PARALLEL TIME INTEGRATION WITH MULTIGRID[*]

R. D. FALGOUT[‡], S. FRIEDHOFF[†], TZ. V. KOLEV[‡],
S. P. MACLACHLAN[†], AND J. B. SCHRODER[‡]

Abstract. We consider optimal-scaling multigrid solvers for the linear systems that arise from the discretization of problems with evolutionary behavior. Typically, solution algorithms for evolution equations are based on a time-marching approach, solving sequentially for one time step after the other. Parallelism in these traditional time-integration techniques is limited to spatial parallelism. However, current trends in computer architectures are leading towards systems with more, but not faster, processors. Therefore, faster compute speeds must come from greater parallelism. One approach to achieve parallelism in time is with multigrid, but extending classical multigrid methods for elliptic operators to this setting is not straightforward. In this paper, we present a non-intrusive, optimal-scaling time-parallel method based on multigrid reduction (MGR). We demonstrate optimality of our multigrid-reduction-in-time algorithm (MGRIT) for solving diffusion equations in two and three space dimensions in numerical experiments. Furthermore, through both parallel performance models and actual parallel numerical results, we show that we can achieve significant speedup in comparison to sequential time marching on modern architectures.

Key words. parabolic problems, reduction-based multigrid, multigrid-in-time, parareal

AMS subject classifications. 65F10, 65M22, 65M55

1. Introduction.

One of the major challenges facing the computational science community with future architectures is that faster compute speeds must come from increased concurrency, since clock speeds are no longer increasing but core counts are going up sharply. As a consequence, traditional time marching is becoming a huge sequential bottleneck in time integration simulations in the following way: improving simulation accuracy by scaling up the spatial resolution requires a similar (or greater) increase in the temporal resolution, which is also required to maintain stability in explicit methods. As a result, numerical time integration involves many more time steps leading to long overall compute times, since parallelizing only in space limits concurrency. Solving for multiple time steps in parallel and, therefore, increasing concurrency would remove this time integration bottleneck.

Because time is sequential in nature, the idea of simultaneously solving for multiple time steps is not intuitive. Yet it *is* possible, with work on this topic going back to as early as 1964 [33]. However, most research on this subject has been done within the past 30 years including [2, 7–10, 14–22, 25, 28, 31, 32, 38, 40–44]. One approach to achieve parallelism in time is with multigrid methods. The *parareal in time* method, introduced by Lions, Maday, and Turinici in [25], can be interpreted as a two-level multigrid method [16], even though the leading idea came from a spatial domain decomposition approach. The algorithm is optimal, but concurrency is limited since the coarse-grid solve is still sequential. Considering true multilevel (not two-level) schemes, only a few methods exhibit full multigrid optimality and concurrency such as [21, 42, 43], and most are designed for specific problems or discretizations. Furthermore, these methods are full space-time algorithms, whereas our algorithm employs a

[*]This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-JRNL-645325). The work of SF and SM was partially supported by the National Science Foundation, under grant DMS-1015370.

[†]Department of Mathematics, Tufts University, 503 Boston Avenue, Medford, MA 02155. email: {stephanie.friedhoff, scott.maclachlan}@tufts.edu

[‡]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P. O. Box 808, L-561, Livermore, CA 94551. email: {rfalgout, tzanio, schroder2}@llnl.gov.

non-intrusive semi-coarsening strategy using coarsening only in the time dimension.

Classical multigrid methods rely on multiscale representations in space, arising naturally by decomposing a function into a hierarchy of frequencies from global smooth modes to local oscillations. These approaches do not extend to evolutionary variables in a straightforward manner, because of the fundamentally local structure of the evolution. There are two approaches for extending classical multigrid methods to include the time dimension: multigrid only in time and space-time multigrid. In this paper, we present a multigrid-reduction-in-time algorithm (MGRIT) that is based on multigrid reduction (MGR) techniques [36,37]. The advantage of this approach is that it is easily integrated into existing codes, because it only requires a routine to integrate from one time to the next with some adjustable time step; i.e., our MGRIT algorithm simply calls an existing time-stepping routine. However, to achieve the full benefit of computing multiple time steps at once, space-time multigrid methods, where time is simply another dimension in the grid, have to be considered. This approach is more intrusive on existing codes and is a separate research topic not explored here.

It is important to note that the goal of adding parallelism to time integration is fundamentally different from that which motivates spatial multigrid. When using a scalable solver for each time-step, classical time-stepping is already an algorithmically optimal process, with best possible complexity. Instead, we aim to develop an approach that retains optimal algorithmic scaling, but with a much larger constant factor. By distributing this work over more processors than the increased factor, we aim to achieve speedup. As we are, in effect, accepting added computational work for added parallelism, the speedups achieved are modest in comparison to those seen for comparing optimal spatial solvers, such as multigrid, to sub-optimal techniques, such as LU factorization. Nonetheless, such speedups are important, as they allow us to make better use of parallel resources than can be achieved by spatial parallelism with sequential time-stepping.

This paper is organized as follows. In §2, ideas of reduction-based multigrid methods are reviewed and applied to time integration resulting in the MGRIT algorithm. In §3, we describe the parabolic model problem for our numerical experiments and demonstrate optimality of MGRIT for solving this model problem. Section 4 starts with weak scaling studies emphasizing optimal choices of MGRIT components for best overall time to solution as well as for robustness, followed by strong scaling studies comparing MGRIT with sequential time stepping. Finally, in §5, we draw some conclusions and discuss future work.

2. Multigrid in time based on MGR.

Considering the connection of time integration methods to the solution of lower block triangular linear systems of equations allows a connection to reduction-based multigrid methods that is crucial for our optimal-scaling, time-parallel method. In §2.1, we consider this correspondence, which is the basis of our description of the parareal algorithm in §2.2. Note that our presentation of the parareal algorithm as a standard residual correction scheme is not typical (though known [28]) but it allows us to show in §2.3 how the method can be interpreted as a two-level MGR scheme, corresponding to a two-level variant of the MGRIT algorithm that we describe in §2.4.

2.1. Connection to linear systems.

We consider a system of ordinary differential equations (ODEs) of the form

$$\mathbf{u}'(t) = \mathbf{f}(t, \mathbf{u}(t)), \quad \mathbf{u}(0) = \overline{\mathbf{u}}_0, \quad t \in [0, T], \tag{2.1}$$

such as in a method of lines approximation of a parabolic PDE. Let $t_i = i\delta t$, $i = 0, 1, \ldots, N_t$, be a temporal mesh with constant spacing $\delta t = T/N_t$, and, for $i = 1, \ldots, N_t$, let $\mathbf{u}_i$ be an approximation to $\mathbf{u}(t_i)$ and $\mathbf{u}_0 = \mathbf{u}(0)$. Then, a general one-step time discretization method for (2.1) can be written as

$$\begin{aligned} &\mathbf{u}_0 = \mathbf{u}(0) \\ &\mathbf{u}_i = \Phi_i(\mathbf{u}_{i-1}) + \mathbf{g}_i, \quad i = 1, 2, \ldots, N_t. \end{aligned} \tag{2.2}$$

