Institut für Numerische Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn
Friedrich-Hirzebruch-Allee 7 • 53115 Bonn • Germany
phone +49 228 73-69828 • fax +49 228 73-69847
www.ins.uni-bonn.de

M. Griebel, M. A. Schweitzer, L. Troska

A Parallel-in-Time Combination Method for Parabolic Problems

INS Preprint No. 2505

September 2025

A PARALLEL-IN-TIME COMBINATION METHOD FOR PARABOLIC PROBLEMS

MICHAEL GRIEBEL*, MARC ALEXANDER SCHWEITZER*, AND LUKAS TROSKA*

Abstract. In this article, we present a parallel discretization and solution method for parabolic problems with a higher number of space dimensions. It consists of a parallel-in-time approach [10] using the multigrid reduction-in-time algorithm `MGRIT` [9] with its implementation in the library `XBraid` [1], the sparse grid combination method [7, 17] for discretizing the resulting elliptic problems in space, and a domain decomposition method [18] for each of the subproblems in the combination method based on the space-filling curve approach. As a result, we obtain an extremely fast and embarrassingly parallel solver with excellent speedup and scale-up qualities, which is perfectly suited for parabolic problems with up to six space dimensions.

We describe our new parallel approach and show its superior parallelization properties for the heat equation, the chemical master equation and some exemplary stochastic differential equations.

Key words. parabolic differential equation, parallel-in-time method, multigrid reduction-in-time, sparse grid combination method, space filling curve, domain decomposition, parallelization

MSC codes. 65M55, 65M22, 65M99, 65F08, 65Y05, 65Y99

1. Introduction. We consider the numerical approximation of parabolic partial differential equations

$$\frac{\partial u}{\partial t} + \mathcal{L}u = f \quad \text{in } \Omega \times (T_{\text{start}}, T_{\text{end}}], \tag{1.1}$$

with $\Omega \subset \mathbb{R}^d$ and linear elliptic differential operator $\mathcal{L}$, where we are especially interested in the case of larger spatial dimensions d. Such problems arise, for example, from stochastic differential equations, where modeling with the probability density function of the underlying Markov process leads to the well-known general Kolmogorov forward or the Fokker-Planck equation. It is used, for example, to describe the particle velocities in gases and liquids in statistical mechanics and thermodynamics, the reactions of concentrations of reactants in a chemical system, the dynamics of friction and wear of mechanical systems in engineering, the evolution of populations of species in cell biology, genetics and ecology, the firing rate of neurons in neuroscience, the dynamics of quantum mechanical systems through the Wigner function, the evolution of asset prices or volatility for option pricing and portfolio optimization in financial mathematics or the modeling of cognitive processes in behavioral science and decision making in psychology, to name just a few applications.

For its numerical approximation we encounter the method of lines, where the problem is first discretized in space and the resulting system of ordinary differential equations is then treated by a numerical integration approach such as the Runge-Kutta method. An alternative and more common approach is the Rothe method, where the problem is first discretized in time, ressulting in a sequence of elliptic problems, each of which is then discretized in space by a finite element or finite difference method. The latter approach uses a sequential time stepping scheme, where within each time step an elliptic subproblem must be solved, where the operator and right hand side depend on $\mathcal{L}$, f, and the respective time stepping method.

There are two major problems here: First, when it comes to parallelization, in Rothe's approach the use of large-scale parallel systems is usually limited to parallel elliptic solvers, i.e. only the spatial

*Institut für Numerische Simulation, Universität Bonn, Friedrich-Hirzebruch-Allee 7 53115 Bonn and Fraunhofer Institut für Algorithmen und Wissenschaftliches Rechnen SCAI, Schloss Birlinghoven, 53757 Sankt Augustin

dimensions are considered for parallelization and the sequence of time steps is processed only sequentially, which typically requires very long runtimes to achieve the necessary accuracy. To solve this problem, we resort to a parallel-in-time approach [10], to be precise to `MGRIT` [9] in the form of the multigrid reduction-in-time algorithm via its implementation in the library `XBraid` [1]. It gives us a parallelization of the whole problem for the time coordinate, which can be considered as a *first scale of parallelization*. Second, for larger spatial dimensions d we encounter the well-known curse of dimension, either in the size of the ODE system for the method of lines or in the discretization of the spatial problems in Rothe's method. There, a space discretization on a uniform grid with mesh size 2^L in each coordinate direction is usually used, with associated level L. Thus, the cost scales as $O(2^{Ld})$, i.e. it basically scales exponentially with d, which makes any conventional discretization approach impossible for $d > 3$, both for the method of lines and for Rothe's approach.

In this article we will focus on Rothe's method, but instead of a uniform discretization in space, we will use a sparse grid approach [7] to discretize the elliptic problems at each time step in the form of the so-called combination method [17]. This alleviates the curse of dimension and practically allows values of d up to six. It exploits the fact that the resulting elliptic problems in the time step sequence (right hand side and boundary conditions permitting) generally have a smooth solution in each time step (perhaps except for the first one). This is due to the smoothing effect of $\mathcal{L}$ over time. The sparse grid combination method involves solving the elliptic problems discretized as several different subproblems on (mostly) anisotropic grids Ω_l, with mesh sizes $h_l = (2^{-l_1}, \ldots, 2^{-l_d})$, where

$$l = (l_1, \ldots, l_d) \quad \text{with } l_1 + \ldots l_d = L + d - 1 - q, \quad q = 0, \ldots, d-1, \tag{1.2}$$

and L is the level of the corresponding classical discretization with uniform mesh size $(2^{-L}, \ldots, 2^{-L})$ in each time step of Rothe's method, which would be prohibitive due to the curse of dimension. For practical reasons, we also consider a coarsest minimal level $l > L_0 := (L_{0,1}, \ldots, L_{0,d})$. All these different subproblems of the combination method per each time step can be solved completely in parallel, which is a *second scale of parallelization*. Only their solutions have to be combined properly, which involves communication and some sequential computations. Third, for the solution of each of the subproblems on the levels l of the combination method and for each time step, we use a domain decomposition method, which is based on the space filling curve approach. It was already developed and analyzed in detail in [18]. Thus, the resulting subdomain problems per subproblem (pus a coarse scale problem) of the combination method for each time step can also be treated in parallel, which involves a *third scale of parallelization*.

Overall, by combining the parallel-in-time method, the sparse grid combination method, and the domain decomposition method, we obtain a parallel solution method for parabolic problems (1.1) that allows three scales of nested parallelization, since both spatial and temporal dimensions are treated in parallel. This opens a way to efficiently use an extremely large number of cores. Moreover, the associated communication pattern for the different parallelization scales is characterized as follows: For spatial parallelization, communication is local to each subproblem, since only processes involved in the same subproblem need to exchange data. For the combination method, communication is global, but only on the subproblem scale, i.e. all subproblems have to communicate with each other, but not all processes of one subproblem have to communicate with all processes of another subproblem. Communication on the time scale is again local to each subproblem. The result is an extremely parallel solver with excellent speed-up and scale-up properties that is perfectly suited for modern large-scale massively parallel computing systems of the exascale age. It allows solving Fokker-Planck problems for several important practical applications, bypassing parallel stochastic methods with much inferior accuracy and parallelization properties.

The remainder of this paper is organized as follows: In section 2 we give the theoretical framework and the details of the building blocks of our new approach. In section 3 we present the parallel-in-time combination method for two algorithmic variants, the multigrid reduction-in-time on the sparse grid and the multigrid reduction-in-time on the subproblems. In section 4 we give the results of our parallelization experiments. We consider the heat equation, the chemical master equation, and certain exemplary stochastic differential equations. In section 5 we give some concluding remarks.

2. Theoretical framework. We consider a general parabolic problem of the form

$$\begin{aligned} \frac{\partial u}{\partial t} + \mathcal{L}u = f \text{ in } \Omega \times (T_{\text{start}}, T_{\text{end}}] \\ u = \bar{u}_{\Gamma_D} \text{ on } \partial\Omega \times (T_{\text{start}}, T_{\text{end}}], u = \bar{u}_0 \text{ in } \Omega \times \{T_{\text{start}}\} \end{aligned} \tag{2.1}$$

on some domain $\Omega \subset \mathbb{R}^d$ for start and end time $0 \leq T_{\text{start}} < T_{\text{end}}$, some linear elliptic differential operator $\mathcal{L}$, some Dirichlet boundary conditions $\hat{u}_{\Gamma_D}$ and initial condition $\hat{u}_0$. We are particularily interested in the case $d > 3$ focusing on moderate values of d up to $d = 6$ which precludes any conventional space discretization on uniform grids due to the curse of dimension.

Following the method of lines, we discretize (2.1) for some grid Ω_l on Ω, with mesh size $h_l \approx (2^{l_1}, \ldots, 2^{-l/d})$, where $l = (l_1, \ldots, l_d)$ is a d-dimensional multi-index parameterizing the spatial discretization. This results in a semi-discrete problem, namely the coupled system of ordinary differential equations

$$\begin{aligned} \frac{\partial u_l}{\partial t} + \mathcal{L}_l u_l = f_l \text{ in } \Omega_l \times (T_{\text{start}}, T_{\text{end}}] \\ u_l = \bar{u}_{\Gamma_D} \text{ on } \partial\Omega_l \times (T_{\text{start}}, T_{\text{end}}], u_l = \bar{u}_0 \text{ in } \Omega_l \times \{T_{\text{start}}\}. \end{aligned} \tag{2.2}$$

Next, let $T_{\text{start}} = t_{l,0} < t_{l,1} < \ldots < t_{l,N_l} = T_{\text{end}}$ be a partition of the time interval of interest into $N_l + 1$ time steps $t_{l,n}, 0 \leq n \leq N_l$ with time step size $\Delta t_{l,n} \coloneqq T_{l,n} - T_{l,n-1}, 1 \leq n \leq N_l$. Note that the time partition may depend on the spatial discretization parameter l. Now denote by $\Phi_{l,n}$ the time propagator for time step n of the semi-discrete problem (2.2), i.e.

$$\begin{aligned} u_{l,0} &= g_{l,0} \coloneqq \bar{u}_0 \\ u_{l,n} &= \Phi_{l,n}(u_{l,n-1}) + g_{l,n}, \quad 1 \leq n \leq N \end{aligned} \tag{2.3}$$

for some forcing term $g_{l,n}$. In this paper we will mainly use the backward Euler method, where we have $\Phi_{l,n} = (1 + \Delta t_{l,n}\mathcal{L}_{l,n})^{-1}$ and $g_{l,n} = (1 + \Delta t_{l,n}\mathcal{L}_{l,n})^{-1}\Delta t_{l,n} f_{l,n}$. However, other linear time propagators could be used analogously. Thus, at each time step n we encounter the linear system of equations

$$\tilde{\mathcal{L}}_{l,n} u_{l,n} \coloneqq (\mathcal{L}_{l,n} + \frac{1}{\Delta t_{l,n}}\mathcal{I}_{l,n})u_{l,n} = f_{l,n} + \frac{1}{\Delta t_{l,n}} u_{l,n-1} =: \tilde{f}_{l,n} \quad 1 \leq n \leq N \tag{2.4}$$

with the matrices $\mathcal{L}_{l,n}$, $\mathcal{I}_{l,n}$ and the vectors $f_{l,n}$ of size about $2^{l_1+\ldots+l_d}$ discretizing the operator $\mathcal{L}$, the identity $\mathcal{I}$, and the right-hand-side function f in space on grid Ω_l at time point n.

2.1. Combination method in space for elliptic problems and domain decomposition Solver.

2.1.1. Fundamentals of the combination method. The following is a brief summary of the combination method. For details, the reader is referred to [18] and the references cited therein. The

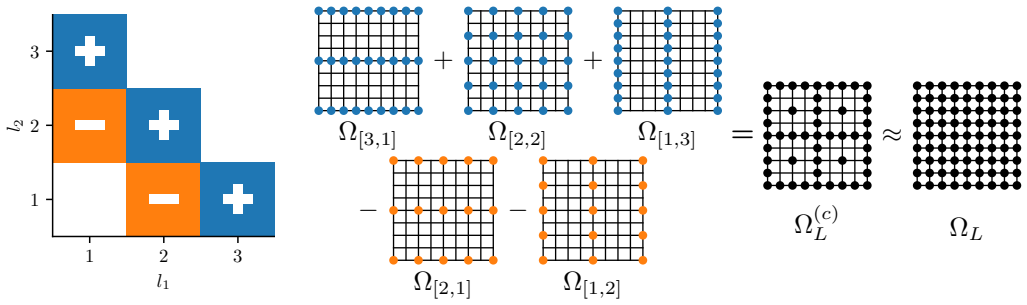

FIGURE 1. *The combination method in two dimensions for level* $L = 3$.

combination method provides a solution to elliptic sparse grid problems in d dimensions by combining solutions to subproblems on (in general) anisotropic grids. In particular, consider a domain $\Omega \subset \mathbb{R}^d$ and let $l = (l_1, \ldots, l_d) \in \mathbb{N}_+^d$ be a level parameterizing a collection of grids $\Omega_l \subset \Omega$ with grid sizes $h = (h_1, \ldots, h_d), h_j \approx 2^{-l_j}, j = 1, \ldots, d$. Then the subproblem associated with l consists of solving the elliptic problem of interest (2.2) for the solution u_l on the subproblem grid Ω_l. With the usual ℓ_1 norm $\|l\|_1 \coloneqq l_1 + \ldots + l_d$ we get the approximate sparse grid solution $u_L^{(c)}$ by combining all subproblem solutions u_l by the combination method according to the formula

$$u(x) \approx u_L^{(c)}(x) \coloneqq \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{\substack{\|l\|_1 = L + (d-1) - q \\ l \geq L_0}} u_l(x) \tag{2.5}$$

for some level $L \in \mathbb{N}$ and initial level $L_0 \in \mathbb{N}$, $1 \leq L_0 \leq L$, where $l \geq L_0$ is to be understood component-wise. If not stated otherwise, we assume that $L_0 = 1$, i.e. we consider the complete selection of subproblems. In some cases it is necessary to discard extremely anisotropic grids by choosing $L_0 > 1$, e.g. to be able to adequately represent initial conditions on each subproblem grid Ω_l. Figure 1 shows an example construction of $u_L^{(c)}$.

