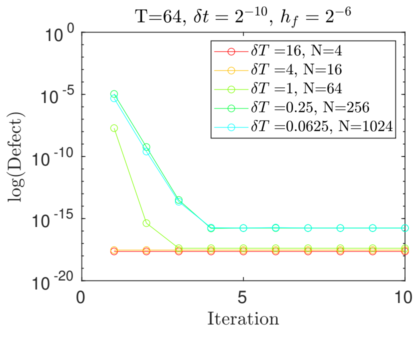
(h) MG-parareal $T = 16$



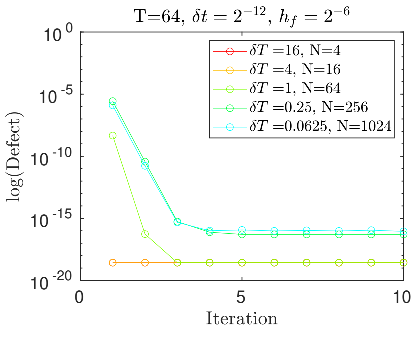
(i) MG-parareal $T = 16$



(j) MG-parareal $T = 64$

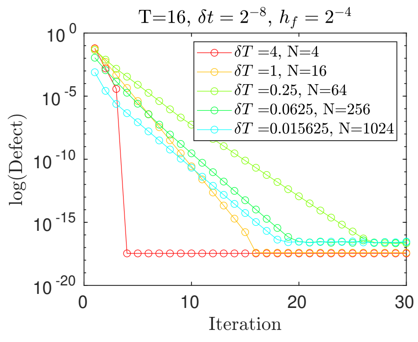


(k) MG-parareal $T = 64$

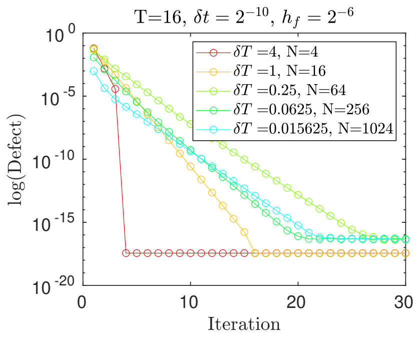


(l) MG-parareal $T = 64$

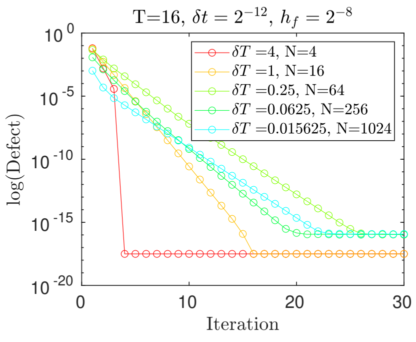
Figure 14: Defect of the parareal and MG-parareal algorithm for $\delta t = 2^{-8}, 2^{-10}, 2^{-12}$ and $h_f = 2^{-6}$ per column with $T = 16, 64$.