In the case of a linear function $\mathbf{f}$, the function $\Phi_i(\cdot)$, corresponds to a matrix-vector product. For simplicity, we consider a time-independent discretization, thus, function $\Phi_i(\cdot)$ corresponds to a matrix-vector product with a fixed matrix denoted by Φ, $\Phi_i(\mathbf{u}_{i-1}) = \Phi\mathbf{u}_{i-1}$; specific examples of Φ will be given in §3.1. Then, the time discretization method (2.2) is equivalent to the linear system of equations

$$A\mathbf{u} \equiv \begin{bmatrix} I & & & \\ -\Phi & I & & \\ & \ddots & \ddots & \\ & & -\Phi & I \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N_t} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{N_t} \end{bmatrix} \equiv \mathbf{g}, \tag{2.3}$$

where $\mathbf{g}_0 = \mathbf{u}(0)$. Note that traditional time marching corresponds to a block forward-solve of this system, which is not directly parallelizable. Considering the lower block bidiagonal structure, we could apply cyclic reduction, a parallel direct factorization method that can equivalently be viewed as a multigrid method with a block smoother (called F-relaxation) and a Petrov-Galerkin coarse-grid operator that converges in one V-cycle. However, although cyclic reduction is optimal for scalar systems, for (2.3), it requires products of spatial blocks, the Φ matrices, that produce fill-in in the spatial dimension, yielding a method that is overall non-optimal. Nonetheless, the cyclic-reduction viewpoint can be useful in developing truly optimal and parallelizable methods. In fact, there are many spatial multigrid methods that have been designed from a similar reduction viewpoint [4, 5, 12, 23, 26, 35–37], replacing interpolation and/or the Petrov-Galerkin coarse-grid operator with suitable approximations. Using this perspective for the time dimension allows us to design an optimal-scaling time-parallel method. Before we pursue this approach, we first describe how parareal can be viewed as a standard residual correction scheme, laying the foundation of our interpretation of parareal as a two-level reduction-based multigrid method, considered in §2.3. The connection to reduction-based multigrid methods is crucial to achieve optimality when generalizing the two-level algorithm to multiple levels.

2.2. Parareal. One interpretation of parareal is to solve the system (2.3) iteratively, instead of with a direct method, by introducing a preconditioner on a coarse temporal mesh. Therefore, let $T_j = j\Delta T$, $j = 0, 1, \ldots, N_t/m$, be a coarse temporal mesh with constant spacing $\Delta T = m\delta t$, where m is a positive integer (see Figure 2.1).

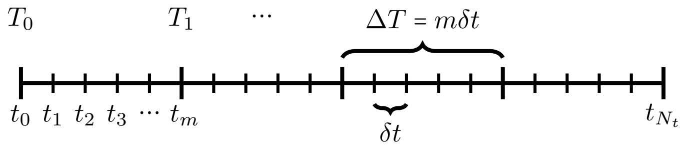


Fig. 2.1: Uniformly-spaced fine and coarse time discretization meshes.

It is easy to verify that the solution, $\mathbf{u}$, of (2.3) at mesh points $i = jm$, $j =$

$0, 1, \ldots, N_t/m$, satisfies the coarse system of equations

$$A_\Delta \mathbf{u}_\Delta = \begin{bmatrix} I & & & \\ -\Phi^m & I & & \\ & \ddots & \ddots & \\ & & -\Phi^m & I \end{bmatrix} \begin{bmatrix} \mathbf{u}_{\Delta,0} \\ \mathbf{u}_{\Delta,1} \\ \vdots \\ \mathbf{u}_{\Delta,N_t/m} \end{bmatrix} = R_\Phi \mathbf{g} \equiv \mathbf{g}_\Delta, \tag{2.4}$$

where $\mathbf{u}_{\Delta,j} = \mathbf{u}_{jm}$ and R_Φ is the rectangular restriction operator

$$R_\Phi = \begin{bmatrix} I & & & & & & & & \\ & \Phi^{m-1} & \cdots & \Phi & I & & & & \\ & & & & & \ddots & & & \\ & & & & & & \Phi^{m-1} & \cdots & \Phi & I \end{bmatrix}. \tag{2.5}$$

The parareal algorithm solves this coarse system iteratively, then computes the remaining fine values in parallel using (2.2) on each interval (t_{jm}, t_{jm+m-1}). To solve the coarse system (2.4), parareal uses the simple residual correction scheme

$$\mathbf{u}_\Delta^{k+1} = \mathbf{u}_\Delta^k + B_\Delta^{-1}(\mathbf{g}_\Delta - A_\Delta \mathbf{u}_\Delta^k), \tag{2.6}$$

where B_Δ is some coarse-scale time discretization of (2.1) (the analog of A on the coarse mesh),

$$B_\Delta = \begin{bmatrix} I & & & \\ -\Phi_\Delta & I & & \\ & \ddots & \ddots & \\ & & -\Phi_\Delta & I \end{bmatrix}.$$

The action of A_Δ is computed in parallel, but B_Δ^{-1} is computed sequentially, typically on a single processor. The residual correction (2.6) is usually presented as the following equivalent update step in the parareal literature

$$\mathbf{u}_{\Delta,j+1}^{k+1} = \Phi_\Delta \mathbf{u}_{\Delta,j}^{k+1} + \Phi^m \mathbf{u}_{\Delta,j}^k - \Phi_\Delta \mathbf{u}_{\Delta,j}^k + \mathbf{g}_{\Delta,j}, \quad j = 1, 2, \ldots, \tag{2.7}$$

with $\mathbf{u}_{\Delta,0}^k = \mathbf{g}_{\Delta,0}$, where Φ and Φ_Δ play the roles of the so-called fine and coarse propagators.

2.3. Parareal as a two-level multigrid reduction method. The key feature of parareal is the use of a coarse-scale time discretization that approximates the fine-scale evolution over the coarse-scale subspace. Since this idea is similar to the idea of MGR, it is not difficult to see how the parareal algorithm can be interpreted as a two-level MGR method. Therefore, let us partition the temporal mesh into C-points, given by the set of coarse time-scale points, $\{i = jm\}$, and F-points. Reordering the fine-grid operator, A, by F-points first and using the subscripts c and f to indicate the two sets of points, we consider the following well-known matrix decomposition, valid for any invertible matrix A with invertible submatrix A_{ff},

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} = \begin{bmatrix} I_f & 0 \\ A_{cf}A_{ff}^{-1} & I_c \end{bmatrix} \begin{bmatrix} A_{ff} & 0 \\ 0 & A_{cc} - A_{cf}A_{ff}^{-1}A_{fc} \end{bmatrix} \begin{bmatrix} I_f & A_{ff}^{-1}A_{fc} \\ 0 & I_c \end{bmatrix}, \tag{2.8}$$

where I_c and I_f are identity operators. We define the operators R, P (known as "ideal" restriction and interpolation since they define a Schur complement coarse grid), and S by

$$R = \begin{bmatrix} -A_{cf}A_{ff}^{-1} & I_c \end{bmatrix}, \quad P = \begin{bmatrix} -A_{ff}^{-1}A_{fc} \\ I_c \end{bmatrix}, \quad S = \begin{bmatrix} I_f \\ 0 \end{bmatrix}. \tag{2.9}$$

Then, since $A_{ff} = S^T AS$ and $A_{cc} - A_{cf}A_{ff}^{-1}A_{fc} = RAP$, it is straightforward to see from (2.8) that

$$A^{-1} = P\left(RAP\right)^{-1} R + S\left(S^T AS\right)^{-1} S^T,$$

and, thus,

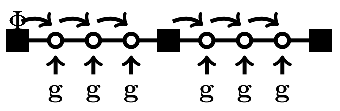
$$0 = I - A^{-1}A = I - P\left(RAP\right)^{-1} RA - S\left(S^T AS\right)^{-1} S^T A \tag{2.10}$$
$$= (I - P(RAP)^{-1}RA)(I - S(S^T AS)^{-1}S^T A), \tag{2.11}$$

where equivalence occurs since $RAS = 0$. We call (2.10) the additive identity and (2.11) the multiplicative identity. The multiplicative identity (2.11) defines the error propagator of an exact two-level multigrid method, with the first term corresponding to the error propagator of coarse-grid correction using the ideal Petrov-Galerkin coarse-grid operator (the Schur complement), RAP, and the second term being the error propagator of F-relaxation. To produce an iterative multigrid method, MGR methods [12, 23, 26, 35–37], for example, replace ideal interpolation and/or the ideal Petrov-Galerkin coarse-grid operator with various approximations and potentially add relaxation. Other methods in the literature that are closely related to the additive identity (2.10), such as ARMS [24, 39] and multigraph [4, 5], often use ILU factorizations to approximate A_{ff} and the Schur complement. Comparisons of the two types of algebraic multilevel methods have been considered in [27, 29, 30, 34]. Furthermore, as discussed in [29, 30], the order of coarse-grid correction and F-relaxation in (2.11) can be reversed although we do not consider this here.