Note that all subproblems are independent of each other and can be computed completely in parallel. The cost of the combination method (2.5) must be compared with that of a conventional discretization of a grid with uniform mesh size 2^{-L} for each coordinate direction. There, the number of degrees of freedom scales as $O(2^{dL})$, which expresses the well-known curse of dimensionality that makes such a conventional discretization practically impossible for $d > 3$. In contrast, each of the grids involved in the combination method has only $O(2^L)$ degrees of freedom. Moreover, the number of involved different grids Ω_l is $O(L^{d-1})$, and each of the associated problems can be computed independently in parallel. Only the combination (2.5) of the computed different solutions u_l requires some sequential computations and some communication. The combination method has been shown in [15] to provide an approximation for the solution of elliptic problems that has the same convergence rate and the same order of error as the corresponding sparse grid approximation by the Galerkin method. Furthermore, it is known that the Galerkin sparse grid approximation with piecewise linear basis functions leads to an error of $O(2^{-2L} L^{d-1})$ with respect to the L_2 norm, provided that the solution of the considered continuous elliptic PDE is in the Sobolev space of bounded 2nd mixed deriatives $\mathcal{H}_{mix}^2$, see [18]. For our application, this is indeed the case (for smooth boundary conditions, and perhaps except for the very first time step $n = 1$ for non-smooth initial conditions). This is due to the smoothing property of the elliptic operator $\mathcal{L}$ over time for our parabolic problem (2.1). This convergence rate

must be compared to the rate $O(2^{-2L})$, which would be obtained for a conventional discretization with piecewise linear basis functions on a grid of uniform mesh size $(2^{-L}, \dots, 2^{-L})$. We see that the combination method loses only marginally in terms of error rate (i.e. by a factor of L^{d-1}), but gains enormously in terms of degrees of freedom involved, i.e. it avoids the curse of dimension in the leading cost term. In this paper, we restrict ourselves to the case of piecewise linear basis functions for simplicity. Note however that there are sparse grid combination methods with convergence rates of higher order s based on piecewise polynomials of higher degree for smoother solutions in spaces $\mathcal{H}^s_{mix}$, $s > 2$.

2.1.2. Spatial subproblem solver by domain decomposition. For our application (2.4), all subproblems with index l in the above combination method involve, for any fixed time step n, the solution of an elliptic problem discretized on the grid Ω_l, i.e. each subproblem l can be understood as solving

$$\tilde{\mathcal{L}}_{l,n} u_{l,n} = \tilde{f}_{l,n} \tag{2.6}$$

for $u_{l,n}$ with the given discretized operator $\tilde{\mathcal{L}}_{l,n}$ and the discretized right-hand side $\tilde{f}_{l,n}$ for time step n. Thus, we need a robust and efficient solver for the large number of subproblems on generally anisotropic grids involved in the combination method (2.5), i.e. for the corresponding linear systems of equations (2.6). In this work we use our dimension-oblivious domain decomposition method presented in [18] as the preconditioner for a Krylov iterative solution method. The solver, equipped with our preconditioner, is capable of efficiently handling the solution procedure on all subproblem grids generated by the combination method. In the following we give a short overview of the preconditioning algorithm, for details see [18].

Let Ω_l be any, usually anisotropic, subproblem grid in arbitrary dimension d that appears in the combination method, and let $\tilde{L}_{l,n}$ and $\tilde{f}_{l,n}$ be the corresponding discretized elliptic operator and right-hand side at time point n. We first generate an initial partition $\{\bar{\Omega}_{l,i}\}_{i=1}^{P_l^x}$ of Ω_l by dividing the grid points along a discrete space-filling curve, i.e. the Hilbert curve in d dimensions, into $P_l^x \geq 1$ (approximately) equal-sized disjoint subdomains $\bar{\Omega}_{l,i}$ of size $\lfloor \frac{|\Omega_l|}{P_l^x} \rfloor$ or $\lfloor \frac{|\Omega_l|}{P_l^x} \rfloor + 1$, to be exact, in $P_l^x - (|\Omega_l| - P_l^x \lfloor \frac{|\Omega_l|}{P_l^x} \rfloor)$ subdomains of size $\lfloor \frac{|\Omega_l|}{P_l^x} \rfloor$ and $|\Omega_l| - P_l^x \lfloor \frac{|\Omega_l|}{P_l^x} \rfloor$ subdomains of size $\lfloor \frac{|\Omega_l|}{P_l^x} \rfloor + 1$. Then, each of these subdomains is enlarged by an overlap factor $\gamma > 0$, which controls how many grid points of the neighboring disjoint subdomains $\bar{\Omega}_{l,i\pm k}, k = 1, \dots, \lceil 2\gamma \rceil$ along the space-filling curve must be added to the current subdomain $\bar{\Omega}_{l,i}$ to generate the extended subdomain $\Omega_{l,i}$. The final overlapping domain decomposition of Ω_l is then given by the collection of these $\Omega_{l,i}$. This construction is illustrated for $d = 2$ in Figure 2.

Note that the Hilbert curve is isotropic, but we encounter anisotropic grids in the combination method. To deal with the anisotropy of a grid Ω_l, we adjust in this paper the non-maximum dimensions to the maximum dimension by counting an appropriate number of *virtual* nodes uniformly between the actual grid nodes. Note that these nodes are not inserted into the grid in praxis. They can rather be interpreted as an offset in the enumeration along the space-filling curve. This is in contrast to [18], where the anisotropic grids were virtually padded during enumeration along the space-filling curve by extending the non-maximum dimensions to match the maximum dimension. Our new approach allows for more uniformly shaped subdomains, which slightly improves the convergence rate of the spatial solver. The specific choice

$$\gamma = \frac{1}{2} n \tag{2.7}$$

for some $n \in \mathbb{N}, n \leq P_l^x - 1$, ensures that each grid point is contained in exactly $n + 1$ extended

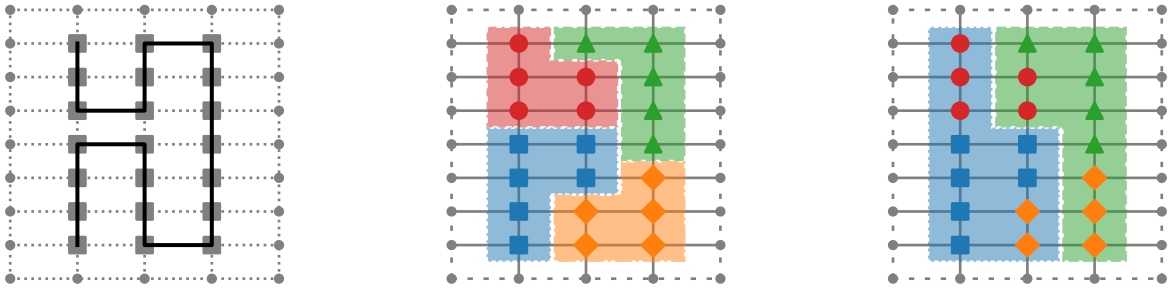

FIGURE 2. *Decomposition of an anisotropic grid with $l = (2, 3)$ by the Hilbert curve approach. Enlargement of the subdomains by $\gamma = 0.25$ and associated overlapping domain decomposition of two of the four subdomains.*

subdomains, see [18]. This also allows the solver to recover from the failure of n neighboring processes in the same iteration, which is an ongoing research topic. The restriction $R_{l,i} \in \mathbb{R}^{|\Omega_{l,i}|\times|\Omega_l|}$ and prolongation $P_{l,i} \coloneqq R_{l,i}^T \in \mathbb{R}^{|\Omega_l|\times|\Omega_{l,i}|}$ corresponding to subdomain $\Omega_{l,i}$, $i = 1, \dots, P_l^x$, are simply given by the trivial injection operators.

Besides a decomposition of the domain on the fine scale like the partition $\{\Omega_{l,i}, i = 1, \dots, P_l^x\}$, a good domain decomposition method also needs a *coarse space problem* for fast convergence. For this, let $|\bar{\Omega}_{l,i}|$ be the size of the i-th disjoint subdomain, and let $1 \leq q_l \leq \min_i |\bar{\Omega}_{l,i}|$ be an integer denoting the number of coarse degrees of freedom per fine-scale subdomain, such that the coarse space contains $q_l \cdot P_l^x$ degrees of freedom. The j-th coarse degree of freedom for subdomain i, $1 \leq j \leq q_l$, consists of the sum of the j-th set of $\frac{|\bar{\Omega}_{l,i}|}{q_l}$ degrees of freedom of $\Omega_{l,i}$, where again the sums for the first few coarse degrees of freedom may include an additional fine-scale degree of freedom if $|\bar{\Omega}_{l,i}|$ is not evenly divisible by q_l. In particular, the algebraic restriction to the coarse space is given by

$$R_{l,0} \coloneqq \operatorname{blockdiag}_{i=1}^{P_l^x}(R_{l,i,0}) \in \mathbb{R}^{(P_l^x \cdot q_l)\times(\prod_{i=1}^{P_l^x}|\Omega_{l,i}|)}, \tag{2.8}$$

where $R_{l,i,0} \in \mathbb{R}^{q_l \times |\Omega_{l,i}|}$ and

$$R_{l,i,0} \coloneqq \overbrace{\begin{pmatrix} 1 & \dots & 1 & & & & & & \\ & & & 1 & \dots & 1 & & & \\ & & & & & & \ddots & & \\ & & & & & & & 1 & \dots & 1 \end{pmatrix}}^{q_l \text{ blocks of size } \approx |\bar{\Omega}_{l,i}|/q_l} \Bigg\} q_l \text{ rows}. \tag{2.9}$$

Note that we associated here the coarse problem to the index $i = 0$ and we associated the subdomain problems to the indices $i = 1, \dots, P_l^x$. Next, we define the local operators as $\tilde{\mathcal{L}}_{l,n,i} \coloneqq R_{l,i}\tilde{\mathcal{L}}_{l,n}R_{l,i}^T, i = 0, \dots, P_l^x$. Let $C_{l,n,(1),D}^{-1}$ be the one-level preconditioner given by

$$C_{l,n,(1),D}^{-1} \coloneqq \sum_{i=1}^{P_l^x} R_{l,i}^T D_{l,i} \tilde{\mathcal{L}}_{l,n,i}^{-1} R_{l,i}, \tag{2.10}$$

where $D_{l,i}$ are diagonal matrices $D_{l,i} \in \mathbb{R}^{N_{l,i}\times N_{l,i}}$ dealing with the multiplicity of degrees of freedom due to the overlapping nature of the subdomains. In this paper, we restrict ourselves to $D_{l,n,i} = \omega_{l,i} I$,

$i = 1, \ldots, P_l^x$, where $\omega_{l,i} = 1/\max_{j \in \bar{\Omega}_{l,i}}(|\{\Omega_{l,k}, k = 1, \ldots, P_l^x : j \in \bar{\Omega}_{l,k}\}|)$, i.e. $\omega_{l,i}$ is the inverse of the maximum number of subdomains that overlap any point contained in $\bar{\Omega}_{l,i}$. Moreover, our specific choice of γ from (2.7) allows us to simplify to $D_{l,n,i} = \frac{1}{n+1} I$ since $|\{\Omega_{l,k}, k = 1, \ldots, P_l^x : j \in \bar{\Omega}_{l,k}\}| \equiv n + 1$ for all grid points j. In [18] other weightings $\omega_{l,i}$ were also presented and studied.