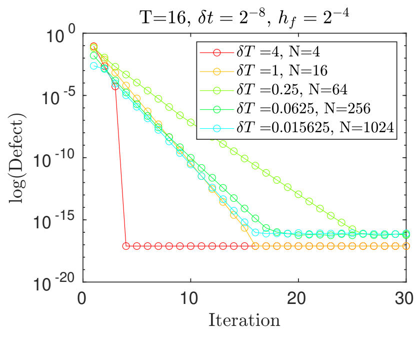
(a) Parareal



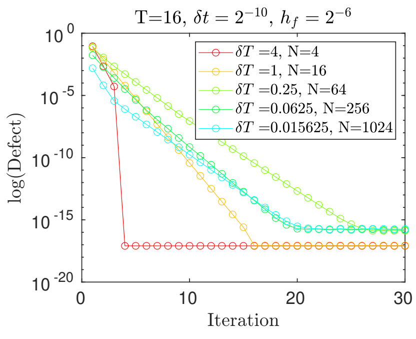
(b) Parareal



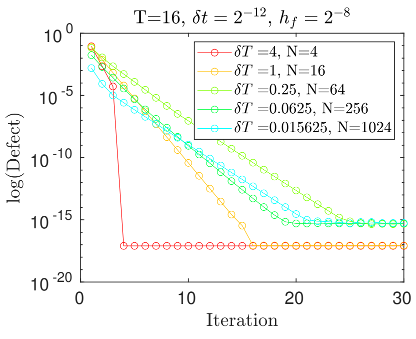
(c) Parareal



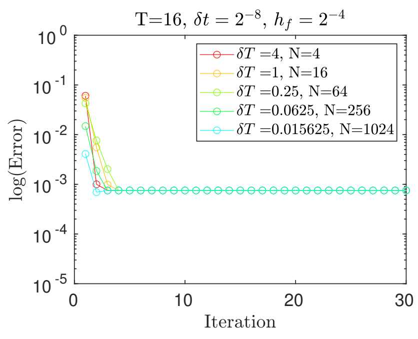
(d) SC-parareal



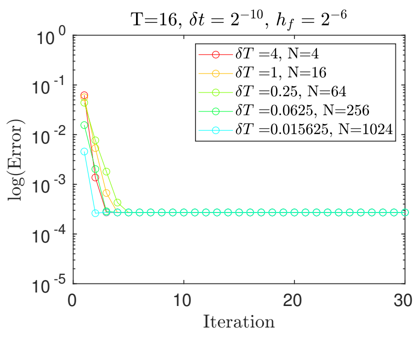
(e) SC-parareal



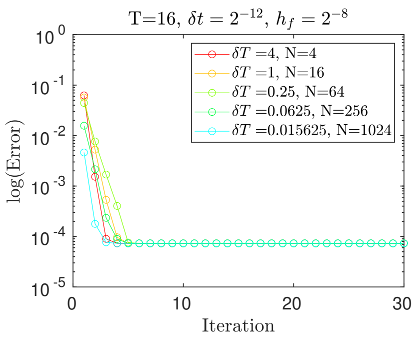
(f) SC-parareal



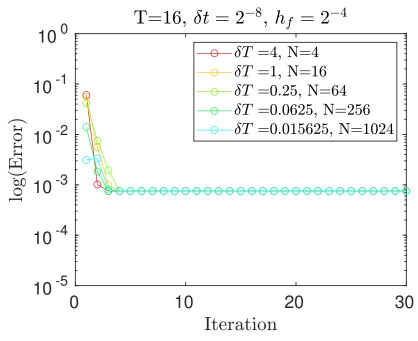
(g) Parareal



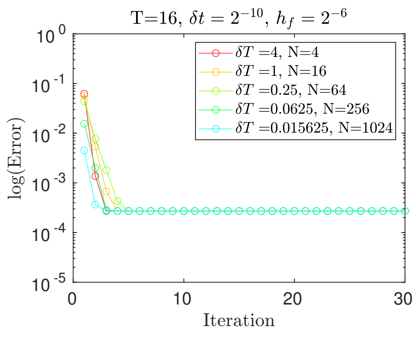
(h) Parareal



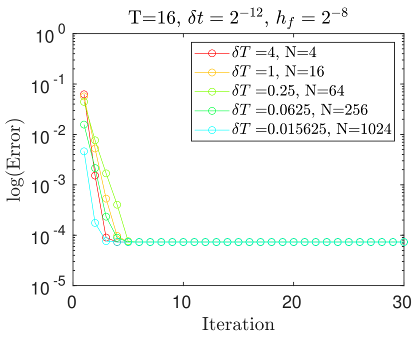
(i) Parareal



(j) SC-parareal



(k) SC-parareal

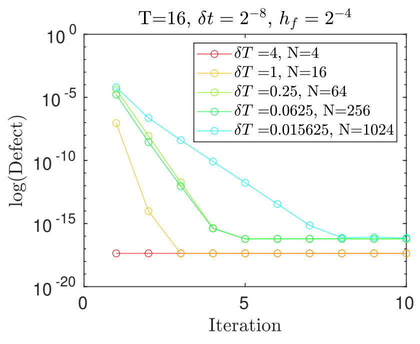


(l) SC-parareal

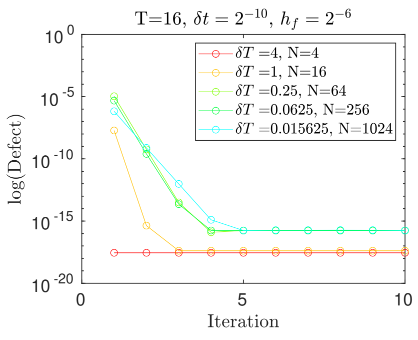
Figure 15: Defect (line one and two) and error (line three and four) of the parareal and SC-parareal algorithm for $\delta t = 2^{-8}, 2^{-10}, 2^{-12}$ and $h_f = 2^{-4}, 2^{-6}, 2^{-8}$ per column with $T = 16$.

The size of the defect also increases for smaller spatial step sizes and micro time step sizes. The errors of the methods in lines three and four are identical.

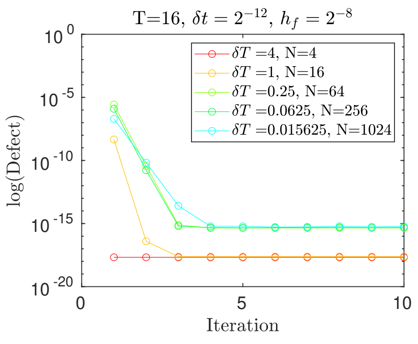
MG-parareal, MG-SC-parareal and MG-LSC-parareal If we compare the defects of our multigrid versions in Figure 16, there is almost no difference. With all of our multigrid variants, the convergence speed is better for a smaller micro time step size and spatial step size. The level of stagnation of the defect also increases minimally in all procedures. The errors that we see in Figure 17 are identical. The spatial coarsening strategy has no influence on the errors, since it only influences the spatial step sizes and not the micro time step size.