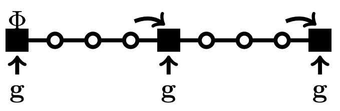
The parareal algorithm does something similar to MGR. Considering (2.6) at original time scale and defining $R_I = \begin{bmatrix} 0 & I_c \end{bmatrix}$ to be the coarse-scale injection operator, the error propagator for parareal is given by $P(I - B_\Delta^{-1}A_\Delta)R_I$. With the fine-grid operator, A, given by (2.3), the restriction operator, R, in (2.9) is the same as R_Φ in (2.5). Furthermore, the parareal coarse-grid operator, A_Δ, is equal to the Schur complement and thus, satisfies the Petrov-Galerkin condition, $A_\Delta = RAP$. Hence, the error propagator for parareal is equal to $(I - PB_\Delta^{-1}RA)PR_I$ which, using the operator S defined in (2.9), can be written as

$$(I - PB_\Delta^{-1}RA)(I - S(S^T AS)^{-1}S^T A). \tag{2.12}$$

Thus, the error propagator for the parareal algorithm is given by (2.11), with A_Δ replaced by the coarse-scale time discretization, B_Δ. To save computational work, we can replace R with R_I since $RAP = R_I AP$. Figure 2.2 shows schematic views of the actions of F- and C-relaxation for coarsening by a factor of four ($m = 4$). Furthermore, we note that ideal interpolation, P, corresponds to F-relaxation with a zero right-hand side.



(a) F-relaxation = time stepping in coarse time interval



(b) C-relaxation

Fig. 2.2: Schematic view of the actions of (a) F-relaxation and (b) C-relaxation for coarsening by a factor of four; ∘ represent F-points and ■ represent C-points.

Another two-level multigrid interpretation of parareal was given by Gander and Vandewalle in [16]. This interpretation is described in the full approximation storage (also called the full approximation scheme, FAS) framework [6] for the full nonlinear setting of (2.2) and is based on a different choice of operators. For the linear case, it is straightforward to show that the resulting two-level methods are the same. The advantage of the operator choice of [16] is its simplicity, but the advantage of our operator choice is that the multigrid components are more typical and, thus, allow us to extend the two-level method to an optimal multilevel algorithm. Furthermore, generalizing our MGR ideas to the nonlinear setting is straightforward in the FAS framework.

2.4. The optimal-scaling multilevel algorithm. Parareal is normally considered as a two-level method. The main drawback to this approach is that it requires a sequential forward solve of the coarse-grid system. For problems that involve a large number of time steps, the coarse-grid problem is still large and, thus, solving it in a sequential manner is a bottleneck analogous to the fine-grid forward solve. Results in §3.3 show the obvious generalization to a multilevel parareal algorithm does not yield scalable performance for V-cycles. Thus, we use the MGR viewpoint to develop an optimal multilevel algorithm. Previous research on MGR for developing optimal spatial multilevel methods [12, 23, 26, 35–37] motivates using the same coarse-grid operator, B_Δ, as in the two-level parareal method, but replacing F-relaxation with FCF-relaxation, and then applying the resulting method recursively. More precisely, we keep the first term in the two-level method (2.12), $I - PB_\Delta^{-1}RA$, and replace the second term, $I - S(S^TAS)^{-1}S^TA$, corresponding to the error propagator of F-relaxation with the product

$$(I - S(S^TAS)^{-1}S^TA)(I - R_I^T(R_IAR_I^T)^{-1}R_IA)(I - S(S^TAS)^{-1}S^TA),$$

corresponding to the error propagator of FCF-relaxation. With the error propagator of F-relaxation equal to PR_I, we can write the error propagator of FCF-relaxation as

$$P(I - A_{cc}^{-1}(A_{cc} - A_{cf}A_{ff}^{-1}A_{fc}))R_I = P(I - A_\Delta)R_I,$$

where equivalence occurs since $A_{cc} = I$. Thus, FCF-relaxation corresponds to Jacobi smoothing on the coarse time grid with the true Schur complement coarse-grid operator, $A_\Delta = RAP$.

To describe the MGRIT algorithm, we consider a hierarchy of time discretization meshes, Ω_l, $l = 0, 1, \ldots, L = \log_m(N_t)$, with constant spacing δt on level 0, $m\delta t$ on level 1, etc., for a positive coarsening factor, m. Let $A_l\mathbf{u}^{(l)} = \mathbf{g}^{(l)}$ be the linear system of equations on level $l = 0, 1, \ldots, L$, where A_l is the time discretization on the mesh Ω_l, characterized by the matrix Φ_l. For each level, l, we decompose the matrix A_l into F- and C-points and define the interpolation operator, P, as in (2.9). Then, the MGRIT V-cycle algorithm for solving (2.1) can be written as follows:

```
MGRIT($l$)
if $l$ is the coarsest level, $L$
    • Solve coarse-grid system $A_L\mathbf{u}^{(L)} = \mathbf{g}^{(L)}$.
else
    • Relax on $A_l\mathbf{u}^{(l)} = \mathbf{g}^{(l)}$ using $FCF$-relaxation.
    • Compute and restrict residual using injection,
      $\mathbf{g}^{(l+1)} = R_I(\mathbf{g}^{(l)} - A_l\mathbf{u}^{(l)})$.
    • Solve on next level: MGRIT($l+1$).
    • Correct using "ideal interpolation", $\mathbf{u}^{(l)} \leftarrow \mathbf{u}^{(l)} + P\mathbf{u}^{(l+1)}$.
end
```

Note that as in the spatial MGR context [37], W- and F-cycle versions of the MGRIT algorithm can be defined. Assuming exact arithmetic, one iteration of F-relaxation computes the exact solution at all F-points in the first coarse-scale time interval, (T_0, T_1), whereas one iteration of FCF-relaxation computes the exact solution at all F-points in the first two coarse-scale time intervals, (T_0, T_1) and (T_1, T_2), as well as at T_1 corresponding to the first C-point. Each additional iteration of F-relaxation or FCF-relaxation computes the exact solution at all points of one or two additional coarse-scale time interval(s), respectively. Thus, parareal and MGRIT solve for the exact solution in N_t/m or $N_t/(2m)$ iterations, respectively, corresponding to the number of points on the first coarse grid or to half the number of points on the first coarse grid. This is an interesting property of the two algorithms; however, in practice, the fact that they converge to some error tolerance in $\mathcal{O}(1)$ iterations is more relevant.

3. Numerical results.

To test the MGRIT approach developed in §2.4, we consider a parabolic model problem, the diffusion equation in d space dimensions. In §3.1, we describe implicit and explicit discretizations of this model problem and their correspondence to Φ in (2.3), followed by a brief description about implementation details in §3.2. Optimality of MGRIT for solving the model problem in two space dimensions with implicit and explicit time discretization is then demonstrated in §3.3.