Now let $F_{l,n} \coloneqq R_{l,0}^T \tilde{\mathcal{L}}_{l,n,0}^{-1} R_{l,0}$. Then our additive two-level domain decomposition preconditioner is given by

$$C_{l,n,(2),D,add}^{-1} \coloneqq C_{l,n,(1),D}^{-1} + F_{l,n}. \tag{2.11}$$

Finally, let $G_{l,n} \coloneqq I - \tilde{\mathcal{L}}_{l,n} F_{l,n}$. Then a balanced version of our two-level domain decomposition preconditioner, e.g. with further improved convergence properties, is

$$C_{l,n,(2),D,bal}^{-1} \coloneqq G_{l,n}^T C_{l,n,(1),D}^{-1} G_{l,n} + F_{l,n}. \tag{2.12}$$

Note that the choice of the number of subdomains P_l^x, and thus the number of processes used, is crucial for the parallel efficiency of the spatial solver when applied to each subproblem l of the combination method. Recall that all subproblems are solved simultaneously. For a uniform distribution of the total load, each process should get an equal amount of work. However, the number of degrees of freedom varies between different subproblem layers, i.e. different values q in the combination method (2.5) (but not within each layer). In [18] it was shown that

$$P_l^x \coloneqq \hat{P}^x \cdot 2^{d-q-1} \tag{2.13}$$

for some spatial speedup factor $\hat{P}^x \in \mathbb{N}_{>1}$ results in disjoint subdomain sizes $|\bar{\Omega}_{l,i}| \approx \frac{2^L}{\hat{P}^x}$, so the enlarged subdomain will contain $|\Omega_{l,i}| \approx (1 + 2\gamma)\frac{2^L}{\hat{P}^x}$ nodes. Note that with this particular choice, the size of the enlarged subdomains of each subproblem, i.e. the number of nodes stored on each process, is independent of dimension d and layer q, and in particular independent of subproblem l, leading to an approximately balanced use of all processes. The results of [18] show that this gives a robust, dimension-independent preconditioner with optimal order convergence behavior and excellent scalability properties. Finally note that the number of processes $\hat{P}_l^x$ of (2.13) is constant for each subproblem layer q, i.e. it does not depend on the subproblem l. However, since the grids of the (extremely) anisotropic subproblems on each layer q contain fewer grid points than the more isotropic grids on the same layer due to boundary handling, the number of grid points assigned to each spatial process can still differ. For example, consider the subproblems on layer $q = 0$ for $d = 2$ and $L = 11$. One of the extremely anisotropic grids is associated with the subproblem $l = (1, 11)$. It contains one interior node along the first dimension and 2.047 interior nodes along the second dimension, for a total of 2.047 nodes. In contrast, the isotropic grid $l = (6, 6)$ contains 63 interior grid nodes in each dimension, for a total of 3.969 nodes. However, for the domain decomposition of problems $\hat{P}^x \cdot 2^{d-q-1} = \hat{P}^x \cdot 2$ spatial processes are used in parallel, since they belong to the same layer. Therefore, the subdomains of the isotropic subproblem will be almost twice as large as those of the anisotropic subproblem. To improve this aspect, we use a subproblem-dependent choice of the number of processes. For this purpose, denote by $2^S \leq |\Omega_l|$ the target number of nodes per subdomain, i.e. we want the disjoint subdomains to be of size $|\bar{\Omega}_{l,i}| \approx 2^S$. To achieve this, we choose

$$P_l^x \coloneqq \lceil \frac{|\Omega_l|}{2^S} \rceil = \lceil \frac{\prod_{j=1}^d (2^{l_j} - 1)}{2^S} \rceil, \tag{2.14}$$

so that $|\bar{\Omega}_{l,i}| = \frac{|\Omega_l|}{P_l^x} = \frac{|\Omega_l|}{\lceil \frac{|\Omega_l|}{2^S} \rceil} \approx 2^S$. Looking again at the two subproblems above and choosing the target number of nodes as 2^{10}, i.e. $S = 10$, the subproblem $l = (1, 11)$ will now use two spatial processes, while $l = (6, 6)$ will now use four spatial processes, resulting in a balanced subdomain size of about 1.024 nodes for each subproblem.

2.2. Time-sequential combination method. In the following we describe a general approach for the time-sequential solution of parabolic problems using the combination method given in subsection 2.1.1 and the spatial subproblem solver discussed in subsection 2.1 for the discretized elliptic problems (2.6).

2.2.1. General algorithm. Let $T_{\text{start}} = \tau_0 < \tau_1 < \ldots < \tau_s = T_{\text{end}}, s \geq 1$, be a partition of the time interval of interest $[T_{\text{start}}, T_{\text{end}}]$ into $s + 1$ so-called recombination steps τ_k. Following the notation from the beginning of this section, let $T_{\text{start}} = t_{l,0} < t_{l,1} < \ldots < T_{l,N_l} = T_{\text{end}}$ be a partition of $[T_{\text{start}}, T_{\text{end}}]$ independently for each subproblem l of the combination method. Denote by $0 \leq n(l, k) \leq N_l$ the unique time step index such that $t_{l,n(l,k)} = \tau_k, 0 \leq k \leq s$. We require that $n(l, k)$ is well defined for all k and all subproblems l. This ensures that all recombination steps τ_k are present in the time partitions of all subproblems. In particular we have $n(l, 0) = 0$ and $n(l, s) = N_l$. An example of time partitions for $M = 3$ subproblems, here for simplicity indexed as $l = 1, 2, 3$, is illustrated in Figure 3.

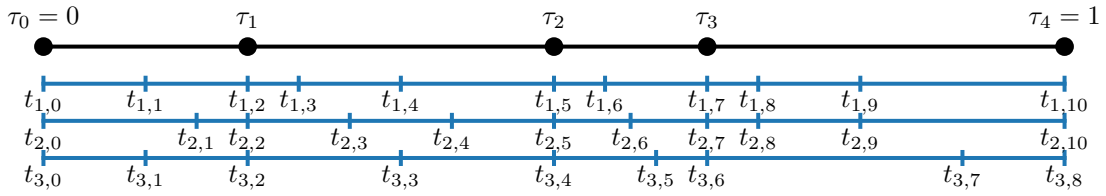

FIGURE 3. *Partition of the time interval* $[0, 1]$ *into recombination steps and an example of subproblem-dependent time steps for three subproblems.*

In the exemplary time partition in Figure 3 the time step for the subproblem $l = 1$ corresponding to the recombination step τ_2 is $t_{1,n(1,2)} = t_{1,5}$. Let $\Phi_{l,n}$ be the time propagators for the considered parabolic problem with time steps $t_{l,n}, 1 \leq n \leq N_l$, for subproblem l analogous to (2.3). Using this notation, the solution of (2.1) by the combination method in space in a time-sequential manner is given by Algorithm 2.1. A graphical illustration is given in Figure 4.

2.2.2. Combining spatial subproblem solutions. The missing ingredient for the implementation of Algorithm 2.1 is an efficient combination of all intermediate subproblem solutions $u_{l,n(l,k)}$ to obtain $u_{L,k}^{(c)}$, as required in line 2 and line 8 of Algorithm 2.1, and projections of $u_{L,k-1}^{(c)}$ onto the subproblem grids Ω_l resulting in $\bar{u}_{l,n(l,k-1)}$, as required in line 5 of Algorithm 2.1. Note that the combination of the subproblem solutions is immediately followed by a projection step from the combination solution to each subproblem in all iterations. By using a basis transformation in space to a hierarchical basis representation of the coefficients on the respective grids, we can perform the combination and the projection at once, which allows to avoid the construction of the intermediate solutions $u_{L,k}^{(c)}$ for $1 \leq k < s$ of the combination method. In the following we give a short overview, details can be found in [11, 17].

To simplify the notation we drop the time indices and assume homogeneous Dirichlet boundary

Algorithm 2.1 Time-Sequential Combination method for Parabolic Problems

Input Subproblem grids Ω_l for the combination method with sparse grid level L
Time propagators $\Phi_{l,n}$ and time steps $t_{l,n}, 1 \le n \le N_l$, for each subproblem l
Recombination steps τ_k, $0 \le k \le s$

Output Approximate sparse grid solution $u_{L,s}^{(c)}$ at recombination step $\tau_s := T_{\text{end}}$

Set $u_{l,n(l,0)}$ to the initial value $\bar{u}_0$ for all subproblems l.
Combine all $u_{l,n(l,0)}$ using (2.5) to obtain $u_{L,0}^{(c)}$. ▷ inter-subproblem comm.
for $1 \le k \le s$ **do**
  for each subproblem l simultaneously **do**
    Project $u_{L,k-1}^{(c)}$ onto Ω_l to obtain $\bar{u}_{l,n(l,k-1)}$.
    Compute $u_{l,n(l,k)}$ from $\bar{u}_{l,n(l,k-1)}$ via the time propagators $\{\Phi_{l,n(l,k-1)+1}, \dots, \Phi_{l,n(l,k)}\}$.
  end for
  Combine all $u_{l,n(l,k)}$ using (2.5) to obtain $u_{L,k}^{(c)}$. ▷ inter-subproblem comm.
end for
return $u_{L,s}^{(c)}$

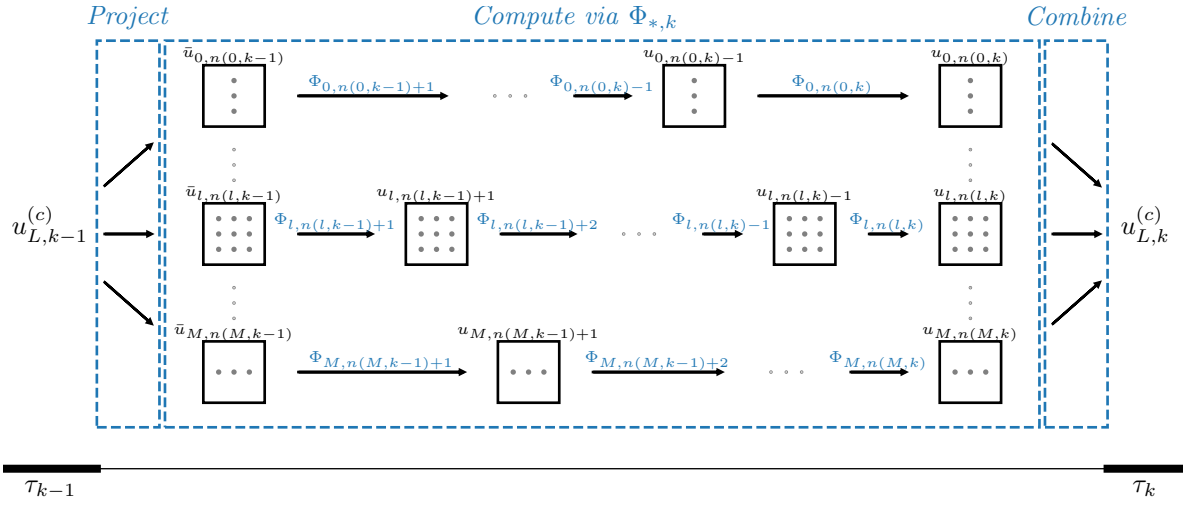

FIGURE 4. *Overview of one time propagation step of the time-sequential combination method in Algorithm* 2.1 *for various subproblems.*

conditions,[1] so that we can consider only interior basis functions. Let V_l denote the space of piecewise d-linear interior nodal basis functions associated with the subproblem grid Ω_l. Then we define by

$$W_l := V_l \setminus \oplus_{j=1}^{d} V_{l-e_j} \tag{2.15}$$

the hierarchical difference space W_l, where e_j is the unit vector in the direction of the j-th dimension. We set $V_l := 0$ if any component of l is negative. Figure 5 shows the example of a one-dimensional nodal basis for V_3 (bottom only, left) and the associated hierarchical basis for $W_1 \cup W_2 \cup W_3$ (union

[1] Inhomogeneous boundary conditions can be treated in a similar way.

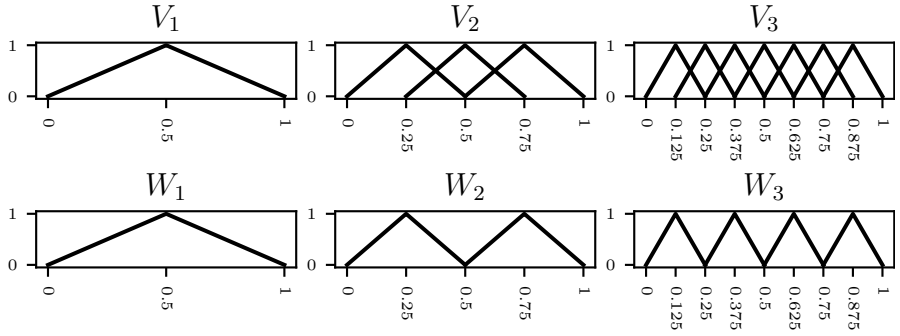

FIGURE 5. *One-dimensional nodal and hierarchical basis functions for levels* $l = 1, 2, 3$.

of the functions, right). Note that the size of the support of each nodal basis function associated with each node of V_l is 2^{-l+1} and changes with l, while it is $2^{-l'+1}$ for the functions in $W_{l'}, l' = 1, 2, l$, which altogether form the hierarchical basis. Thus, the hierarchical basis functions of coarser levels are just subsets of that on finer levels. This allows to replace the combination of subproblem solutions on the function level by a simple combination of the coefficients of their representations in the hierarchical basis via basically the same formula (2.5), i.e. just the coefficient values of the representation in the hierarchical basis of a discrete function must be properly added or subtracted.

In particular, after solving the subproblem on Ω_l in the nodal basis V_l and obtaining u_l, we hierarchize its coefficients $\tilde{u}_l \in \mathbb{R}^{|\Omega_l|}$ associated with the nodal basis to obtain coefficients $\hat{u}_l \in \mathbb{R}^{|\Omega_l|}$ associated with the hierarchical basis. These hierarchical coefficients can then be combined using (2.5) to obtain the hierarchical coefficients $\hat{\bar{u}}_l \in \mathbb{R}^{|\Omega_l|}$ of $\bar{u}_l$, the projection of $u_L^{(c)}$ onto Ω_l. A de-hierarchization of $\hat{\bar{u}}_l$ then yields the corresponding nodal coefficients $\tilde{\bar{u}}_l \in \mathbb{R}^{|\Omega_l|}$. The processes of hierarchization and de-hierarchization thus involves basically the linear transformations between the representations in the conventional nodal basis and the hierarchical basis and are given in detail in e.g. [14]. Note that they require communication between all subproblems, since, for example, the central grid point is contained in each subproblem grid. However, in view of the parallel spatial subproblem solver of subsection 2.1.2, only those processes of each subproblem that contain the same node need to exchange data, i.e. each process of each subproblem only needs to communicate with a subset of processes of other subproblems. More details can be found in [21].