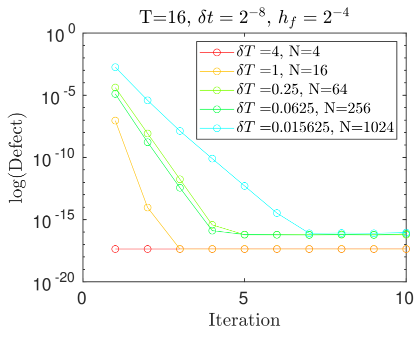
(a) MG-parareal



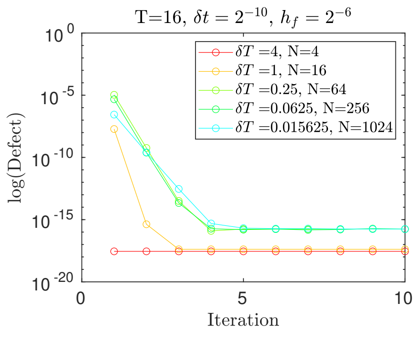
(b) MG-parareal



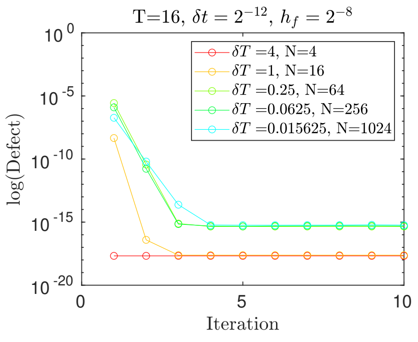
(c) MG-parareal



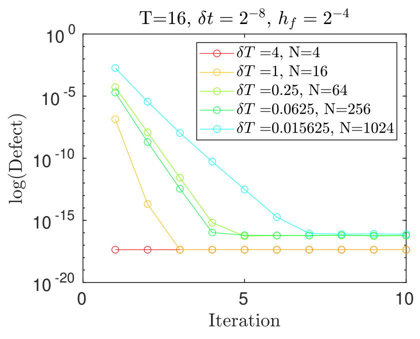
(d) MG-SC-parareal



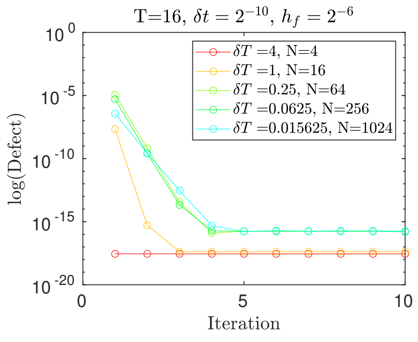
(e) MG-SC-parareal



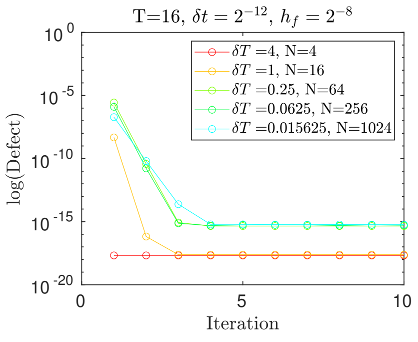
(f) MG-SC-parareal



(g) MG-LSC-parareal

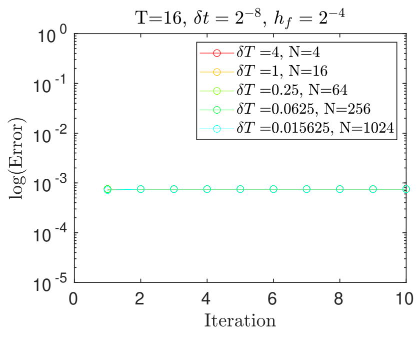


(h) MG-LSC-parareal

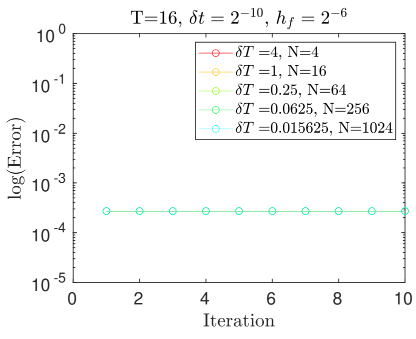


(i) MG-LSC-parareal

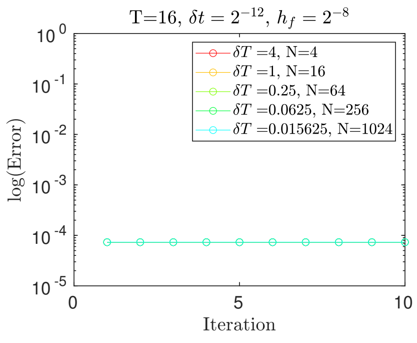
Figure 16: Defect of the multigrid versions for $\delta t = 2^{-8}, 2^{-10}, 2^{-12}$ and $h_f = 2^{-4}, 2^{-6}, 2^{-8}$ per column with $T = 16$.