3.1. The parabolic model problem.

We consider the diffusion equation in d space dimensions,

$$\mathbf{u}_t - \kappa\Delta\mathbf{u} = \mathbf{b}(\mathbf{x}, t), \quad \kappa > 0, \quad \mathbf{x} \in \Omega = [0, \pi]^d,\ t \in [0, T], \tag{3.1}$$

subject to an initial condition and homogeneous Dirichlet boundary conditions,

$$\mathbf{u}(\mathbf{x}, 0) = \overline{\mathbf{u}}_0(\mathbf{x}), \quad \mathbf{x} \in \Omega \tag{3.2}$$

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{0}, \quad \mathbf{x} \in \partial\Omega,\ t \in [0, T]. \tag{3.3}$$

We transform our model problem to a system of ODEs of the form (2.1) by using central finite differences for discretizing the spatial derivatives and first-order methods for the time derivative. In the following, we consider only the case $d = 2$; the operators for $d = 1$ and $d = 3$ are defined analogously. Let $(x_j = j\Delta x,\ y_k = k\Delta y,\ t_i = i\delta t)$, $j = 0, 1, \ldots, N_x$, $k = 0, 1, \ldots, N_y$, $i = 0, 1, \ldots, N_t$, be a uniform space-time mesh with spacing $\Delta x = \pi/N_x$, $\Delta y = \pi/N_y$, and $\delta t = T/N_t$, respectively. Furthermore, for $i = 0, 1, \ldots, N_t$, let $\mathbf{u}_i$ be an approximation to $\mathbf{u}(\mathbf{x}, t_i)$ with $\mathbf{u}_0 = \overline{\mathbf{u}}_0(\mathbf{x})$ using the initial condition, (3.2). If we use backward Euler for the time discretization, we obtain

$$(I + \delta t M)\mathbf{u}_i - \delta t\mathbf{b}_i = \mathbf{u}_{i-1}, \quad i = 1, 2, \ldots, N_t,$$

defining a one-step method of the form (2.2) with $\Phi = (I + \delta t M)^{-1}$ and $\mathbf{g}_i = (I + \delta t M)^{-1} \delta t \mathbf{b}_i$ for $i = 1, 2, \ldots, N_t$, where M is the usual central finite-difference discretization of $-\kappa \Delta \mathbf{u}$,

$$M = \begin{bmatrix} & -a_y & \\ -a_x & 2\left(a_x + a_y\right) & -a_x \\ & -a_y & \end{bmatrix} \quad \text{with } a_x = \frac{\kappa}{(\Delta x)^2}, \; a_y = \frac{\kappa}{(\Delta y)^2}.$$

If we use forward Euler for the time discretization, then $\Phi = I - \delta t M$ and $\mathbf{g}_i = \delta t \mathbf{b}_{i-1}$ for $i = 1, 2, \ldots, N_t$. Thus, for our simple model problem using an implicit time discretization, the time integrator Φ corresponds to a spatial solve, whereas it coincides with a matrix-vector product in the explicit case. We use the same discretization technique (with adjusted time step size) to define the discrete operators on all levels.

In the following, we primarily report on tests of solving the model problem on the space-time domain $[0, \pi]^d \times [0, T]$ with a zero right-hand side, $\kappa = 1$, and subject to the initial condition $\mathbf{u}(x, y, 0) = \sin(x)\sin(y), \; 0 \le x, y \le \pi$, in the case that $d = 2$ or $\mathbf{u}(x, y, z, 0) = \sin(x)\sin(y)\sin(z), \; 0 \le x, y, z \le \pi$, if $d = 3$. Choosing a zero right-hand side allows us to easily verify that MGRIT computes a good approximation to the true solution. However, we emphasize that a non-zero right-hand side does not change our algorithm, since it only defines the right-hand side, $\mathbf{g}$, of the linear system (2.3). When noted, we also report results for the same problem in two space dimensions, with $\mathbf{b}(\mathbf{x}, t) = -\sin(x)\sin(y)(\sin(t) - 2\cos(t))$. On the finest grid, the initial condition is used as the initial guess for $t = 0$, and a random initial guess for all other times. Choosing a random initial guess for all times $t > 0$ corresponds to not using any knowledge of the right-hand side that could affect convergence. However, in practice, a random initial guess is not recommended.

Notation. We present results for a variety of discretizations of the model problem in two and three space dimensions. To facilitate readability, in general, only the space-time grid size is specified in the caption of tables and figures, and the following labels are used

Implicit2D(T = ·) model problem in two space dimensions with backward Euler time discretization

***Implicit2D(T = ·)*-F** model problem in two space dimensions with non-zero forcing term, with backward Euler time discretization

Implicit3D(T = ·) model problem in three space dimensions with backward Euler time discretization

Explicit2D(T = ·) model problem in two space dimensions with forward Euler time discretization

Note that the space-time grid size and the final time, T, of the time interval uniquely define the step sizes of the discretization using the relationships $\Delta x = \pi / N_x$, $\Delta y = \pi / N_y$, $\Delta z = \pi / N_z$, and $\delta t = T / N_t$.

3.2. Implementation details. We have implemented the MGRIT algorithm described in §2.4 in parallel using C and Message Passing Interface (MPI). The code decomposes the original temporal grid such that each processor owns a time interval of roughly the same size, and coarse grids are distributed in the usual multigrid fashion according to their parent fine grids. As a result, the first (likewise, last) point on a processor on any given level could be an F-point or a C-point, and a processor may not own any points at all on some coarse levels. The code attempts to overlap communication and computation by computing on the rightmost F-interval first and sending information downstream as soon as possible. Also, to save on memory, only solution values at C-points are stored.

Since MGRIT is a non-intrusive approach, the time integrator Φ in MGRIT is essentially the same as in an algorithm with sequential time stepping. We have implemented Φ for both time integration approaches in parallel using C and the *hypre* [1] package. In the implicit time discretization case, the spatial systems are solved using the *hypre* solver *PFMG* [3, 11]. *PFMG* is a parallel alternating semicoarsening multigrid V-cycle solver that automatically determines the direction of semicoarsening minimizing problem anisotropies. For our experiments, we use $V(1,1)$-cycles with red-black Gauss-Seidel relaxation, full coarsening (skip = 1), and coarse-grid operators formed algebraically by the non-Galerkin process described in [3]. The convergence tolerance is based on the relative residual and chosen to be 10^{-9} unless otherwise specified.

3.3. Optimality of MGRIT. To demonstrate optimality of MGRIT, we consider iteration counts for solving the model problem in two space dimensions with implicit and explicit time discretizations. The iteration counts are based on achieving an absolute space-time residual norm of less than 10^{-9}, measured in the discrete L_2-norm. For the implicit case, we perform domain-refinement studies commonly used for multigrid methods, whereas, for the explicit case, studies focus on the CFL condition.

3.3.1. Implicit time integration schemes. We consider a domain-refinement study for two-level and true multilevel, $L = \log_2(N_t)$, variants of MGRIT V- and F-cycle algorithms with factor-2 coarsening and various relaxation schemes. For this purpose, we fix a space-time domain and simultaneously scale up the spatial and temporal resolutions. More precisely, the time step on the finest grid is chosen to be $\delta t = (\Delta x)^2 = (\Delta y)^2$, and the time step on each coarse grid is given by $2^{l-1}\delta t,\ l > 0$. Hence, we quadruple the number of points in time when doubling the number of points in space. The following three relaxation schemes are considered: F-relaxation, FCF-relaxation, and F-FCF-relaxation defined as F-relaxation on the finest grid and FCF-relaxation on all other levels, $l > 0$. F-relaxation corresponds to the parareal method and the obvious generalization to a multilevel parareal algorithm, FCF-relaxation is motivated by the MGR viewpoint and F-FCF-relaxation is chosen for efficiency reasons. Both V- and F-cycle variants are considered, knowing added coarse-grid work in an F-cycle better approximates the two-level method.

Table 1 shows that for MGRIT V- and F-cycles using FCF or F-FCF-relaxation, the iteration counts appear to be bounded independently of the problem size. While the same holds true for two-level MGRIT with F-relaxation, i.e., the parareal method, iteration counts increase for multilevel V-cycles, and F-cycles are necessary to achieve good scaling. Thus, the obvious generalization of parareal to multiple levels does not produce an optimal algorithm. Put another way, the additional full FC relaxation sweep on the fine grid is necessary to achieve optimality in the multilevel V-cycle algorithm. While F-cycles maintain optimal algorithmic scaling, they require more communication than V-cycles do and, as such, often lead to poorer overall parallel performance, as will be seen in §4.1.

Tests also indicate that the above results are largely independent of the coarsening factor, i.e., the number of iterations does not change significantly when coarsening by larger factors. However, there is another parallel performance tradeoff with this apparent reduction in work that we also discuss in §4.1.