Now denote by $\tilde{u}_L^{(c)} \coloneqq \text{blockdiag}_l(\tilde{u}_l) \in \mathbb{R}^{\prod_l |\Omega_l|}$ the collection of all nodal coefficients $\tilde{u}_l$ of the subproblems, analogously for $\hat{u}_L^{(c)}$. Let $H_l^{(c)} \in \mathbb{R}^{|\Omega_l| \times |\Omega_l|}$ and $H_l^{(c),-1} \in \mathbb{R}^{|\Omega_l| \times |\Omega_l|}$ be the local hierarchization and de-hierarchization operators on Ω_l, i.e. $\hat{u}_l = H_l^{(c)} \tilde{u}_l$ and $\tilde{u}_l = H_l^{(c),-1} \hat{u}_l$. Furthermore, let $H_L^{(c)} = \text{blockdiag}_l(H_l^{(c)}) \in \mathbb{R}^{(\prod_l |\Omega_l|) \times (\prod_l |\Omega_l|)}$ and $H_L^{(c),-1} = \text{blockdiag}_l(H_l^{(c),-1}) \in \mathbb{R}^{(\prod_l |\Omega_l|) \times (\prod_l |\Omega_l|)}$ denote the global hierarchization and global de-hierarchization operators such that $\hat{u}_L^{(c)} = H_L^{(c)} \tilde{u}_L^{(c)}$ and $\tilde{u}_L^{(c)} = H_L^{(c),-1} \hat{u}_L^{(c)}$. Next, denote by $S(i, \Omega_l) \coloneqq \{(j, \Omega_m) : j \in \Omega_m, x_j = x_i\}$ the nodes on all subproblem grids that correspond to the node i on Ω_l, where $x_i \in \mathbb{R}^d$ denotes the coordinates of the i-th node in Ω_l, and let $\text{layer}(\Omega_l) \coloneqq L + (d-1) - \|l\|_1$ be the layer of subproblem l. Finally, let $\bar{Q}_L$ be the combination operator representing (2.5), i.e

$$(\bar{Q}_L \tilde{u}_L^{(c)})_{(l,i)} \coloneqq \sum_{(j,\Omega_m) \in S(i,\Omega_l)} (-1)^{\text{layer}(\Omega_m)} \binom{d-1}{\text{layer}(\Omega_m)} (\tilde{u}_L^{(c)})_{(m,j)}. \tag{2.16}$$

Then, the process of combination followed by projection corresponding to lines 5 and 8 of Algorithm 2.1 is given by

$$Q_L^{(c)} \coloneqq (H^{(c)})_L^{-1} \bar{Q}_L^{(c)} H_L^{(c)}. \tag{2.17}$$

Its application to a global nodal coefficient $\tilde{\bar{u}}_L^{(c)}$ of some function $\bar{u}_L^{(c)}$, i.e. $\tilde{u}_L^{(c)} \coloneqq Q_L^{(c)} \tilde{\bar{u}}_L^{(c)}$, can be understood as follows: We first hierarchize all subproblem coefficients $\tilde{\bar{u}}_l^{(c)}$ independently of each other, i.e. we form $H_L^{(c)} \tilde{\bar{u}}_L^{(c)}$, resulting in the hierarchical coefficients $\hat{\bar{u}}_l$. We then combine the hierarchical coefficients, $Q_L^{(c)} \hat{\bar{u}}_l$, resulting in the recombined hierarchical coefficients $\hat{\tilde{u}}_l^{(c)}$. Finally, we de-hierarchize the subproblem coefficients independently of each other, i.e. we form $(H^{(c)})_L^{-1} \hat{\tilde{u}}_l^{(c)}$, obtaining the nodal coefficients $\tilde{u}_L^{(c)}$ of a function $u_L^{(c)}$, where $u_L^{(c)}$ is the recombination and projection of $\bar{u}_L^{(c)}$ to each subproblem. Now, reintroducing the time index, let $\Phi_{L,k}^{(c)} \coloneqq \mathrm{blockdiag}_l(\Phi_{l,k})$ and $\tilde{u}_k \coloneqq \mathrm{blockdiag}_l(\tilde{u}_{l,n(l,k)})$. With this, Algorithm 2.1 can equivalently be written as

$$\tilde{u}_{L,s}^{(c)} = Q_L^{(c)} \Phi_{L,s}^{(c)} Q_L^{(c)} \Phi_{L,s-1}^{(c)} Q_L^{(c)} \cdots \Phi_{L,1}^{(c)} \tilde{u}_{L,0}^{(c)}. \tag{2.18}$$

Note that there are other techniques with improved properties for combining subproblem solutions, for example via the use of biorthogonal hierarchical basis functions as in [25], or via the use of prewavelets [16]. For the present work, however, the standard hierarchical basis described above proved to be sufficient.

2.3. Multigrid reduction-in-time. Parallel-in-time techniques offer significant opportunities for parallelization by solving the PDE in space-time, using independent discretizations of space and time instead of treating each time step in a time-sequential manner. Prominent examples are the parallel-in-time preconditioners of [24], `Parareal` ([22]), and `MGRIT` ([9]). In this work we use the multigrid reduction-in-time algorithm `MGRIT` of [9] via its implementation in the library `XBraid`[2] (see [1]), for parallelizing time. `XBraid` was chosen because of its mature state and non-intrusive `C` implementation. This allows for seamless integration with our domain decomposition and sparse grid software framework implemented in `C++`.

2.3.1. Algorithmic overview. `XBraid` implements a variant of `MGRIT` based on a full approximation scheme (FAS). We will give a description of the two-level algorithm for simplicity, the general-level algorithm results by recursively applying the two-level scheme. We follow the presentation and notation of `XBraid`, [1], where details can be found. For simplicity, we assume a uniform fine time partition $T_{\text{start}} = t_{l,0} < t_{l,1} < \ldots < t_{l,N} = T_{\text{end}}$ and $\Delta t_{l,n} = \Delta t_l$ for all n. Note that `XBraid` can handle variable and adaptive time step sizes as well. We call $t_{l,n}$ a fine time step and $\Delta t_{l,n}$ a fine time step size.

Recall the discretization of (2.1) on a fixed discretization Ω_l in space given by (2.2) and its iterative solution procedure with respect to time by linear time propagators $\Phi_{l,n} : \Omega_l \to \Omega_l$ from (2.3), namely

$$\begin{aligned} u_{l,0} &= g_{l,0}, \\ u_{l,n} &= \Phi_{l,n} u_{l,n-1} + g_{l,n}, \quad 1 \le n \le N. \end{aligned} \tag{2.19}$$

[2] In this work we use the version of `XBraid` at git commit `4bbd644`.

This process is directly equivalent to the forward solution of the *space-time* system

$$B_l^{(N)}\mathbf{u}_l^{(N)} := \begin{pmatrix} \mathcal{I}_l & & & \\ -\Phi_{l,1} & \mathcal{I}_l & & \\ & \ddots & \ddots & \\ & & -\Phi_{l,N} & \mathcal{I}_l \end{pmatrix} \begin{pmatrix} u_{l,0} \\ u_{l,1} \\ \vdots \\ u_{l,N} \end{pmatrix} = \begin{pmatrix} g_{l,0} \\ g_{l,1} \\ \vdots \\ g_{l,N} \end{pmatrix} =: \mathbf{g}_l^{(N)}, \tag{2.20}$$

with the vectors $\mathbf{u}_l$ and $\mathbf{g}_l$ of length $N+1$ where each component $u_{l,n} \in \mathbb{R}^{|\Omega_l|}$ and $g_{l,n} \in \mathbb{R}^{|\Omega_l|}$ is associated to the repective grid Ω_l. The system matrix $B_l^{(N)}$ is a block matrix of size $(N+1)\times(N+1)$, with $\mathcal{I}_l : \Omega_l \to \Omega_l$ denoting the identity.

Choosing a time coarsening factor $c > 1$ allows to define a uniform coarse time partition $T_{\text{start}} = T_{l,0} < T_{l,1} < \ldots < T_{l,N_{\text{coarse}}} = T_{\text{end}}$ with $N_{\text{coarse}}+1$ coarse time steps and coarse time step size $\Delta T_{l,m'} = \Delta T_l = c\Delta t_l$, where $N_{\text{coarse}} = \frac{N}{c}$. The block operator of the space-time system for the coarse time partition of size $(N_{\text{coarse}}+1)\times(N_{\text{coarse}}+1)$ is then given by

$$B_{\text{coarse},l}^{(N)} := \begin{pmatrix} \mathcal{I}_l & & & \\ -\Phi_{\text{coarse},l,1} & \mathcal{I}_l & & \\ & \ddots & \ddots & \\ & & -\Phi_{\text{coarse},l,N_{\text{coarse}}} & \mathcal{I}_l \end{pmatrix}, \tag{2.21}$$

for appropriately chosen coarse linear time propagators $\Phi_{\text{coarse},l,m}, m = 1,\ldots,N_{\text{coarse}}$. A simple choice for $\Phi_{\text{coarse},l,m}$ are the fine-level time propagators $\Phi_{l,n}$, but using the coarse time step size $\Delta T_{l,m}$ instead of the fine partitioms $\Delta t_{l,n}$. For example, for the backward Euler scheme, the coarse operator corresponding to a fine time partition with the time step size $\Delta t_{l,n}$, i.e. $\Phi_{l,n} = (1+\Delta t_{l,n}\mathcal{L}_{l,n})^{-1}$, would be given by $\Phi_{\text{coarse},l,m} = (1+\Delta T_{l,m}\mathcal{L}_{l,m})^{-1}$

Next, we classify all fine time steps into C time steps, i.e. time steps that exist in both the coarse and fine time partitions, and F time steps, i.e. time steps that exist only in the fine time partition. This allows to define relaxation schemes and transfer operators between the two time partitions, thus completing all the necessary algorithmic ingredients for a two-level method. An example is given in Figure 6.

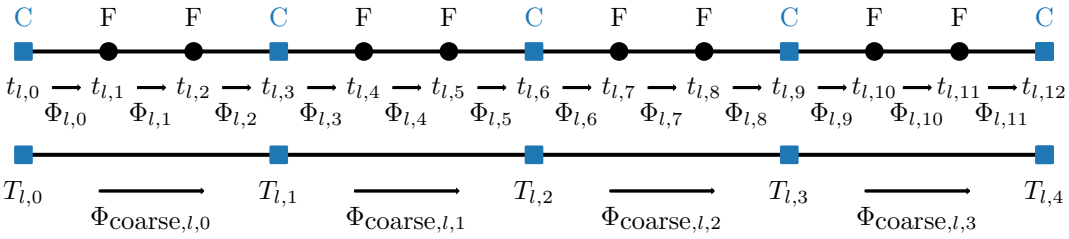

FIGURE 6. *Classification of $N = 12$ fine time steps into F and C time steps for time coarsening factor $c = 3$.*

The components for relaxation schemes available in `XBraid` are so-called F- and C-relaxation. There, F-relaxation propagates the value of $u_{l,m}$ at each C time step T_m, which is given by $u_{l,cm}$, over its corresponding coarse time interval $[T_m, T_{m+1})$ by using the sequence of fine partition time propagators $\Phi_{l,cm+1},\ldots,\Phi_{l,(c+1)m-1}$ successively to the values $u_{l,cm},\ldots,u_{l,(c+1)m-2}$ to obtain new values of u_l at all F time steps $t_{cm+1},\ldots,t_{(c+1)m-1}$ contained in $[T_m, T_{m+1})$. Note that this process can be done independently for each coarse time interval, processing each interval simultaneously, but

the propagation over F-values in each interval is inherently sequential. C-relaxation computes the value $u_{l,cm}$ at a C time step T_m from the value $u_{l,cm-1}$ at the previous F time step t_{cm-1} using the fine-level time propagator $\Phi_{l,cm}$. Again, each of these updates can be done simultaneously. Combinations of these two relaxation schemes, such as CF- or FCF-relaxation, are possible. Again, see [1] for details.

The restriction operator $R_{l,N}$ in time is given by discarding the F-values and the respective prolongation $P_{l,N}$ is given by injection followed by F-relaxation, which corresponds to harmonic interpolation in time, see [1, 9]. So we have

$$R_l^{(N)} \coloneqq \begin{pmatrix} \mathcal{I}_l & & \\ 0 & & \\ \vdots & & \\ 0 & & \\ & \mathcal{I}_l & \\ & 0 & \\ & \vdots & \\ & 0 & \\ & & \ddots \end{pmatrix}^T, \text{ and } P_l^{(N)} \coloneqq \begin{pmatrix} \mathcal{I}_l & & \\ \Phi_{l,0} & & \\ \Phi_{l,1} \circ \Phi_{l,0} & & \\ \vdots & & \\ \Phi_{l,c-2} \circ \ldots \circ \Phi_{l,0} & & \\ & \mathcal{I}_l & \\ & \Phi_{l,cm} & \\ & \Phi_{l,cm+1} \circ \Phi_{l,cm} & \\ & \vdots & \\ & \Phi_{l,c(m+1)-2} \circ \ldots \circ \Phi_{l,cm} & \\ & & \ddots \end{pmatrix}. \tag{2.22}$$

With all algorithmic components defined, the two-level variant of `MGRIT` implemented in `XBraid` is given for a spatial discretization on Ω_l by Algorithm 2.2. Here, the vectors on the coarse and fine scale are of different sizes and, of course, have their corresponding lengths.