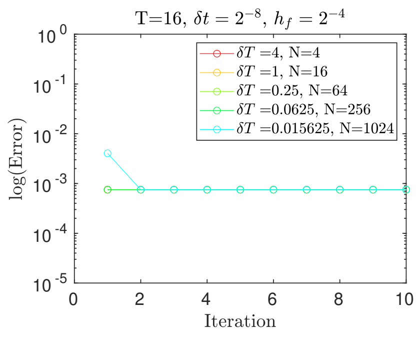
(a) MG-parareal



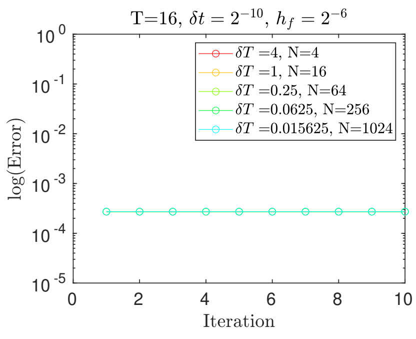
(b) MG-parareal



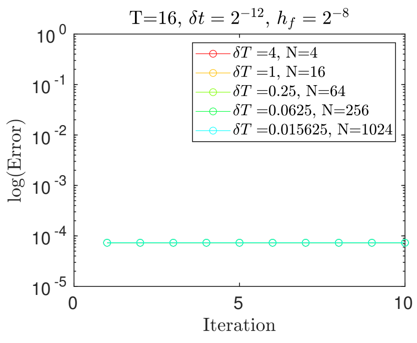
(c) MG-parareal



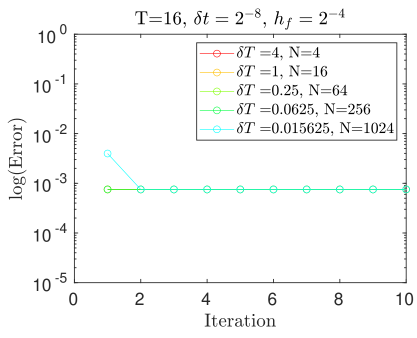
(d) MG-SC-parareal



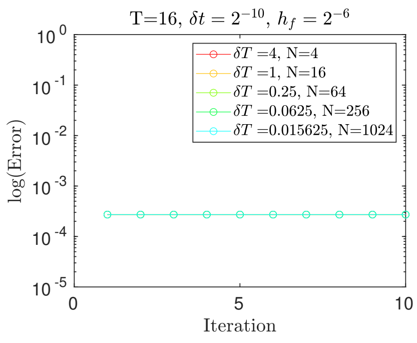
(e) MG-SC-parareal



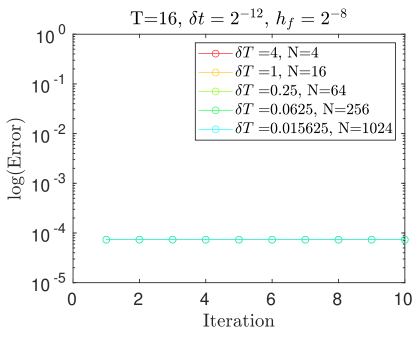
(f) MG-SC-parareal



(g) MG-LSC-parareal



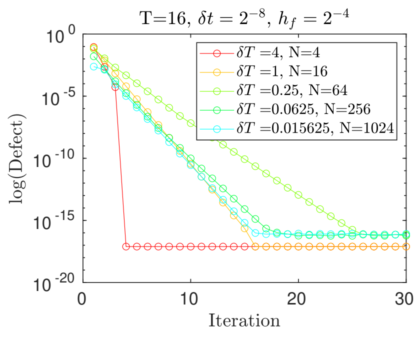
(h) MG-LSC-parareal



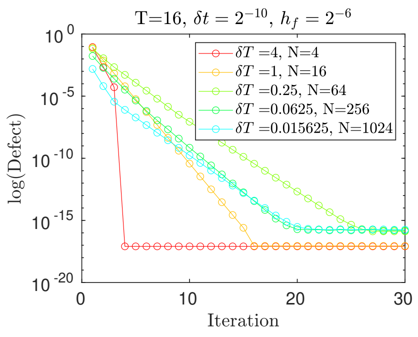
(i) MG-LSC-parareal

Figure 17: Error of the multigrid versions for $\delta t = 2^{-8}, 2^{-10}, 2^{-12}$ and $h_f = 2^{-4}, 2^{-6}, 2^{-8}$ per column with $T = 16$.

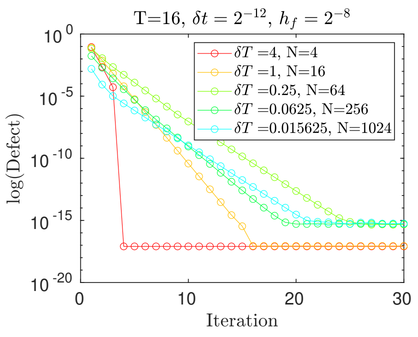
Spatial coarsening variants In Figure 18 we present a comparison of the SC-parareal algorithm, the MG-SC-parareal algorithm and the MG-LSC-parareal algorithm. Small micro time and spatial step sizes were used for each column. If we compare the two MG-parareal variants with the parareal algorithm, we see that both multigrid variants converge better. Again, we can convert the MG-parareal iterations into parareal iteration, so that with a smoothing step, a pre- and post-smoothing step, one MG-parareal step represents three parareal steps. In the last column, the SC-parareal needs 25, 19 and 21 iterations for the smallest macro step sizes with $N = 64, 256, 1024$. The MG-SC-parareal and the MG-LSC-parareal use only 9, 9 and 12 parareal iterations.