	$N_x^2 \times N_t$ =	$(2^4)^2 \times 2^5$	$(2^5)^2 \times 2^7$	$(2^6)^2 \times 2^9$	$(2^7)^2 \times 2^{11}$	$(2^8)^2 \times 2^{13}$
FCF-relax.	two-level	7	8	8	7	7
	V-cycle	7	9	9	10	10
	F-cycle	7	8	7	7	7
F-FCF-relax.	V-cycle	10	11	11	11	11
	F-cycle	10	11	10	10	10
F-relax.	two-level	10	11	10	10	10
	V-cycle	12	17	24	29	31
	F-cycle	10	10	10	10	10

Table 1: Number of iterations for solving *Implicit2D($T = \pi^2/8$)* on $(N_x+1)^2 \times (N_t+1)$ space-time grids using two-level MGRIT schemes, MGRIT V- and F-cycles with factor-2 coarsening and various relaxation schemes.

	c =	1	2	4	8	16	32	64
FCF-relax.	two-level	7	7	8	8	8	7	7
	V-cycle	10	10	9	9	9	8	7
	F-cycle	7	7	7	8	8	7	7
F-FCF-relax.	V-cycle	11	11	11	11	11	11	10
	F-cycle	10	10	10	10	11	11	10
F-relax.	two-level	10	10	10	10	11	11	10
	F-cycle	10	10	10	10	10	10	10

Table 2: Number of two-level and multilevel MGRIT iterations for solving *Implicit2D($T = \pi^2/8$)* for different ratios $c = \delta t/(\Delta x)^2$ with $\Delta x = \pi/128$ fixed, coarsening by a factor of 2 and various relaxation schemes.

The choice of $\delta t = (\Delta x)^2$ arises from the differences in accuracy of the spatial and temporal discretizations. In order to achieve balanced accuracy, it is necessary to take $\delta t = c(\Delta x)^2$ for some moderate constant c. In Table 2, we look at the effects of the time-step size on iteration counts. We consider the space-time domain $[0,\pi]^2 \times [0,\pi^2/8]$, fixed spatial mesh sizes of $\Delta x = \Delta y = \pi/128$, and increase the time-step size with the ratio $c = \delta t/(\Delta x)^2$. The results show that iteration counts are largely independent of the time-step size.

3.3.2. Accuracy of spatial solves. Using an implicit time discretization method, the function $\Phi(\cdot)$ corresponds to a spatial solve. In particular, for our model problem, $\Phi = (I + \delta t M)^{-1}$. In practice, however, these spatial problems are solved with an iterative method such as multigrid. As a consequence, instead of solving the linear system of equations (2.3), we actually solve

$$\hat{A}\mathbf{u} \equiv \begin{bmatrix} I & & & \\ -\hat{\Phi} & I & & \\ & \ddots & \ddots & \\ & & -\hat{\Phi} & I \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N_t} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{N_t} \end{bmatrix} \equiv \mathbf{g}, \tag{3.4}$$

where $\hat{\Phi} \approx \Phi$ and $\mathbf{g}_0 = \mathbf{u}(0)$.

Considering an iterative method for the spatial solves at each time step raises the question of how to choose a good initial guess for these solves. In traditional time marching, typically the solution from the previous time step is used. Note that this is a good choice if the solution is smooth in time. MGRIT solves the linear system (3.4) iteratively and thus, each MGRIT iteration computes a better approximation of the solution at each time step. Therefore, a good initial guess for the spatial solve at a particular time step is the current approximation of the solution at this time step, i.e., the solution at the time step obtained in the previous MGRIT iteration. As discussed in §3.2, in order to achieve some degree of efficient memory use, our implementation stores only the solution at the C-points of each grid. Thus, we cannot use the current approximation of the solution as the initial guess for the spatial solve.

Similarly to time-stepping algorithms, we use the solution from the previous time step of the current MGRIT iteration as the initial guess. This choice is optimal with respect to the constraint of efficient memory use. However, when not using the current approximation of the solution for the spatial solves, the accuracy of the spatial solves becomes crucial, particularly for optimizing parallel performance. More precisely, choosing an approximation $\hat{\Phi} \approx \Phi$ raises two questions: first, should we use the same approximation for all MGRIT iterations? And second, how accurately should we solve the spatial problems on coarse time grids? To answer these questions, we consider the effect on iteration counts for a heuristic choice for the accuracy of the approximation. Since our goal is to reduce overall compute time, in addition to considering iteration counts, we also look at runtimes in a weak scaling study in §4.1.

In general, there are two simple strategies for choosing the accuracy of the spatial solves: a stopping tolerance-based accuracy, i.e., solving the spatial problems to a given stopping tolerance, and a fixed iteration-based accuracy, i.e., using a fixed number of iterations to solve the spatial problems. We choose a combination of these two strategies: on the finest grid, we use a stopping tolerance-based accuracy and on all other levels, $l > 0$, we use a fixed iteration-based accuracy. With this choice, we limit computational work on the coarse grids, and we approximate Φ on the finest grid as is typically done in time marching schemes. Furthermore, to save some computational work on the finest grid, we start with a loose stopping tolerance for the spatial solves and tighten it as we converge in time. More precisely, our choice for the stopping tolerance, tol_x, for the spatial solves on the finest grid is based on the norm of the residual of the previous MGRIT iteration, $\mathrm{acc}_t = \|\mathbf{r}_{k-1}\|$, as follows: we pick a loose and a tight spatial stopping tolerance,

$$\mathrm{tol}_x^{(\mathrm{loose})} = 10^{-\mathrm{loose}} \quad \text{and} \quad \mathrm{tol}_x^{(\mathrm{tight})} = 10^{-\mathrm{tight}},$$

where $\mathrm{tol}_x^{(\mathrm{tight})}$ is the same stopping tolerance we would use for time stepping. Furthermore, we define an accuracy, $\mathrm{acc}_t^{(\mathrm{close})} = 10^{-\mathrm{close}}$, characterizing "being close" to converging in time. If $\mathrm{acc}_t \le \mathrm{acc}_t^{(\mathrm{close})}$, $\mathrm{tol}_x^{(\mathrm{tight})}$ is used as the stopping tolerance for the spatial solves. Note that this ensures that we solve the same linear system that we solve with time stepping. If $\mathrm{acc}_t > \mathrm{acc}_t^{(\mathrm{close})}$, we use a linear function of the logarithm of the residual norm to determine the logarithm of the stopping tolerance for the spatial solves, as depicted in Figure 3.1.

Note that changing the tolerance of the spatial solves changes the approximation, $\hat{\Phi}$ and, thus, the problem that we consider. Large changes in the tolerance might result in an increase of the residual norm, acc_t. In that case, instead of loosening the tolerance again, the same tolerance of the spatial solves is kept until the residual norm decreases enough such that the tolerance is tightened further. Put another way, we never decrease the accuracy of spatial solves from one MGRIT iteration to the next.