Algorithm 2.2 Two-Level cycle of `MGRIT` for fixed Ω_l implemented in `XBraid`

1: Relax $B_l^{(N)}\mathbf{u}_l = \mathbf{g}_l^{(N)}$ using F-relaxation followed by $n_{\text{relax}} \geq 0$ applications of CF-relaxation
2: Restrict fine grid solution and residual

$$\mathbf{u}_{\text{coarse},l}^{(N)} \leftarrow R_l^{(N)}\mathbf{u}_l, \quad \mathbf{r}_{\text{coarse},l}^{(N)} \leftarrow R_l^{(N)}(\mathbf{g}_l^{(N)} - B_l^{(N)}\mathbf{u}_l^{(N)})$$

3: Solve $B_{\text{coarse},l}^{(N)}\mathbf{v}_{\text{coarse},l}^{(N)} = B_{\text{coarse},l}^{(N)}\mathbf{u}_{\text{coarse},l}^{(N)} + \mathbf{r}_{\text{coarse},l}^{(N)}$
4: Compute the coarse grid error $\mathbf{e}_{\text{coarse},l}^{(N)} \coloneqq \mathbf{v}_{\text{coarse},l}^{(N)} - \mathbf{u}_{\text{coarse},l}^{(N)}$
5: Correct fine grid solution: $\mathbf{u}_l^{(N)} \leftarrow \mathbf{u}_l^{(N)} + P_l^{(N)}\mathbf{e}_{\text{coarse}}^{(N)}$

Note that in practice the effective restriction is actually the transpose of the harmonic interpolation operator $P_l^{(N)}$, since restriction is always immediately preceded by F-relaxation. Note also that Algorithm 2.2 does not exploit the linear nature of our model problem (2.1). In line 3 one could directly solve for $\mathbf{e}_{\text{coarse},l}^{(N)}$ due to the linearity of $B_{\text{coarse},l}^{(N)}$. This would make the restriction of $\mathbf{u}_l^{(N)}$ to $\mathbf{u}_{\text{coarse},l}^{(N)}$ unnecessary. Nevertheless, we show the algorithm here as it is implemented in XBraid and presented in [1]. This more general FAS procedure allows the handling of nonlinear systems. However, this is not relevant for our work.

2.3.2. Parallelization. `XBraid` allows the distribution of fine time steps across multiple processes in an `MPI` parallel fashion. The time steps are distributed in groups, where each C time step $T_m = t_{cm}$

and all its subsequent F time steps, i.e. $\{t_{cm}, t_{cm+1}, \ldots, t_{(c+1)m-1}\}$, are on the same process. The last process also contains the last time step. An exemple distribution is shown in Figure 7.

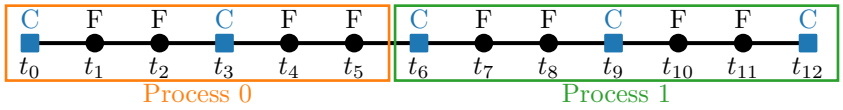

FIGURE 7. *Distribution of $N = 12$ fine time steps onto 2 processes for time coarsening factor $c = 3$.*

Due to its non-intrusive approach, `XBraid` is completely agnostic to the spatial decomposition. This means that the spatial dimension, i.e. the discretization Ω_l of Ω, can be parallelized *independently* of the temporal parallelization. In traditional sequential time-stepping schemes, however, each process involved in the spatial parallelization is responsible for one part of the spatial discretization and its associated spatial data, such as matrix and solution entries, for all time steps. These storage requirements can be reduced by noting that thae spatial data is only needed for some of the time steps, depending on the chosen time propagator. For example, the backward Euler method requires only the spatial solution vector of the last time step and all spatial data for the current time step. In contrast, the multigrid reduction-in-time approach requires the storage of all spatial data associated with the current process for all time steps, since each of the time steps can be solved repeatedly and without sequential order. Thus, the additional temporal parallelization level provided by `XBraid` ensures that each process only needs to store its part of the time partition and thus only the corresponding spatial data associated with the process. This can be seen in Figure 8 for general time propagators, possibly depending on all previous time steps. When only using time coarsening, i.e. no space-time coarsening,

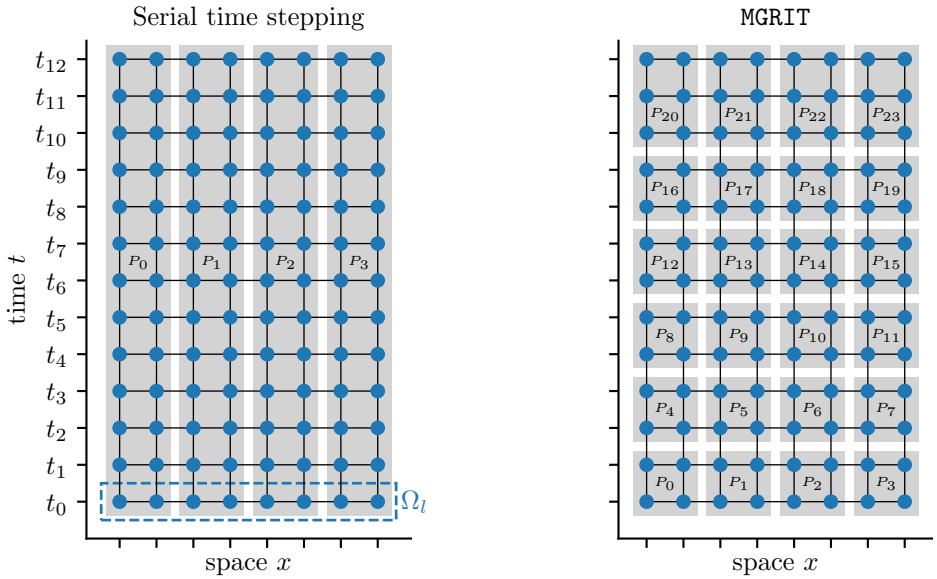

FIGURE 8. *Space-time distribution onto processes. Figure reproduced and adapted from [1].*

the required memory per process for multigrid reduction-in-time increases by a factor of $O(\log_c N)$ with a time coarsening factor of $c > 1$ compared to sequential time stepping, see [1]. In practice, `XBraid`

supports memory usage reduction by storing the solution only at the C time steps and reconstructing solutions at F time steps by F relaxation whenever required, which is used in this work.

3. A parallel-in-time combination method. In the following, we exploit the additional level of parallelization in the temporal component provided by the multigrid reduction-in-time scheme of `XBraid` in the context of the combination method and with our domain decomposition method for each of the subproblems arising there. There are two prominent approaches: On the one hand, multigrid reduction-in-time can be used at the sparse grid level, resulting in a single global parallelization of time. On the other hand, multigrid reduction-in-time can be applied within the combination method, i.e. it can be used to parallelize time for each subproblem of the combination method independently. In the following we will focus on the latter, since it allows, compared to the former, fine-grained control of the time step sizes for each subproblem mostly independently of each other with only loose coupling at the recombination step. This enables more temporal parallelism due to the fact that each subproblem can be parallelized independently. Additionally, the multigrid reduction-in-time on the subproblem level permits the use of a subproblem queue, i.e. between each of the recombination steps the subproblem solutions can be computed in any order, yielding independent compute tasks. This also allows the use of compute hardware with less compute capabilites, since the independent subproblem tasks can be managed by a job scheduling system such as SLURM ([19]). In contrast to this, multigrid reduction-in-time produces one large compute task, however with the ability to recombine at any point in time.

3.1. Multigrid reduction-in-time on the subproblems. In the following, we apply multigrid reduction-in-time to each subproblem in the sparse grid combination method individually. For this, consider Algorithm 2.1, where the time propagators $\{\Phi_{l,n(l,k-1)+1}, \dots, \Phi_{l,n(l,k)}\}$ are used to sequentially propagate the solution from recombination step τ_{k-1} to τ_k for $1 \leq k \leq s$ and all subproblems l. Since the time propagators $\Phi_{l,n}$ are independent of each other across subproblems, we can apply multigrid reduction-in-time to the time interval $(\tau_{k-1}, \tau_k]$ for each subproblem individually and simultaneously. This replaces the sequential nature of the solution scheme in time, i.e. the sequential application of $\{\Phi_{l,n(l,k-1)+1}, \dots, \Phi_{l,n(l,k)}\}$, by an additional level of parallelism. The resulting method is described in Algorithm 3.1, an example is given in Figure 9. Note also that, analogous to Algorithm 2.1, the projection and combinations in lines 5 and 8 can be performed simultaneously as described in subsection 2.2.2.

Algorithm 3.1 adds a level of parallelism at the subproblem scale: For each subproblem, the problems associated with each time interval $(\tau_{k-1}, \tau_k]$ are solved simultaneously and independently using the multigrid reduction-in-time scheme. The spatial problem can be parallelized again by the domain decomposition approach of subsection 2.1.2, resulting in a third level of parallelization. We call this approach $CTMGRIT^{loc}$ and will use it throughout our numerical experiments in section 4.

4. Numerical experiments. Now we discuss the results of our numerical experiments for the parallel $CTMGRIT^{loc}$ approach. First, we give the parameters of our solver used in the experiments and describe the high-performance computer system we used. Then we show the parallelization properties of $CTMGRIT^{loc}$ by applying it to the heat equation, the chemical master equation and some exemplary stochastic differential equations.

4.1. Solver parameters and computer system. Each of the components of $CTMGRIT^{loc}$ can be run with different parameter values and settings. Here, we do not intend to find an optimized parameter set[3] for any particular problem under consideration, but we rather settle on a parameter

[3]This is the subject of future research.

Algorithm 3.1 *CTMGRIT*^{loc}: Combination method with multigrid reduction-in-time on the subproblems

Input Subproblem grids Ω_l for the combination method with sparse grid level L
Time propagators $\Phi_{l,k}$ between recombination steps $\tau_k, 1 \leq k \leq s$

Output Approximate sparse grid solution $u_{L,s}^{(c)}$ at time step T_s

Set $u_{l,n(l,0)}$ to the initial value $\bar{u}_0$ for all subproblems l.
Combine all $u_{l,n(l,0)}$ using (2.5) to obtain $u_{L,0}^{(c)}$. ▷ inter-subproblem comm.
for $1 \leq k \leq s$ **do**
  for each subproblem l simultaneously **do**
    Project $u_{L,k-1}^{(c)}$ onto Ω_l to obtain $\bar{u}_{l,n(l,k-1)}$.
    Compute $u_{l,n(l,k)}$ from $\bar{u}_{l,n(l,k-1)}$ via Algorithm 2.2
      with the time propagators $\{\Phi_{l,n(l,k-1)+1}, \ldots, \Phi_{l,n(l,k)}\}$
    Combine all $u_{l,n(l,k)}$ using (2.5) to obtain $u_{L,k}^{(c)}$. ▷ inter-subproblem comm.
  end for
end for
return $u_{L,s}^{(c)}$

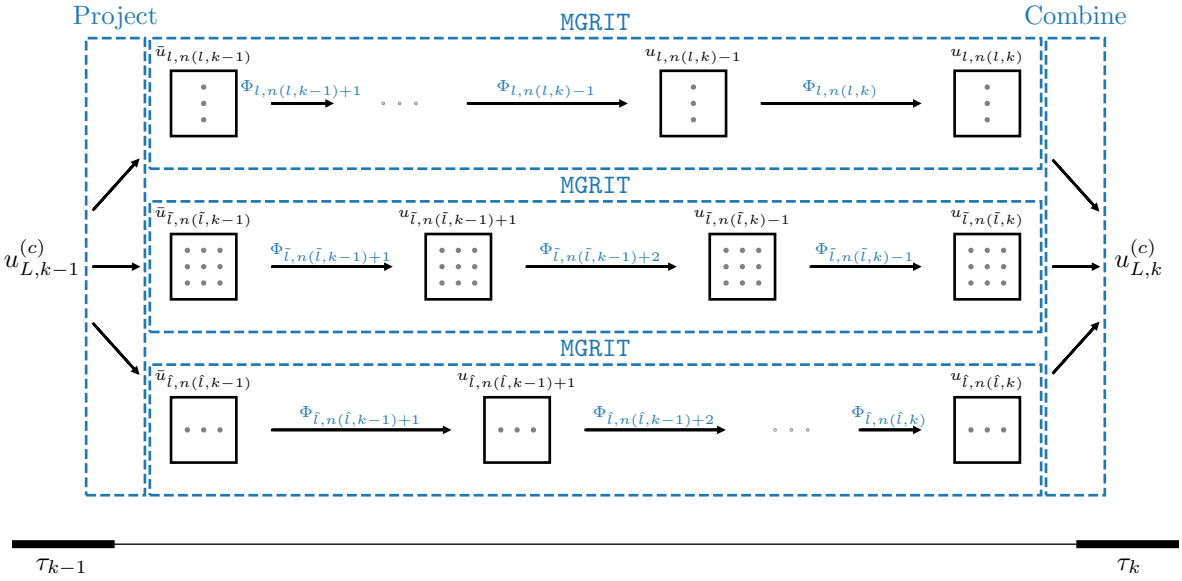

FIGURE 9. *Example of one step of the combination method with multigrid reduction-in-time on the subproblems (CTMGRIT*^{loc}*).*

set that results in good performance for all problems considered. A spatio-temporal infinity norm with tolerance 10^{-8} is used, i.e. *CTMGRIT*^{loc} terminates each `MGRIT` block whenever all spatial l_2 norms are less than 10^{-8}. We employ the two-level version of `MGRIT` with time coarsening factor $c = 2$ within *CTMGRIT*^{loc}.