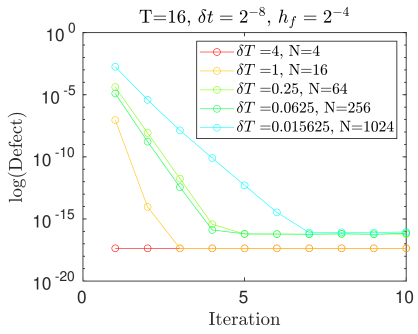
(a) SC-parareal



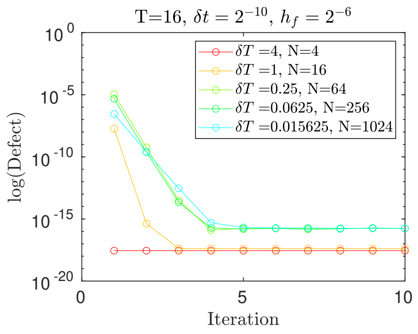
(b) SC-parareal



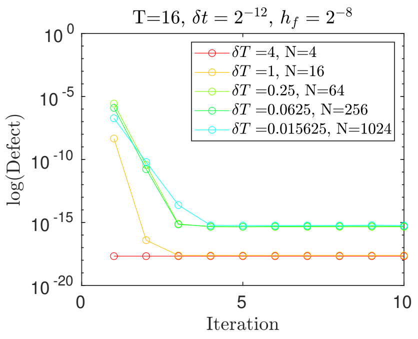
(c) SC-parareal



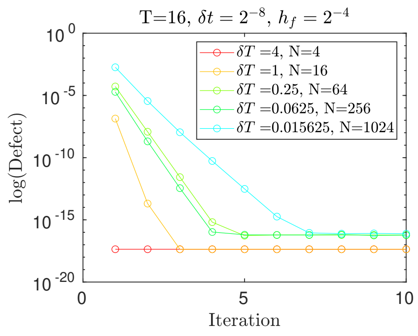
(d) MG-SC-parareal



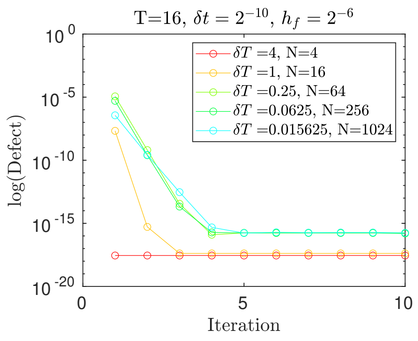
(e) MG-SC-parareal



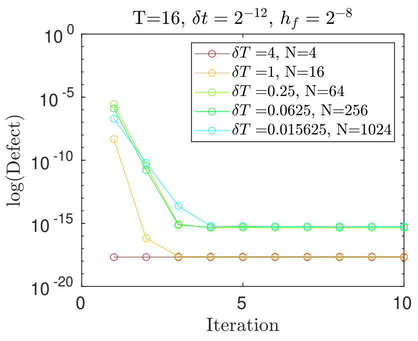
(f) MG-SC-parareal



(g) MG-LSC-parareal



(h) MG-LSC-parareal



(i) MG-LSC-parareal

Figure 18: Defect of the variants with spatial coarsening for $\delta t = 2^{-8}, 2^{-10}, 2^{-12}$ and $h_f = 2^{-4}, 2^{-6}, 2^{-8}$ per column with $T = 16$.

Numerical complexity We consider all the presented methods for a small spatial step size and a medium micro step size. The stopping tolerance is set as 10^{-9} and the CPU time is given in seconds. The MG-variants are carried out with one pre- and post-smoothing step. The CPU time of the micro operator, the residual calculation and the spatial transfer per processor are given, since this is calculated in parallel.