To demonstrate that MGRIT computes a good approximation of the true discrete solution when using the heuristic choice for the accuracy of the spatial solves, we

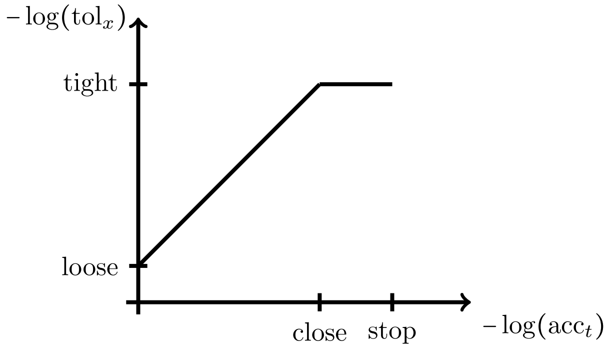


Fig. 3.1: Defining the stopping tolerance for the spatial solves on the finest grid, tol_x, as a function of the norm of the residual of the previous MGRIT iteration, $\mathrm{acc}_t = \|\mathbf{r}_{k-1}\|$.

compare the solution obtained by time stepping with the solution obtained by several MGRIT variants using a stopping tolerance-based accuracy with loose = 2 and tight = 9 on the finest grid and two V-cycles on all coarse grids. We look at the norm of the error to the true discrete solution for the problem with non-zero forcing. More precisely, let s denote the time integration scheme used to solve the discrete space-time problem, i.e., s is either time stepping or an MGRIT variant. Furthermore, for each time step t_i, let $e_s = \|\mathbf{e}_i^{(s)}\|$ denote the norm of the error to the true discrete solution using the scheme s, and let $e_{\min} = \min_s \|\mathbf{e}_i^{(s)}\|$ be the minimum of these norms over all time integration schemes. The left-hand side of Figure 3.2 plots e_s for time-stepping and four MGRIT variants. The norm of the error for all schemes is roughly of the same size, independently of the time integration scheme used for the numerical computation, although some small added accuracy is seen for this problem with MGRIT V-cycles with $F-FCF$ relaxation. At the right of Figure 3.2, we plot $e_s - e_{\min}$ for each scheme, where $e_{\min}$ is computed without including the $F-FCF$ relaxation results. Here, we see that there is no inherent advantage or disadvantage for the MGRIT approaches with this heuristic for accuracy of spatial solves. The solutions obtained by all MGRIT variants correspond to the solution obtained by time stepping within some tolerance. We emphasize that only the fact that all schemes compute the same solution up to some tolerance is important. The scheme with minimum error can vary from time step to time step and as a result of parameter choices.

In Table 3, we look at the effects of our heuristic choice for the accuracy of the spatial solves on iteration counts. We consider the same parameters as in Table 1 with the exception that instead of solving all spatial problems to 10^{-9} accuracy, we use our heuristic. Compared to Table 1 we see an increase in iteration counts, but the number of iterations still appears to be bounded independently of the problem size. While two-level schemes as well as F-cycles perform only a little worse, we observe some degradation for V-cycles. However, this degradation is not critical, especially since parallel time-to-solution is more important than iteration counts.

3.3.3. Explicit time integration schemes. For explicit time discretization schemes, considering parallel time integration is highly relevant because the number of time steps in these methods is usually quite large. However, stability issues on coarse grids render the straightforward application of our approach infeasible. One possibility for circumventing this problem is to use an implicit discretization on coarse grids. Using this approach in conjunction with aggressive coarsening on the finest grid

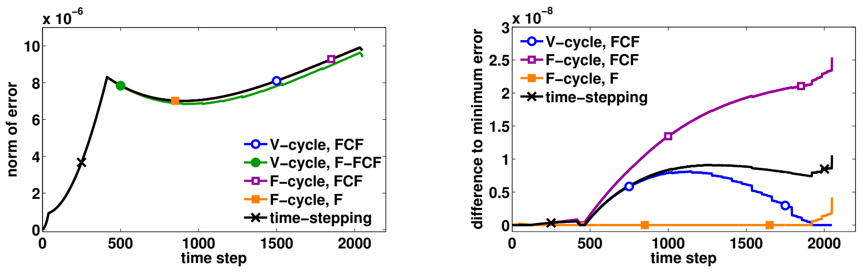


Fig. 3.2: Comparison of norms of errors to the true discrete solution of *Implicit2D*$(T = \pi^2/8)$-F on a $129^2 \times 2049$ space-time grid at each time step, $\|\mathbf{e}_i^{(s)}\|$, using different time integration schemes, s. At left, $\|\mathbf{e}_i^{(s)}\|$, and at right, difference between $\|\mathbf{e}_i^{(s)}\|$ and the minimum error norm at each time step over all time integration schemes, $\min_s \|\mathbf{e}_i^{(s)}\|$, omitting $F-FCF$ results.

	$N_x^2 \times N_t =$	$(2^4)^2 \times 2^5$	$(2^5)^2 \times 2^7$	$(2^6)^2 \times 2^9$	$(2^7)^2 \times 2^{11}$	$(2^8)^2 \times 2^{13}$
FCF-relax.	two-level	12	12	11	11	12
	V-cycle	13	16	16	15	18
	F-cycle	12	12	11	11	12
F-FCF-relax.	V-cycle	17	17	17	19	17
	F-cycle	13	13	14	13	14
F-relax.	two-level	13	13	13	13	14
	F-cycle	13	13	13	13	14

Table 3: Results similar to those in Table 1, but using a stopping tolerance-based accuracy for the spatial solves on the finest grid as depicted in Figure 3.1 (with loose = 2 and tight = 9) and fixed iteration-based accuracy (2 iterations) for spatial solves on all other levels, $l > 0$, instead of solving all spatial problems to 10^{-9} accuracy.

to reduce the cost of the implicit coarse solve makes this a worthwhile approach to achieve speedup in comparison to traditional time-marching methods. The downside from a user perspective is that both explicit and implicit time-stepping routines are needed.

The explicit-implicit approach allows us to avoid stability issues on the coarse grids, but it raises the question of robustness. To answer this question, we consider the model problem in two space dimensions with explicit time discretization on the space-time domain $[0,\pi]^2 \times [0,\pi^2/8]$. For simplicity, we assume that we use the same spatial mesh size in both dimensions, $\Delta y = \Delta x$. Thus, the CFL condition is given by $0 < \delta t < (\Delta x)^2/4$. Table 4 shows iteration counts for the two-level variant of our MGRIT algorithm for different CFL numbers, $c = \delta t/(\Delta x)^2$, and coarsening factors 2 and 16. Note that since we consider a fixed time interval, we have to decrease the number of time steps when increasing the time step size with the CFL number. Table 4 shows that iteration counts for the case that δt is away from the CFL limit (small values of c), are independent of the CFL number. However, as δt approaches the CFL limit, iteration counts increase, especially for factor-2 coarsening.

The increase in iteration counts as δt approaches the CFL limit results from properties of the discrete solution. For the discretization considered in our numerical experiments, the discrete solution can be expressed in terms of eigenvalues and eigenvectors of the discrete Laplacian. If we denote the eigenvectors and

$c =$	0.15	0.2	0.22	0.23	0.24	0.245	0.249	0.2499
$m = 2$	11	11	13	19	37	71	283	472
$m = 16$	9	9	9	9	9	12	47	65

Table 4: Number of two-level V-cycle MGRIT iterations with FCF-relaxation for solving *Explicit2D(*$T = \pi^2/8$*)* for different CFL numbers, $c = \delta t/(\Delta x)^2$ with $\Delta x = \pi/64$ fixed, coarsening by a factor of 2 or 16, forward Euler on the fine temporal grid and backward Euler on the coarse grid.

corresponding eigenvalues of the negative discrete Laplacian, M, by $\mathbf{v}_{j,k}$ and $\lambda_{j,k}$, $j = 1, \ldots, N_x - 1,\ k = 1, \ldots, N_y - 1$, respectively, we can write the initial condition, $\overline{\mathbf{u}}_0(\mathbf{x})$, as a linear combination of the $\mathbf{v}_{j,k}$,

$$\overline{\mathbf{u}}_0(\mathbf{x}) = \sum_{j,k} \alpha_{j,k} \mathbf{v}_{j,k}.$$

The discrete solution at time t_i is then given by

$$\mathbf{u}_i = \sum_{j,k} \alpha_{j,k} \left(1 - \delta t \lambda_{j,k}\right)^i \mathbf{v}_{j,k}.$$

For modes that are oscillatory in both space dimensions, we have that $\lambda_{j,k} \approx 8/(\Delta x)^2$. If we write the damping factor, $\left(1 - \delta t \lambda_{j,k}\right)^i$, at time t_i in terms of the CFL number c, we obtain $(1 - 8c)^i$. Close to the CFL limit, i. e., $c \approx 0.25$, we have $1 - 8c \to -1$. Thus, the space-time solution contains a component that is oscillatory in time with an amplitude that gets damped very slowly in time. As a consequence, as δt approaches the CFL limit, convergence of MGRIT depends on the number of time steps, N_t, which controls the magnitude of the oscillations at the end of the time interval. For large N_t, MGRIT cannot effectively reduce this oscillatory component of the error since FCF-relaxation is not a good smoother for this component, especially if we consider small coarsening factors. Furthermore, coarse-grid correction does not help since oscillatory modes are not visible on the coarse grid. Therefore, close to the CFL limit sequential time stepping should be used. For MGRIT, we would have to consider large coarsening factors relative to the number of time steps which limits the amount of parallelism in relaxation and, thus, the benefits of our approach. However, since the existence of a component that is oscillatory in time is not physical, this is not a real restriction for using the MGRIT algorithm. A domain refinement study similar to that in Table 1 for fixed c (away from the CFL limit) shows that iteration counts for the multilevel explicit-implicit approach appear to be bounded independently of the problem size, but a full study of the benefit in a parallel simulation code is still needed.