For problems with a symmetric operator as in subsection 4.2, the spatial solver of each subproblem

itself is given by the conjugate gradient method preconditioned by our domain decomposition preconditioner using the balanced approach from (2.12) described in subsection 2.1.2. All spatial problems are solved up to a l_2 norm of the residual of 10^{-8}. The overlap parameter γ is chosen as $\frac{1}{2}$, the coarse grid parameters q_l are set to $q_l \equiv q \coloneqq 2^{S-4}$, where 2^S is the size of the spatial subdomains $\bar{\Omega}_{l,i}$ from (2.14). All overlapping spatial subproblems and the spatial coarse grid problem are solved by a direct solver via the LU decomposition of `MUMPS` [3, 2] through the `PETSc` solver suite [6, 4, 5]. Furthermore, S and thus P_l^x from (2.14) are chosen on a problem-by-problem basis.

The spatial solver for non-symmetric operators is given by the biconjugate gradient stabilized method, which is preconditioned by our domain decomposition method using the additive approach, where the LU decomposition is again used for the overlapping spatial subproblems and the spatial coarse grid problem. All other parameters are chosen as in the symmetric case. The advantage of the additive approach in the non-symmetric setting is the reduction of memory by a factor of two as well as a reduction of communication, since the transposed operator does not have to be formed and stored. In the symmetric setting, the transposed operator is simply given by the operator itself, so there is no additional cost. A comparison of the performance of the additive and the balanced approach for non-symmetric problems will be future work.

All computations have been run on the *Yuma* cluster of the Institut für Numerische Simulation of the University of Bonn. It provides, among others, 45 nodes, each equipped with two AMP EPYC 9435 32-Core processors and 768 GiB of physical memory. The cluster employs an InfiniBand interconnect with 200 Gb/s throughput. In the following numerical experiments we employ a one-to-one mapping of processes to cores. Therefore, each subproblem of $CTMGRIT^{loc}$, which corresponds to a single compute job on the cluster, can by parallelized by at most $45 \cdot 2 \cdot 32 = 2880$ processes. Hence, our our choice of speedup factors is constrained by the hardware limitations to $\hat{P}_l^t P_l^x \leq 2880$. Note however that this is only a limitation for the parallelization of each subproblem. Since the subproblems are independent of each other, the corresponding compute jobs can be processed in any order, for example via the SLURM job scheduling system [19] employed on *Yuma*.

4.2. The heat equation in higher dimensions. We first demonstrate the scalability properties of $CTMGRIT^{loc}$ described in subsection 3.1. To do this, we consider the standard heat equation

$$\begin{aligned} \frac{\partial u}{\partial t} - \Delta u &= f && \text{in } \Omega \times (0,T], \\ u &= 0 && \text{on } \partial\Omega \times (0,T], \\ u &= u^* && \text{in } \Omega \times \{0\}, \end{aligned} \tag{4.1}$$

where $\Omega = [0,1]^d$ for $d = 2, \dots, 6$, $T = 1$ and $u^*(x,t) = \sqrt{\|x\|_2^2 + t^2} e^{-t} \prod_{i=1}^d \sin(\pi x_i)$. Here, f is chosen so that u^* is the exact solution.

We solve (4.1) by $CTMGRIT^{loc}$ on the sparse grid level L using $s+1$ recombination steps $\tau_k = k\frac{T}{s}, 0 \leq k \leq s$, and a uniform time partition $t_{l,n} = t_n = n\frac{T}{N}, 0 \leq n \leq N_l = N$ over all subproblems l, where N is chosen as an integer multiple of $10 \cdot s$, i.e. $N \coloneqq \hat{N}_t \cdot 10 \cdot s, \hat{N}_t \geq 1$, so that between every two recombination steps, $10\hat{N}_t$ time steps are computed in parallel via `MGRIT` for each subproblem. The overlap factor γ is set to $\gamma = 0.5$ and the number of processes for each subproblem P_l^x is chosen according to (2.14) with $S = 10$ so that each subdomain $\Omega_{l,i}$ of each subproblem l, and thus each processor, contains approximately $(1+2\gamma)2^S = 2 \cdot 2^{10} = 2^{11}$ grid points, see subsection 2.1.2. The maximum sparse grid level $L_{\max,d}$ in each dimension d was set to $L_{\max,d} = 19 - d$. The speedup factor in time $\hat{P}^t$ is set to $\hat{P}^t = \hat{N}_t$, which ensures that each processor is responsible for storing the spatial data associated with its grid nodes for 10 time steps. Overall, this results in a balanced distribution

of data across all processes involved. The maximum number of processes used for these choices can be computed according to **??** and is given in Table 1.

TABLE 1
Number of processes used by $CTMGRIT^{loc}$ for $L = 19 - d$, $S = 10$ and $\hat{P}^t = \hat{N}_t$.

$d =$	2	3	4	5	6
	$5636 \cdot \hat{N}_t$	$38.966 \cdot \hat{N}_t$	$136.452 \cdot \hat{N}_t$	$292.727 \cdot \hat{N}_t$	$425.610 \cdot \hat{N}_t$

In the following we call the solution procedures between recombination steps for the subproblem l, i.e. line 7, the space-time solution procedure for the subproblem l. Figure 10 shows the median number of iterations for the space-time solution procedure across all subproblems. We can clearly observe an optimal scaling behavior of $CTMGRIT^{loc}$, independent of dimension d, global level L, subproblems l and therefore anisotropic subproblem grids.

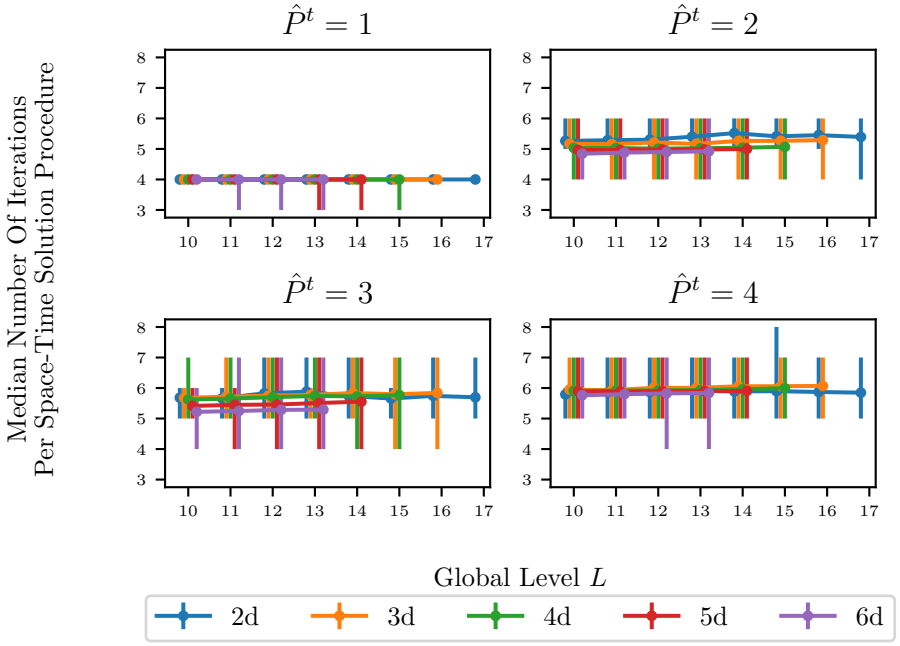

FIGURE 10. *Median number of iterations for all* **MGRIT** *solves for $CTMGRIT^{loc}$ for various choices of d and $\hat{P}^t$.*

Figure 11 shows the median runtime for the space-time solution procedure over all subproblems. In view of the scaling experiments for the spatial solver within the combination method performed in [18], we can now observe that the scaling behavior of $CTMGRIT^{loc}$ is controlled by the scaling behavior of the spatial solver, since it shows qualitatively very similar runtimes for each subproblem. The significantly faster space-time solution procedure for $d = 2$ and $L = 10$ is due to the fact that, for this combination of parameters, each spatial subproblem is distributed over at most two processes. In this particular case, since the spatial subproblems are sufficiently small and the overlap factor was chosen to be $\gamma = 0.5$, the spatial domain decomposition solver is essentially a direct solver, since the

full problem is available for each process. This allows the spatial subproblem to be solved in very few iterations.

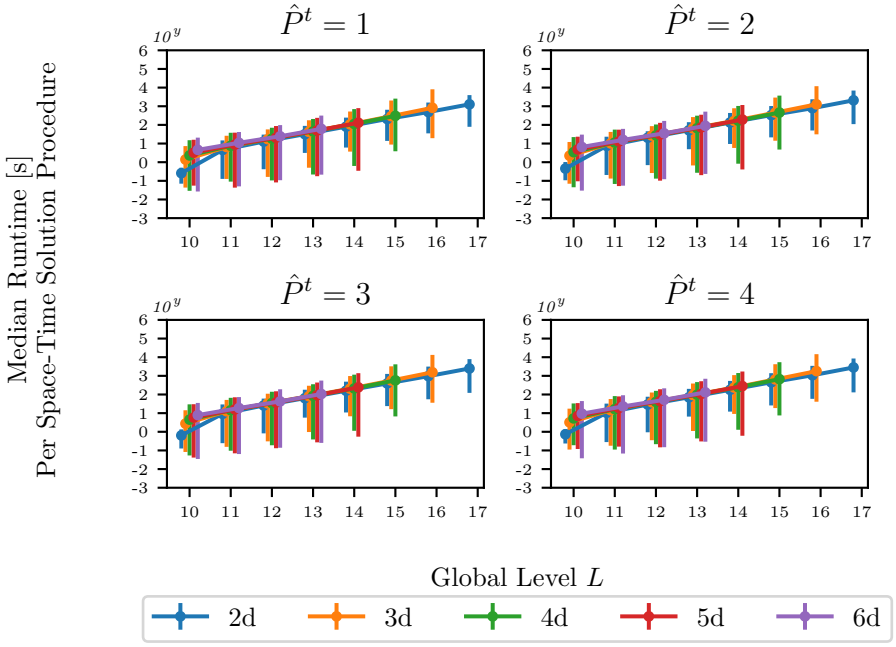

FIGURE 11. *Median runtime for all* `MGRIT` *solves for* $CTMGRIT^{loc}$ *for various choices of* d *and* $\hat{P}^t$.

4.3. An approximation to chemical and biological reaction networks. Our next problem is given by the so-called chemical master equation (CME). It models the evolution of the probability distribution of the configuration of a reaction network, where in particular a configuration describes the number of molecules of different species, such as chemical reactants, undergoing reactions. Examples range from classical stochastic chemical kinetics or gene expression in a population of cells to epidemiological models.

We follow the presentation in [23]. Consider a biochemical system with d species $\{S_1, \dots, S_d\}$. Let $\nu_r \in \mathbb{Z}^d$ and $\alpha_r \geq 0$ denote the stoichiometric vector and the propensity for each reaction $1 \leq r \leq m$ between these species. In particular, the reaction r can be written as

$$a_1 S_1 + \ldots + a_d S_d \xrightarrow{\alpha_r(x)} b_1 S_1 + \ldots + b_d S_d, \tag{4.2}$$

i.e. the current state $x = (x_1, \dots, x_d) \in \mathbb{N}^d_{\geq 0}$ is changed by the reaction r to the state $x + \nu_r$ with the probability per unit time given by $\alpha_r(x)$, where $\nu_r = (b_1 - a_1, \dots, b_d - a_d)$. It is well known that this process can be modeled by a Markov chain on the integer lattice whose probability density function satisfies the chemical master equation

$$\frac{\partial u(x,t)}{\partial t} = \sum_{r=1}^{m} \alpha_r(x - \nu_r) u(x - \nu_r, t) - \alpha_r(x) u(x,t), \tag{4.3}$$

TABLE 2
The two-dimensional genetic toggle switch model as described.

reaction			propensity			rate
A	$\to$	2A	α_1	$=$	$c_1/(c_2 + [B]^\beta)$	$c_1 = 3 \cdot 10^3, c_2 = 1.1 \cdot 10^4, \beta = 2$
2A	$\to$	A	α_2	$=$	$c_3[A]$	$c_3 = 10^{-3}$
B	$\to$	2B	α_3	$=$	$c_4/(c_5 + [A]^\gamma)$	$c_4 = 3 \cdot 10^3, c_5 = 1.1 \cdot 10^4, \gamma = 2$
2B	$\to$	B	α_4	$=$	$c_6[B]$	$c_6 = 10^{-3}$

for some initial condition $u(x, 0) = u_0(x)$. Via an embedding of the integer lattice in $\Omega \subset \mathbb{R}^d$ and the Kramers-Moyal expansion, a Fokker-Planck-type approximation to the chemical master equation (4.3) can be written as

$$\frac{\partial u(x,t)}{\partial t} = \sum_{r=1}^{m} -\nu_r^T \nabla \left[\alpha_r(x) u(x,t)\right] + \frac{1}{2}\nu_r^T (\nabla \otimes \nabla \left[\alpha_r(x) u(x,t)\right])\nu_r, \tag{4.4}$$

which matches our parabolic problem (2.1). By making the computational domain Ω sufficiently large such that the mass is far from the boundary, we can assume homogeneous Dirichlet boundary conditions on $\partial\Omega$.