We start with a small spatial step size and medium micro time step size in Table 5. The required CPU time by the macro operator increases and that of the micro operator decreases for a higher number of processors. The total run time is slightly shorter with the SC-parareal algorithm than with the parareal algorithm. Since we only generate a change in the macro operator's CPU time due to coarsening in space, the macro execution time is reduced. The MG-parareal requires a longer execution time compared to the parareal

Table 5: CPU-time in seconds with $T = 16$, $\delta t = 2^{-12}$ and $h_f = 2^{-10}$

	N	16	64	256	1024
Parareal	k	9	14	11	11
	τ_G	0.0278	0.1323	0.7101	2.4147
	τ_F	6.0201	2.2132	0.4611	0.0942
	τ	6.0480	2.3454	1.1711	2.5089
SC-parareal	k	9	14	11	11
	τ_G	0.0147	0.1111	0.2686	2.0049
	τ_F	5.9633	2.2183	0.4537	0.0893
	τ_T	0.0067	0.0093	0.0066	0.0077
	τ	5.9846	2.3388	0.7289	2.1018
MG-parareal	k	2	3	3	3
	τ_G	0.1115	0.5229	2.7167	4.2189
	τ_F	6.5548	2.5070	0.6763	0.1425
	τ_R	0.0482	0.2605	0.2354	0.5882
	τ	6.7144	3.2904	3.6284	4.9497
MG-SC-parareal	k	2	3	3	3
	τ_G	0.0806	0.3772	0.9795	3.3632
	τ_F	6.5270	2.5946	0.6451	0.1392
	τ_R	0.0172	0.0522	0.1052	0.4809
	τ_T	0.0784	0.0958	0.0757	0.1492
	τ	6.7032	3.1198	1.8056	4.1326
MG-LSC-parareal	k	2	3	3	3
	τ_G	0.0411	0.1696	0.5694	1.8051
	τ_F	4.7100	1.8022	0.3104	0.0987
	τ_R	0.0092	0.0114	0.0227	0.0942
	τ_T	0.0370	0.0543	0.0373	0.0380
	τ	4.7973	2.0375	0.9397	2.0360

algorithm, since the difference between macro and micro step size is not large enough to generate enough levels. The spatial coarsening has a positive effect on the macro execution time of the MG-SC-parareal algorithm whereas the micro execution time remains almost unchanged in comparison with the MG-parareal. The transfer costs of the MG-SC-parareal are higher than for the SC-parareal, because we perform the transfer on the levels. The MG-LSC-parareal algorithm requires the shortest micro and macro CPU time and so as well total CPU time of all shown methods.

Table 6: CPU-time in seconds with $T = 16$, $\delta t = 2^{-14}$ and $h_f = 2^{-10}$

Methods	k	τ_G	τ_F	τ_R	τ_T	τ
Parareal	14	0.1416	8.1760	-	-	8.3176
SC-parareal	14	0.0976	8.1810	-	0.0084	8.2870
MG-parareal	3	0.6656	9.2204	0.0626	-	9.9487
MG-SC-parareal	3	0.4029	9.2200	0.0604	0.1946	9.8779
MG-LSC-parareal	3	0.2219	6.9472	0.0141	0.0690	7.2522

In order to be able to recognize a clear effect of the MG-LSC-parareal algorithm, we consider a smaller micro step size $\delta t = 2^{-14}$ in Table 6. Here we can see that the MG-LSC-parareal approach requires the least total CPU time. In addition, the micro work is much less compared to the other processes. Even if the multigrid variants resulted from the spatial coarsening in order to reduce the serial execution time, we see in the table and also in the theoretical complexity that the micro work is the central issue. The macro effort in long-term simulations with a very small micro time step size is much less than the micro effort, so we can assume that the macro effort is negligible. Accordingly, the MG-LSC-parareal algorithm has huge advantages, since the parallel execution time also drops significantly.

5.3 Comparison with results from the literature

In order to better assess the efficiency of our methods, we compare them with another interpretation of a *multigrid in time algorithm*, which was published by S. Friedhoff, R. D. Falgout, T. V. Kolev, S. MacLachlan, J. B. Schroder in 2013 [2]. They present in their work, *A Multigrid-in-Time Algorithm for Solving Evolution Equations in parallel*, a similar development of the parareal algorithm, which is denoted by *full multilevel algorithm*. This algorithm was referred to as the MGIT algorithm and represents the first publication as an technical report of the MGRIT algorithm, which was published in [1]. They interpreted the parareal algorithm as *two level reduction method*, which resulted in an optimally scaled multigrid method over time.

Similar to our MG-parareal algorithm, it uses a V-cycle. The difference between our version and the *full multilevel algorithm* lies among others in the coarsening strategy of the micro and macro time grids and the parallel access. In our MG-parareal algorithm, the macro step size is not changed and the micro step size is doubled per level. Thus our multigrid level hierarchy arises from using a coarser micro grid where we achieve a parallel access through the parareal algorithm, as the smoother. In their version they use for the parallel access an FCF-relaxation, which also decomposes the time interval in micro and macro time steps, but the multigrid hierarchy is achieved by seeing this macro grid on level l as the coarser micro grid on level $l+1$. So the macro steps are always selected as m times the micro step size: $\Delta T = m\delta t$. On the next level, the micro step size is set as the macro step size on the higher level $\delta t_l = \Delta T_{l-1}$. We used the same strategy in our spatial coarsening with $m = 2$. Furthermore the FCF-relaxation is carried out parallel and also here each macro interval is assigned to a processor. But the number of used processors is reduced per level due to their smaller macro grid. One F-relaxation describes the execution of the micro operator and so the update of the micro time points based on the previous macro time points. The C-relaxation is an update of the macro time points based on the previous micro time points. If we only use an F-relaxation, this would lead to the original parareal algorithm.