4. Parallel results. In this section, we compare the time to solution using MGRIT to the time to use sequential time stepping. In particular, we are interested in answering three questions. First, is it beneficial to use MGRIT on modern architectures? Second, considering the time to solution with both methods as functions of the number of processors, where is the crossover point? And third, what speedup can we expect from using MGRIT?

Numerical results in this section are, unless otherwise noted, generated on Cab, a Linux cluster at Lawrence Livermore National Laboratory consisting of 1,296 compute nodes, with two eight-core 2.6 GHz Intel Xeon processors per node. The nodes are connected by an InfiniBand QDR interconnect. We also include results for weak scaling on Vulcan, a Blue Gene/Q system at Lawrence Livermore National Laboratory consisting of 24,576 nodes, with sixteen 1.6GHz PowerPC A2 cores per node and a 5D Torus interconnect. Since particular choices of various components of the MGRIT algorithm affect parallel performance, we first aim at optimizing choices for best overall time to solution, as well as for robustness. In §4.1, we consider the effect of our heuristic choice for the accuracy of the spatial solves described in §3.3.2 on time to solution and compare various relaxation schemes for MGRIT V- and F-cycles. In §4.2, we look at iteration counts and computation times as functions of the coarsening factor to determine a good coarsening strategy. We then use these results in Sections 4.3 and 4.4 to choose a set of MGRIT variants for strong scaling studies and comparison to sequential time stepping. In §4.5 we briefly review a simple parallel performance model that allows predictions for larger computational scales, followed by a comparison of MGRIT to sequential time-stepping using this model.

4.1. Optimizing MGRIT cycling. The results in §3.3.2 show that our heuristic choice for the accuracy of the spatial solves preserves optimality but increases iteration counts slightly compared to solving all spatial problems to high accuracy. Since we are interested in the best overall time to solution, we look at the effect of the accuracy of the spatial solves on compute time in a weak scaling study. More precisely, we fix the domain size and choose $\delta t = (\Delta x)^2 = (\Delta y)^2$ as in the domain-refinement study in §3.3.1. Halving the spatial step size Δx requires quadrupling the number of time steps and, thus, for proper weak scaling we increase the number of processors by factors of 16.

Figure 4.1 shows weak scaling results for MGRIT V- and F-cycles with factor-2 coarsening and various relaxation schemes. The time curves show that the MGRIT algorithm scales well for both choices of the accuracy of the spatial solves. Note that the log-linear scaling of the axes shows a growth in time roughly proportional to $\log(P)$, where P denotes the number of processors. Furthermore, comparing overall compute times, we see that it is beneficial to use our heuristic choice by as much as a factor of about two.

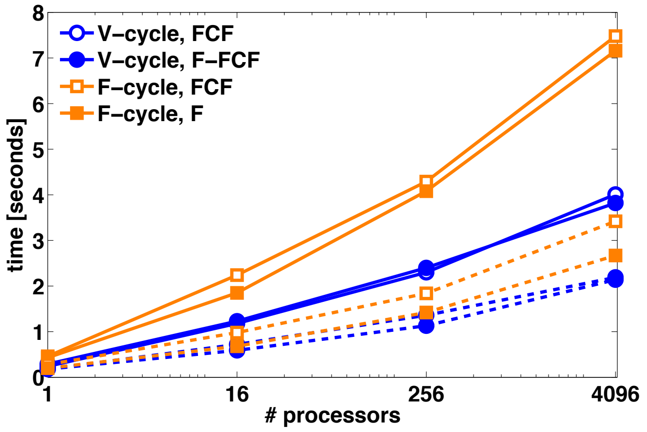


Fig. 4.1: Time to solve *Implicit2D($T = \pi^2/8$)* using MGRIT V- and F-cycles with factor-2 coarsening and various relaxation schemes. The problem size per processor is about $(2^4)^2 \times 2^5$. Solid lines are results for solving all spatial problems to 10^{-9} accuracy and dashed lines represent runtimes for using our heuristic choice for the accuracy of the spatial solves.

4.2. Optimizing MGRIT coarsening. So far, we have only presented results for factor-2 coarsening. To determine the effect of other coarsening factors on parallel performance, we consider solving *Implicit2D($T = \pi^2/8$)* on a $129^2 \times 2049$ space-time grid with the MGRIT V- and F-cycle algorithms using 32 processors for parallelizing only in time. Note that for this particular problem size and number of processors, the local temporal problem size on each processor is about 64. Figure 4.2 shows iteration counts (dashed lines) and compute times (solid lines) as functions of the coarsening factor for various MGRIT variants. In all cases, iteration counts are bounded independently of the coarsening factor. More precisely, for V-cycles using F-relaxation on the finest grid and FCF-relaxation on all other levels, $l > 0$, as well as F-cycles using FCF- or F-relaxation on all levels, the iteration counts increase slightly, stagnate, and then decrease. Considering V-cycles with FCF-relaxation on all levels, iteration counts are non-increasing. One argument for this behavior is that more aggressive coarsening leads to stronger relaxation, and MGRIT looks more like time stepping. In particular, using a coarsening factor of 2048, we have two time levels and thus, considering FCF-relaxation on the fine grid, MGRIT converges in one iteration. For F-relaxation on the fine grid, it converges in two iterations. Note that with a coarsening factor of 2049, all variants converge in exactly one iteration.

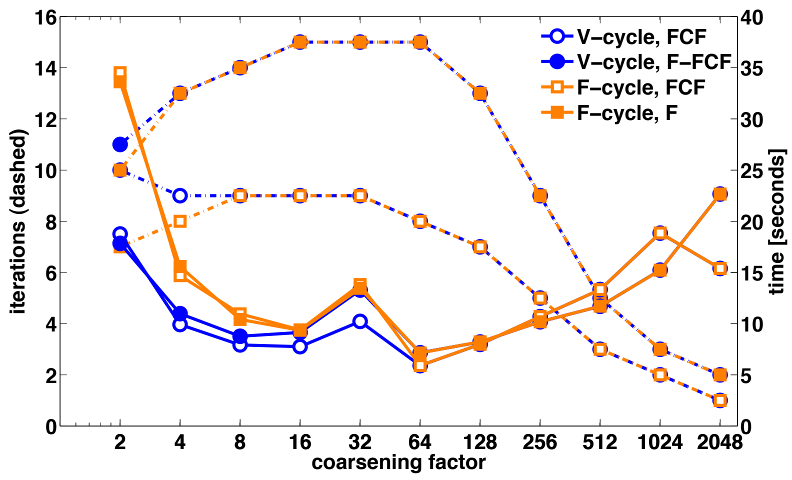


Fig. 4.2: Number of iterations and overall compute time for solving *Implicit2D($T = \pi^2/8$)* on a $129^2 \times 2049$ space-time grid using MGRIT V- and F-cycles with various relaxation schemes, 32 processor for parallelizing only in time, all spatial solves to 10^{-9} accuracy.