4.3.1. A genetic toggle switch in two dimensions. As an example we consider a genetic toggle switch. It models two competing repressors A and B, which are transcribed by two constitutive promoters and can each inhibit the production of the other repressor. An example for this is the genetic toggle switch in Escherichia coli in [12]. The model exhibits a bistable stationary distribution and has been used extensively in the literature to test numerical schemes for the chemical master equation, see [20, 8, 26] among others. Due to its multi-stability, the convergence of traditional stochastical schemes such as the Gillespie algorithm [13] (also called SSA) can be slow to approximate stationary distributions [20]. In this context, deterministic numerical schemes based on the chemical master equation such as our proposed method are clearly advantageous. The four reactions and their propensities used in the following are given in Table 2, where the species $\{S_1, S_2\}$ and the current state (x_1, x_2) have been relabeled to $\{A, B\}$ and $([A], [B])$ for better readability. Following [20] we use a computational domain of $[0, 399]^2$ and let the initial condition u_0 be given by the probability densitiy function of a two-dimensional normal distribution with mean $M = \begin{bmatrix}133 & 133\end{bmatrix}^T$ and variance $C = \text{diag}(133)$. We run $CTMGRIT^{loc}$ on level $L = 13$ with initial level $L_0 = 6$, $S = 10$ and two recombination steps, i.e. $s = 1$. We set $\hat{P}^t = 100$ and compute $N = 100000$ time steps, such that each process handles 1000 time steps. In total $CTMGRIT^{loc}$ consists of five independent spatial-temporal subproblems, three of which utilize 1600 processes, the other 800 processes, for a total number of 6400 processes. Figure 12 shows the solution at time $t = 10^5 s$ of $CTMGRIT^{loc}$ as well as the probability density function extracted from 10^8 trajectories generated by *SSA*. We observe that $CTMGRIT^{loc}$ produces a smooth probability density, whereas the probability density extracted from the trajectories of *SSA* has jumps, since *SSA* is an integer-valued scheme. These jumps can only be smoothed by significantly increasing the number of sampled trajectories.

4.3.2. A genetic toggle switch in three dimensions. [20] proposed an extension of the bistable toggle switch to a tristable toggle switch in three dimensions. Table 3 shows the reactions, where we have relabeled the species as A, B and C. The computational domain is given by $[0, 199]^3$ and we use as the initial condition, analogously to the two dimensional problem, a three-dimensional Gaussian distribution centered at $\begin{bmatrix}133, 133, 133\end{bmatrix}^T$ with variance $\text{diag}(133)$. $CTMGRIT^{loc}$ is run on

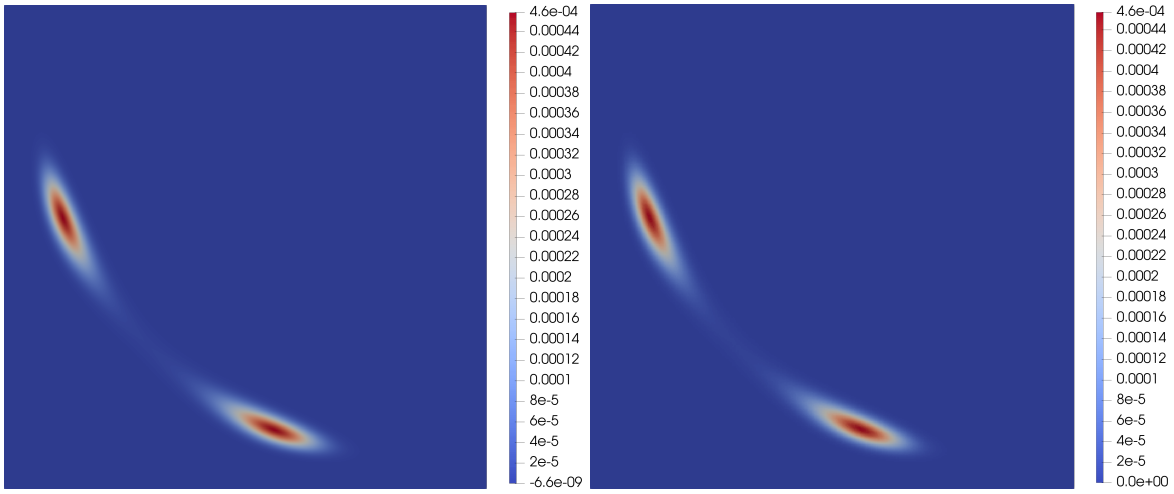

FIGURE 12. *Approximate solution for the two-dimensional toggle switch computed via $CTMGRIT^{loc}$ (left) and SSA (right) at time $t = 10^5 s$.*

TABLE 3
The three-dimensional genetic toggle switch model as described.

reaction			propensity			rate
A	$\to$	2A	α_1	$=$	$c_1/(c_2 + ([B] + [C])^\beta)$	$c_1 = 3 \cdot 10^3, c_2 = 1.1 \cdot 10^4, \beta = 2$
2A	$\to$	A	α_2	$=$	$c_3[A]$	$c_3 = 10^{-3}$
B	$\to$	2B	α_3	$=$	$c_4/(c_5 + ([A] + [C])^\gamma)$	$c_4 = 3 \cdot 10^3, c_5 = 1.1 \cdot 10^4, \gamma = 2$
2B	$\to$	B	α_4	$=$	$c_6[B]$	$c_6 = 10^{-3}$
C	$\to$	2C	α_5	$=$	$c_7/(c_8 + ([A] + [B])^\xi)$	$c_7 = 3 \cdot 10^3, c_8 = 1.1 \cdot 10^4, \xi = 2$
2C	$\to$	C	α_6	$=$	$c_9[C]$	$c_9 = 10^{-3}$

level $L = 20$ with initial level $L_0 = 7$, $S = 15$ and two recombination steps, i.e. $s = 1$. We set $\dot{P}^t = 10$ and compute $N = 10000$ time steps, such that each process handles 1000 time steps. In total $CTMGRIT^{loc}$ consists of four independent spatial-temporal subproblems, three of which utilize 1260 processes, the other 630 processes, for a total number of 4410 processes. All other parameters are the same as in the two-dimensional case. Figure 13 shows the solutions at $t = 10^5 s$ of both $CTMGRIT^{loc}$ and *SSA* with 10^8 trajectories. We notice that $CTMGRIT^{loc}$ again produces a significantly more resolved probability density function. Note that the solution for $CTMGRIT^{loc}$ is sampled on a 300^3 full grid due to the fact that it is not feasible to store the equivalent full grid on level 20 in memory due to its size.

4.4. Stochastic differential equations. Next, we consider stochastic differential equations of the form

$$dX_t = \theta X_t \, dt + \sigma \, dW_t \tag{4.5}$$

where X_t is a $\mathbb{R}^d$-valued stochastic process, $\theta \in \mathbb{R}^{d \times d}$, $\sigma \in \mathbb{R}^{d \times m}$, and W_t is an m-dimensional Gaussian white noise process with zero mean and autocorrelation $\mathbb{E}[W_{t+\tau}\overline{W_t}] = 2D\delta(\tau)$ with $D \in \mathbb{R}^{m \times m}$.

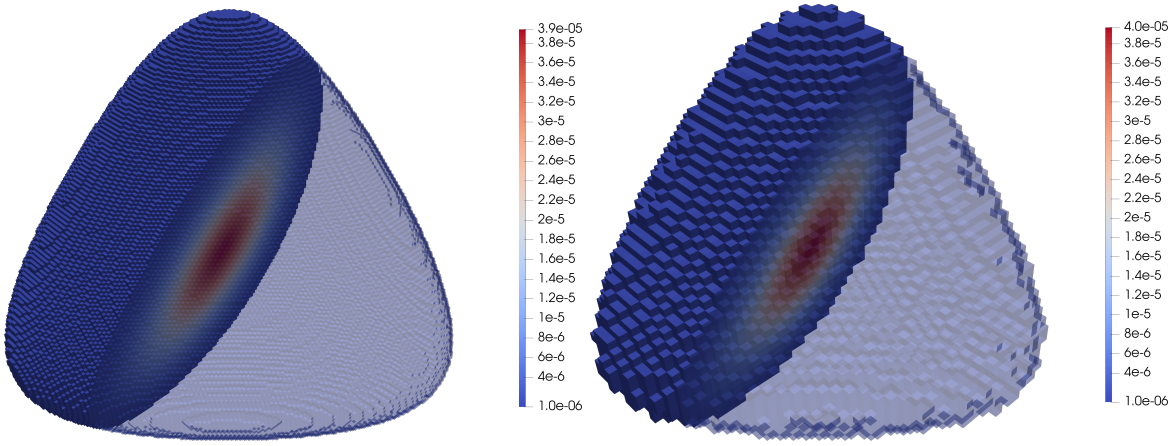

FIGURE 13. *Approximate solution for the three-dimensional toggle switch computed via CTMGRIT^{loc} (left) and SSA (right) at time $t = 10^5 s$. The solution for CTMGRIT^{loc} is sampled on a 300^3 full grid.*

The probability density function $u(x,t)$ of the Markov process X_t can be computed using the Fokker-Planck equation in $\mathbb{R}^d$

$$\frac{\partial u}{\partial t} = -\sum_{i=1}^{d} \frac{\partial}{\partial x_i} \left[(\theta x)_i u\right] + \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j} \left[H_{ij} u\right], \tag{4.6}$$

with $H = 2\sigma D \sigma^T$, boundary conditions $u \to 0$ as $\|x\| \to \infty$ and a suitable initial condition $p(x,0) = p_0(x)$. The formal solution of (4.5) is well known. It is given by

$$X_t = e^{\theta t} X_0 + \int_0^t e^{\theta(t-s)} \sigma \, dW_s. \tag{4.7}$$

and has a d-dimensional normal distribution $\mathcal{N}(e^{\sigma t} X_0, \operatorname{cov}(X_0, X_0) + \int_0^t e^{\theta s} H e^{\theta^T s} \, ds)$.

4.4.1. A two-dimensional linear oscillator. A first example for (4.6) is taken from [27]. It models the response of a two-dimensional linear oscillator subject to additive Gaussian white noise. We consider, in the notation of (4.5) and (4.6), the system given by

$$\theta := \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & -2\xi\omega_0 \end{bmatrix}, \quad \sigma := \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad D = \begin{bmatrix} 0.1 \end{bmatrix}, \tag{4.8}$$

where $\xi = 0.05$ and $\omega_0 = 1$. The system describes a mass-spring-damper model with a one-dimensional mass connected to a fixed point via a spring and a damper. An excitation W_t is applied to the mass. The spatial dimensions of the system correspond to the position (x_1) and velocity (x_2) of the mass. We choose a two-dimensional normal distribution with mean $M = \begin{bmatrix} 5 & 5 \end{bmatrix}^T$ and covariance $C = \operatorname{diag}(\frac{1}{9})$ as initial condition. Using (4.7) and the fact that normal distributions with normal mean are normal, the analytical solution is given by a Gaussian process with mean $e^{\theta t} M$ and covariance $e^{\theta t} C e^{\theta^T t} + \int_0^t e^{\theta s} H e^{\theta^T s} \, ds$.

Since (4.6) is a simple two-dimensional problem, it can be solved with sufficient accuracy using a classical full-grid approach using a second order finite difference discretization. The discretization

and parameters of the full-grid approach are chosen analogously to those of the subproblems of the combination method. The full-grid resolution is chosen such that the number of degrees of freedom is approximately the same as the sum of the degrees of freedom of all subproblems in the combination method. This serves as a means to validate $CTMGRIT^{loc}$ not only against an analytical solution (4.7), but also against a reference solution derived from a classical scheme. In all cases we consider the computational domain $[-10, 10]^2$. The solution of the combination method has been computed on the sparse grid level $L = 14$ with initial level $L_0 = 6$, i.e. $l \geq 6$ component-wise for all subproblems l. The full grid schemes employ a grid of size 421×421 such that the number of degrees of freedom approximately equals the total number of degrees of freedom across all subproblems of $CTMGRIT^{loc}$. We run the simulation until time $T_{\text{end}} = 100 \approx 16\frac{2\pi}{\omega_0}$, which corresponds to about 16 rotations around the origin in phase space. We employ the backward Euler method as the time propagator $\Phi_{l,n}$ on the fine time partition with a uniform time step size $\Delta t = 0.005$. The time coarsening factor is chosen as $c = 2$ such that $\Delta T = 0.01$. We compute the solution using the full grid with sequential time stepping (FGS), the full grid with `XBraid` for time stepping (FGX), and $CTMGRIT^{loc}$ with two recombination steps, i.e. $s = 1$. For all three schemes we use our spatial domain decomposition solver with the parameters described in subsection 4.1. We set $S = 10$ and $q = 6$ such that each process stores about 2.048 fine and 64 coarse spatial degrees of freedom. For the `XBraid` variants we use a temporal speedup factor $\hat{P}^t = 10$ so that each process stores the spatial degrees of freedom for 1.000 time steps. Figure 14 shows the value of u at the origin over time for the different schemes and the error compared to the analytical solution (4.7). All schemes produce values comparable to the analytical solution.