The one-dimensional heat equation was considered in [2]

$$u_t = \kappa u_{xx} + b(x,t), \quad \kappa > 0, \;\; x \in [0,\pi], \; t \in [0,T], \tag{16}$$

with homogeneous Dirichlet boundary conditions and an initial condition

$$\begin{aligned} u(x,0) &= u_0(x), \;\; x \in [0,\pi] \\ u(0,t) &= u(\pi,t) = 0, \;\; t \in [0,T]. \end{aligned}$$

In the numerical tests in [2], the right hand side was set to zero and the initial condition to $u(0,x) = sin(x)$. As stopping condition the tolerance 10^{-9} was used. The grid points were identified with $x_j = j\Delta x,\ \ j = 0, 1, \ldots, N_x + 1$ with a number of spatial grid points $N_x + 1$. The micro step size is $\delta t = (\Delta x)^2$ and the macro step size is $\delta T = 2\delta t$. In our notations, this results in a number of micro intervals per macro interval of $n_\mu = 2$. The parameters N_x and N_t are varied in the following tables, where N_t describes the number of total micro intervals. The micro grid points were given with $t_i = i\delta t,\ \ i = 0, 1, \ldots, N_t = N \cdot n_\mu = N \cdot 2$. Furthermore, the number of processors is $N = \frac{N_t}{2}$ and the end time is $T = \delta t \cdot N_t$. Now the end time changes when the total micro grid points N_t are varied and the macro step size remains constant.

Table 7: Number of iterations for the spatial grid $N_x = 16$ and different time grids N_t. The results of the *two level* and *full multilevel* methods are from Friedhoff et. al. [2], Table one and three.

Methods \ N_t	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}
Parareal	9	9	9	9	9	9	9
SC-parareal	11	11	11	11	11	11	11
MG-parareal	5	5	5	5	5	5	5
MG-SC-parareal	6	6	6	6	6	6	6
MG-LSC -parareal	6	6	6	6	6	6	6
FCF-relaxation two level	6	7	7	7	7	7	7
FCF-relaxation full multilevel	6	7	7	8	8	8	9
F-relaxation two level	9	9	9	9	9	9	9
F-rexation full multilevel	10	13	16	20	23	23	25

In order to be able to compare our processes better, we solve the same problem (16). Our multigrid parareal variants are carried out with one pre- and post-smoothing step. In Table 7, a comparison of the number of required iterations of the algorithms from this work as well as the results from the paper by Friedhoff [2] (Table one and three), with the *two level* and *full multilevel* Versions of their *multigrid-in-time algorithm* with coarsening factor two with F-relaxation and FCF-relaxation, is shown. The number of time intervals varies for a fixed number of spatial intervals and this changes the number of processors and the end time points while the macro time step size is fixed. This test was carried out by Friedhoff et. al. to see if the convergence is independent of the problem size. The results show that our methods, the methods with the FCF-relaxation as well as the *two level* algorithm with F-relaxation are independent of the problem size. Only the *full multilevel* approach with F-relaxation has a large and increasing number of iterations. The *two level* with F-relaxation corresponds to the classic parareal algorithm, so the results are the same. The number of required iterations does not change for any of our multigrid variants for a larger number of processors. Although the number of levels is only two, which is too low to expect good results from the multigrid parareal variants, the number of required iterations is slightly less than the number of *multilevel* procedures from the other paper.

Table 8 shows our results and the results from the paper [2], Table two, with the *two level* and *full multilevel* versions of the *multigrid-in- time algorithm* with coarsening factor two and FCF-relaxation. Here the number of time- and spatial intervals have now been varied. We see that all tested methods are independent of the problem size.

Table 9 presents a comparison of the algorithms from this work as well as the results from the paper by Friedhoff [2], Table four, with coarsening factor four and FCF-relaxation. The

Table 8: Number of iterations for different spatial grids N_x and time grids N_t. The results of the *two level* and *full multilevel* methods are from Friedhoff et. al. [2], Table two.

Methods \ N_x / N_t	2^4 / 2^5	2^5 / 2^7	2^6 / 2^9	2^7 / 2^{11}
Parareal	9	9	9	9
SC-parareal	11	12	13	13
MG-parareal	5	5	5	5
MG-SC-parareal	6	7	6	6
MG-LSC-parareal	6	7	6	6
FCF-relaxation two level	6	7	7	6
FCF-relaxation full multilevel	6	8	9	9

Table 9: Number of iterations for a spatial grid $N_x = 16$ and time grids N_t. The results of the *two level* and *full multilevel* methods are from Friedhoff et. al. [2], Table four.

Methods \ N_t	4^3	4^4	4^5	4^6	4^7
Parareal	12	12	12	12	12
SC-parareal	10	10	10	10	10
MG-parareal	5	5	5	5	5
MG-SC-parareal	4	4	4	4	4
MG-LSC-parareal	4	4	4	4	4
FCF-relaxation two level	7	7	7	7	7
FCF-relaxation full multilevel	7	7	9	9	9

authors thus showed that the results of their *multilevel* procedures seem to be independent of the coarsening factor. For our algorithms, this test means that we have a higher difference between micro and macro operators and therefore a larger number of levels. The number of iterations of our multigrid variants has decreased accordingly. The *two level* and *full multilevel* methods tend to use macro and micro operators with a small difference in contrast to our variants which are getting better for a bigger difference. So we can expect better results for our multigrid variants especially for the MG-LSC-parareal algorithm when the number of processors is limited and we postulate to achieve high accuracy.

6 Conclusions

In this work we introduced new multigrid variants of the parareal algorithm. We used the parareal algorithm as a smoother and a coarsening strategy in time, which resulted from doubling the micro grid and keeping the macro grid fixed. In the case of an ordinary differential equation and the heat equation, it should be emphasized that, in contrast to the classical parareal algorithm, the MG-parareal algorithm takes advantage of a small micro time step size. In the heat equation, we added an additional spatial coarsening strategy to the MG-parareal algorithm. Even if they resulted from spatial coarsening in order to reduce the serial execution time, the MG-LSC-parareal algorithm has for small spatial and micro time step sizes the lowest micro computing time and so the lowest total CPU time. This is of great importance because in long-term simulations with a very small micro time step size, it is not the serial computing time that is problematic, but the parallel computing time, so the effort of the micro operator. From this, we concluded that the macro effort for such problems can be neglected. The strength of our multigrid variants is the large difference between the macro and micro operator and can thus be used optimally for problems with a limited number of processors, which should achieve a high accuracy of the solution. The MG-LSC-parareal algorithm has very good results in reducing the required computing time for this type of problems.

From the comparison with the work of Friedhoff et. al. [2], in which a *full multilevel algorithm* was also presented, we were able to conclude that our multigrid variants produce good results and are comparable to other methods. Further comparisons with other time-parallel algorithms must be examined in future, as well as the application to more complex differential equations.

References

(1) R. D. Falgout, S. Friedhoff, TZ. V. Kolev, S. P. MacLachlan, J. B. Schroder. *PARALLEL TIME INTEGRATION WITH MULTIGRID.* SIAM Journal on Scientific Computing, 36, No.6, pp. C635–C661, 2014

(2) R. D. Falgout, S. Friedhoff, T. V. Kolev, S. MacLachlan, J. B. Schroder. *A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel.* Student paper winner, Sixteenth Copper Mountain Conference on Multigrid Methods, Copper Mountain, Colorado. March, 2013. LLNL-CONF-606952, 2013

(3) M. J. Gander. *50 years of Time Parallel Time Integration.* Multiple Shooting and Time Domain Decomposition Methods, pp.69–113, Springer, 2015

(4) M. J. Gander, S. Vandewalle. *Analysis of the parareal time-parallel time-integration method.* SIAM Journal on Scientific Computing, 29(2), 556–578, 2007

(5) M. J. Gander, S. Vandewalle. *On the superlinear and linear convergence of the parareal algorithm.* In Domain Decomposition Methods in Science and Engineering XVI, Springer, pp. 291–298, 2007

(6) 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 (2018), pp. 59–74, 2018

(7) W. Hackbusch. *Parabolic multi-grid methods.* In Computing Methods in Applied Sciences and Engineering, VI, R. Glowinski and J.-L. Lions, eds., North-Holland, 1984, pp. 189–197, 1984

(8) J.-L. Lions, Y. Maday, and G. Turinici. *A "Parareal" in time discretization of PDE's.* C. R. Math. Acad. Sci. Paris, 332, no. 7, pp. 661-668, 2001

(9) Y. Maday *The parareal in time algorithm.* Technical Report R08030, Universite Pierré et Marie Curie, 2008

(10) Y. Maday, G. Turinici. *A parareal in time procedure for the control of partial differential equations.* Comptes Rendus Mathematique - C R MATH, vol. 335, no. 4, pp. 387-392, 2002

(11) M. L. Minion. *A hybrid parareal spectral deferred corrections method.* Communications in Applied Mathematics and Computational Science, 5 (2010), pp. 265–301, 2012

(12) D. Ruprecht. *Convergence of Parareal with spatial coarsening.* PAMM 14, Nr. 1, 1031–1034, 2014

(13) D. Ruprecht, R. Speck, R. Krause. *Parareal for diffusion problems with space- and time-dependent coefficients.* Domain Decomposition Methods in Science and Engineering XXII, pp. 371-378, Springer, 2016

(14) G. A. Staff. *The Parareal Algorithm- A survey of present work.* Technical report, Norwegian University of Science and Technology, Trondheim, Norway, 2003

(15) K. Stüben, U. Trottenberg. *Multigrid methods: Fundamental algorithms, model problem analysis and applications.* In W. Hackbusch and U. Trottenberg (Eds.), Multigrid Methods, volume 960 of Lecture Notes in Mathematics, pp. 1-176, Springer, 1982