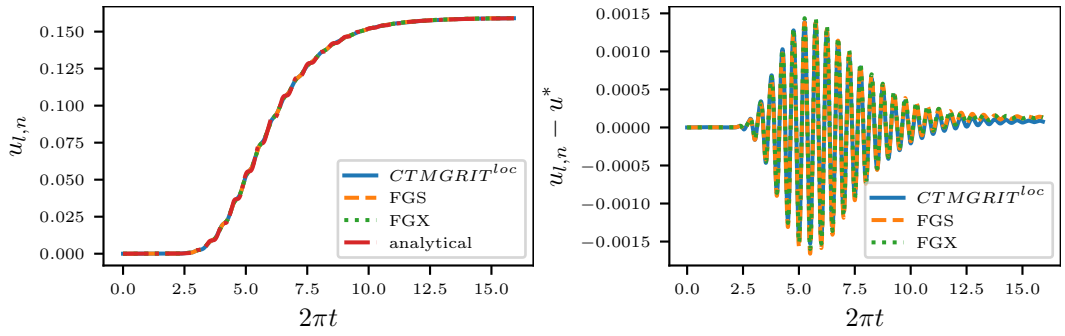

FIGURE 14. *Approximate solution (left) and error (right) at the spatial origin over time of the two-dimensional linear oscillator for $CTMGRIT^{loc}$, FGS and FGX compared to the analytical solution.*

Table 4 shows the runtimes, total number of processes and maximum error over time on the spatial diagonal for each of the schemes. Note that the runtime is averaged over five repetitions. We can see that all schemes produce comparable solutions, where our $CTMGRIT^{loc}$ is both the most accurate and fastest. Note that even though FGS utilizes no temporal speedup, i.e. the number of processes is reduced by a factor of 10, it only has a slightly slower runtime compared to $CTMGRIT^{loc}$. This is due to the fact that the size of the problem at hand is near the crossover point where the performance benefits of using a temporal parallelization via multigrid in time offsets the incurred overheads of such an approach. Additionally, there is an optimal number of spatial processes for the full grid scheme FGS beyond which the performance starts to deteriorate again due to communication overheads. This can also be seen by running FGS with 1760 spatial processes, the same number as total processes for $CTMGRIT^{loc}$, where we observed an average runtime of 61890 seconds for FGS. Therefore $CTMGRIT^{loc}$, or more generally multigrid in time schemes, allow to increase the performance

beyond what is possible with a purely spatial parallelization scheme such as *FGS*. Optimizing the parameters of each of the schemes to obtain the optimal performance is beyond the scope of this paper. We refer to, amongst others, [9] for a parameter study in the context of multigrid in time.

TABLE 4
Average, minimum and maximum runtimes per problem, total number of processes used and maximum error over time for the two-dimensional linear oscillator

	run time [s]	number of processes	max. error over time
$CTMGRIT^{loc}$	10385 (6362/14744)	$1760 = 10 \cdot (4 \cdot 32 + 3 \cdot 16)$	0.00085
FGS	11595 (8656/14599)	176	0.00098
FGX	66449 (66410/66489)	$1760 = 10 \cdot 176$	0.00096

4.4.2. A four-dimensional linear system. The second example of a stochastic differential equation (4.5) is taken from [28]. It is given by

$$\theta = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -(k_1+k_2) & -c1 & k2 & 0 \\ 0 & 0 & 0 & 1 \\ k_2 & 0 & -(k_2+k_3) & -c_2 \end{bmatrix}, \quad \sigma = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 2D_1 & 0 \\ 0 & 2D_2 \end{bmatrix}, \tag{4.9}$$

where $k_1 = k_2 = k_3 = 1, c_1 = c_2 = 0.4$ and $D_1 = D_2 = 0.2$. The system describes a mass-spring-damper model with two one-dimensional masses. Each mass is connected to a separate fixed point via a spring and a damper. Additionally, the masses are connected via a third spring. An excitation given by W_t is applied to both masses. The four dimensions of the system correspond to the positions (x_1, x_3) and velocities (x_2, x_4) of the masses. The chosen initial condition is a zero mean Gaussian distribution with variance $C = \mathrm{diag}(0.5)$. Analogous to subsection 4.4.1, the analytical solution can be derived from the initial distribution and (4.7). We compute the solution for $CTMGRIT^{loc}$ using the sparse grid level $L = 20$ with the initial level $L_0 = 5$ for the computational domain $[-6, 6]^4$. We run the simulation for 5.000 time steps until time $T_{\text{end}} = 20$. Here, a local domain size of $S = 16$ and a local coarse grid size of $q = 12$ per process was used. Figure 15 shows a comparison of the computed solution with the analytical solution. We note again a good agreement between the computed and the analytical solution, with the relative error mostly below five percent over time. The increase in the error compared to the two dimensional case can be attributed to the spatial resolution, which due to the hardware limitations and the choice of temporal speedup factor $\hat{P}_l^t = 5$ is constrained to be small enough to fit on at most $P_l^x \leq 576$ spatial processes. Hence, the initial level $L_0 = 5$ has been chosen one level coarser than in the two dimensional case. We expect, in accordance to the theory, that the four dimensional case with an initial level $L_0 = 6$ will exhibit the same errors as the two dimensional case, due to the fact that increasing the level by one halves the grid spacing, which quarters the error of the computed solution.

5. Concluding remarks. In this article, we combined `XBraid`'s `MGRIT` parallel-in-time integrator [1] with our sparse grid combination approach [17] and our space-filling curve based domain decomposition linear solver [18]. The resulting overall solver for parabolic problems thus exhibits multiple levels of parallelism: a large number of independent (i.e. embarrassingly parallel) subproblems of the combination technique in the spatial dimensions, the parallel-in-time parallelism in the time dimension for each independent parabolic subproblem, and finally a domain decomposition based parallel linear solver for the remaining elliptic part of the subproblems. Therefore, the presented approach

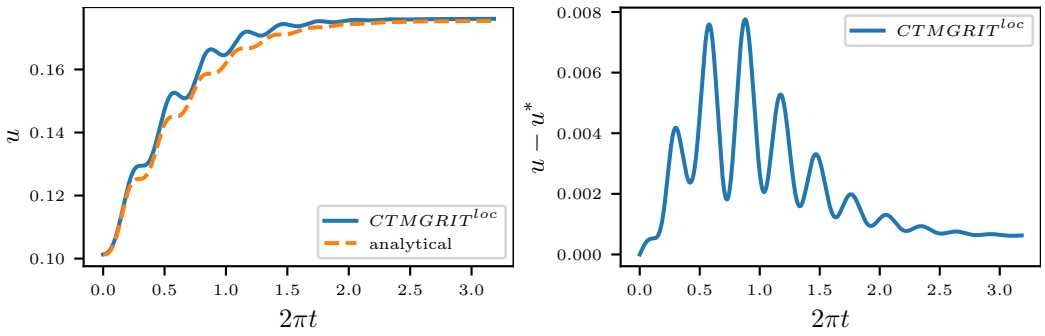

FIGURE 15. *Approximate solution (left) and error (right) at the spatial origin over time of the four-dimensional linear oscillator for* $CTMGRIT^{loc}$.

can utilize a huge number of cores simultaneously where only a moderate number of cores participate in the parallel solution of the considered parabolic problem using the MGRIT time integrator on the respective combination technique subproblem (which employs an anisotropic grid in space). Thus, the proposed approach can be employed in a load balanced way on very large supercomputers and on moderate-sized clusters for the same problem, i.e. the global requirements on hardware ressources are rather small while on much larger hardware, if available, resources can still be efficiently utilized with near perfect speedup. The results of our numerical experiments clearly show these excellent speedup and scale-up properties of the presented approach in up to six spatial and one temporal dimension.

Acknowledgments. Michael Griebel was supported by the *Hausdorff Center for Mathematics* (HCM) in Bonn, funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2047/1 – 390685813 of the Deutsche Forschungsgemeinschaft.

REFERENCES

[1] *XBraid: Parallel Multigrid in Time*. http://llnl.gov/casc/xbraid.

[2] P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary, *Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures*, ACM Transactions on Mathematical Software, 45 (2019), pp. 2:1–2:26, https://doi.org/10.1145/3242094.

[3] P. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent, *A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling*, SIAM Journal on Matrix Analysis and Applications, 23 (2001), pp. 15–41, https://doi.org/10.1137/s0895479899358194.

[4] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, and H. Zhang, *PETSc Users Manual*, Tech. Report ANL-95/11 - Revision 3.6, 2015, http://www.mcs.anl.gov/petsc.

[5] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, and H. Zhang, *PETSc Web Page*. http://www.mcs.anl.gov/petsc, 2015, http://www.mcs.anl.gov/petsc.

[6] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *Efficient Management of Parallelism in Object Oriented Numerical Software Libraries*, in Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser Press, 1997, pp. 163–202, https://doi.org/10.1007/978-1-4612-1986-6_8.

[7] H.-J. Bungartz and M. Griebel, *Sparse Grids*, Acta Numerica, 13 (2004), pp. 147–269, https://doi.org/10.1017/cbo9780511569975.003.

[8] P. Deuflhard, W. Huisinga, T. Jahnke, and M. Wulkow, *Adaptive Discrete Galerkin Methods Applied to the Chemical Master Equation*, SIAM Journal on Scientific Computing, 30 (2008), pp. 2990–3011, https://doi.org/10.1137/070689759.

[9] R. Falgout, S. Friedhoff, T. Kolev, S. MacLachlan, and J. Schroder, *Parallel Time Integration with Multigrid*, SIAM Journal on Scientific Computing, 36 (2014), pp. C635–C661, https://doi.org/10.1002/pamm.201410456.

[10] M. Ganter and T. Lunet, *Time Parallel Time Integration*, vol. 99 of CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, 2024, https://doi.org/10.1137/1.9781611978025.

[11] J. Garcke, *Sparse Grids in a Nutshell*, in Sparse grids and applications, J. Garcke and M. Griebel, eds., vol. 88 of Lecture Notes in Computational Science and Engineering, Springer, 2013, pp. 57–80.

[12] T. S. Gardner, C. R. Cantor, and J. J. Collins, *Construction of a Genetic Toggle Switch in Escherichia Coli*, Nature, 403 (2000), pp. 339–342, https://doi.org/10.1038/35002131.

[13] D. Gillespie, *Exact Stochastic Simulation of Coupled Chemical Reactions*, The Journal of Physical Chemistry, 81 (1977), pp. 2340–2361, https://doi.org/10.1021/j100540a008.

[14] M. Griebel, *Adaptive Sparse Grid Multilevel Methods for Elliptic PDEs Based on Finite Differences*, Computing, 61 (1998), pp. 151–179, https://doi.org/10.1007/bf02684411.

[15] M. Griebel and H. Harbrecht, *On the Convergence of the Combination Technique*, in Sparse grids and Applications, vol. 97 of Lecture Notes in Computational Science and Engineering, Springer, 2014, pp. 55–74, https://doi.org/10.1007/978-3-319-04537-5_3.

[16] M. Griebel and P. Oswald, *Tensor Product Type Subspace Splitting and Multilevel Iterative Methods for Anisotropic Problems*, Adv. Comput. Math., 4 (1995), pp. 171–206, https://doi.org/10.1007/bf02123478.

[17] M. Griebel, M. Schneider, and C. Zenger, *A Combination Technique for the Solution of Sparse Grid Problems*, in Iterative Methods in Linear Algebra, P. de Groen and R. Beauwens, eds., IMACS, Elsevier, North Holland, 1992, pp. 263–281.

[18] M. Griebel, M. Schweitzer, and L. Troska, *A Dimension-Oblivious Domain Decomposition Method Based on Space-Filling Curves*, SIAM Journal on Scientific Computing, 45 (2023), pp. A369–A396, https://doi.org/10.1137/21m1454481.

[19] M. Jette, C. Dunlap, J. Garlick, and M. Grondona, *Slurm: Simple Linux Utility for Resource Management*, (2002), https://doi.org/10.1007/10968987_3, https://www.osti.gov/biblio/15002962.

[20] I. Kryven, S. Röblitz, and C. Schütte, *Solution of the Chemical Master Equation by Radial Basis Functions Approximation with Interface Tracking*, BMC Systems Biology, 9 (2015), https://doi.org/10.1186/s12918-015-0210-y.

[21] R. Lago, M. Obersteiner, T. Pollinger, J. Rentrop, H.-J. Bungartz, T. Dannert, M. Griebel, F. Jenko, and D. Pflüger, *EXAHD: A Massively Parallel Fault Tolerant Sparse Grid Approach for High-Dimensional Turbulent Plasma Simulations*, in Software for Exascale Computing - SPPEXA 2016-2019, H.-J. Bungartz, S. Reiz, B. Uekermann, P. Neumann, and W. Nagel, eds., Cham, 2020, Springer International Publishing, pp. 301–329, https://doi.org/10.1007/978-3-030-47956-5_11.

[22] J.-L. Lions, Y. Maday, and G. Turinici, *A "parareal" in Time Discretization of Pde's*, Comptes Rendus de l'Académie des Sciences. Série I. Mathématique, 332 (2001).

[23] D. Lunz, G. Batt, J. Ruess, and J. F. Bonnans, *Beyond the Chemical Master Equation: Stochastic Chemical Kinetics Coupled with Auxiliary Processes*, PLOS Computational Biology, 17 (2021), p. e1009214, https://doi.org/10.1371/journal.pcbi.1009214.

[24] M. Neumüller and I. Smears, *Time-Parallel Iterative Solvers for Parabolic Evolution Equations*, SIAM Journal on Scientific Computing, 41 (2019), pp. C28–C51, https://doi.org/10.1137/18m1172466.

[25] R. Pollinger, J. Rentrop, D. Pflüger, and K. Kormann, *A Stable and Mass-Conserving Sparse Grid Combination Technique with Biorthogonal Hierarchical Basis Functions for Kinetic Simulations*, Journal of Computational Physics, 491 (2023), p. 112338, https://doi.org/10.1016/j.jcp.2023.112338.

[26] P. Sjöberg, P. Lötstedt, and J. Elf, *Fokker–Planck Approximation of the Master Equation in Molecular Biology*, Computing and Visualization in Science, 12 (2007), pp. 37–50, https://doi.org/10.1007/s00791-006-0045-6.

[27] B. Spencer and L. Bergman, *On the Numerical Solution of the Fokker-Planck Equation for Nonlinear Stochastic Systems*, Nonlinear Dynamics, 4 (1993), pp. 357–372, https://doi.org/10.1007/bf00120671.

[28] S. Wojtkiewicz, L. Bergman, B. Spencer, and E. Johnson, *Numerical Solution of the Four-Dimensional Nonstationary Fokker-Planck Equation*, Springer Netherlands, 2001, pp. 271–287, https://doi.org/10.1007/978-94-010-0886-0_22.