Compute time behaves in the opposite manner of iteration counts. Generally, compute time decreases in the beginning, since MGRIT requires less communication. At some point, we lose parallelism in relaxation, which causes the time to solution to increase even though iteration counts decrease. More precisely, when the number of time levels decreases, which is the case for coarsening factors 4, 8, 16, 64, and 2048, we generally see a drop in compute time. When the number of levels does not change, as for coarsening factors 32 and 128 through 1024, we lose parallelism in relaxation resulting in an increase in compute time.

Summarizing the above results, aggressive coarsening with $m > 2$ reduces the cost of coarse-grid solves. However, relaxation is expensive when the number of time points on each processor is small since, in that case, F-relaxation requires sequential communication for all processors in a given interval of F-points. A coarsening strategy that aims to balance these two effects is to use aggressive coarsening in conjunction with factor-2 coarsening on coarse grids on which the local problem size is small. We consider weak scaling again to look at the effect of such a coarsening strategy on overall time to solution in comparison to coarsening strategies with a fixed coarsening factor on all time levels. For larger coarsening factors to be meaningful, we distribute the space-time domain such that the local space-time domain on each processor consists of $n_x = 2^7$ points in each space dimension and $n_t = 2^8$ time points (instead of $n_x = 2^4$ and $n_t = 2^5$ in our first weak scaling study).

Figure 4.3 shows weak scaling results for MGRIT V-cycles and three different coarsening strategies: factor-2 coarsening on all levels, factor-16 coarsening on all levels, and a combination of factor-16 and factor-2 coarsening. For the latter coarsening strategy, we used factor-16 coarsening on all levels on which the number of time points on each processor is 16 or greater, and factor-2 coarsening on all other levels. The results show that aggressive coarsening in conjunction with factor-2 coarsening minimizes overall compute time. Note that this coarsening strategy could be optimized further by considering even more aggressive coarsening on the first time levels if the local problem size on each processor is very large, and instead of using factor-2 coarsening on all other levels we could gradually decrease the coarsening factor. Compared to the results for factor-2 coarsening in Figure 4.1, we see much more gradual increase in cost here, although still scaling with $\log(P)$; the increased problem size per processor in this example leads to a higher ratio of computation vs. communication, yielding better weak scaling. Comparing the three coarsening strategies for MGRIT F-cycles, the time curves look similar to those in Figure 4.3.

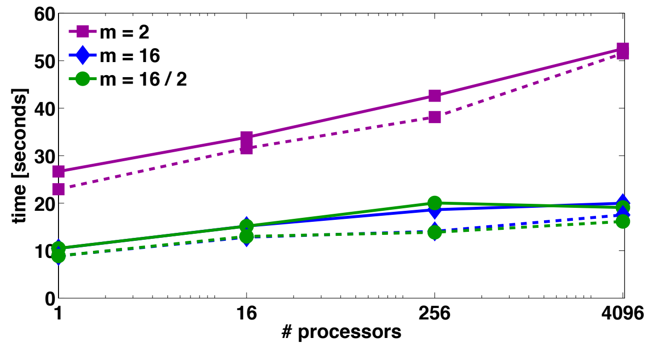


Fig. 4.3: Time to solve *Implicit2D($T = \pi^2/64$)* using the MGRIT V-cycle algorithm with our heuristic choice for the accuracy of the spatial solves, various relaxation schemes, and different coarsening strategies. The problem size per processor is about $(2^7)^2 \times 2^8$. Solid lines are results for using FCF-relaxation on all time levels and dashed lines represent runtimes for using F-relaxation on the finest grid and FCF-relaxation on all coarse grids.

4.3. Parallel performance (2D space). The above results show that particular choices for various components of the MGRIT algorithm lead to a more effective time-parallel method than other choices, but we would also like to know when it is beneficial to use two-level (Parareal) or multi-level MGRIT and how much speedup we can achieve over traditional space-parallel algorithms with sequential time stepping. To answer these questions, we consider two studies. First, we compare two-level and multi-level variants of MGRIT in a weak-scaling study up to 64K processors on Vulcan. We then consider strong scaling for a set of MGRIT variants and a space-parallel algorithm with sequential time stepping with an emphasis on comparing these time integration approaches. We consider the model problem in two space dimensions with implicit time discretization and choose the set of MGRIT variants based on the results in Sections 4.1 and 4.2. We use the heuristic described in §3.3.2 for the accuracy of the spatial solves and the factor-16/factor-2 aggressive coarsening strategy described above.

Figure 4.4 shows the weak-scaling study of the MGRIT algorithm on Vulcan, the Blue Gene/Q machine, for the problem with non-zero forcing. Here, we fix the temporal coarsening factor to be 16, and use the heuristic described above to control the solution of the spatial problems, varying only the two- vs. multi-level nature of the algorithm and the use of $F-$ vs. $FCF-$ (or $F-FCF-$) relaxation. While, for small processor counts, the two-grid method is slightly faster than the multigrid variants, we see that the V-cycle algorithms offer much better parallel scalability, with weak scaling efficiency of about 61% over 64K processors for the $F-FCF-$algorithm, and about 75% for the $FCF-$relaxation algorithm. In contrast, the two-level variants show strong growth in compute times already by 4096 processors. More scalable two-level variants arise by increasing the temporal coarsening factor as the number of points-in-time increases (as discussed in §4.5 below). For the $2049^2 \times 65,536$ problem considered at 65,536 processors, increasing the coarsening factor from 16 to 256 reduces the two-level wall-clock time from 2005 and 2861 seconds for $FCF-$ and $F-$relaxation, respectively, to 313 and 324 seconds. However, these are still markedly longer than the multilevel timings of 256 seconds for $FCF-$relaxation and 238 seconds for $F-FCF-$relaxation. Similar results are obtained for the problem *Implicit2D(*$T = \pi^2/8$*)-*F with $\delta t = 8(\Delta x)^2$, although the use of $F-FCF-$relaxation within the V-cycle algorithm requires only about 63% of the computing time that the V-cycle algorithm with $FCF-$relaxation does.

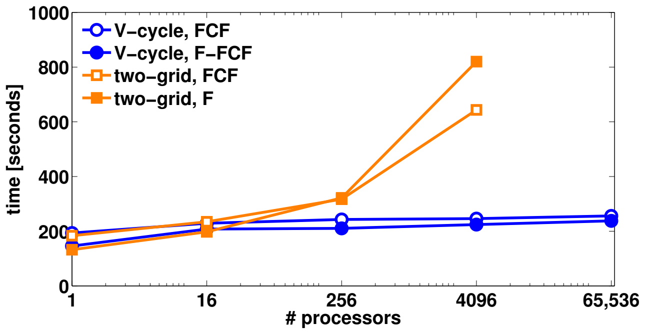


Fig. 4.4: Time to solve *Implicit2D(*$T = \pi^2/64$*)-*F using two-grid and multilevel V-cycle variants of the MGRIT algorithm on Vulcan, with coarsening by a factor of 16 in the temporal direction and spatial problems solved using the heuristic described above. The problem size per processor is about $\left(2^7\right)^2 \times 2^8$, with $\delta t = (\Delta x)^2$. Note the scaling of the time axis reflects the slower performance typical of the Blue Gene/Q architecture.

Figure 4.5 shows compute times for a strong-scaling study on a $129^2 \times 16{,}385$ space-time grid using a parallel algorithm with sequential time stepping and three MGRIT variants. For the time-stepping approach, we parallelize only in space, distributing the spatial domain such that each processor contains approximately a square in space. Since considering 16 processors for distributing the spatial domain minimizes the overall compute time when parallelizing only in space, for MGRIT, we parallelize over 16 processors in the spatial dimension, with increasing numbers of processors in the temporal dimension. More precisely, denoting the number of processors used for temporal parallelism by P_t, the space-time domain is distributed across $16P_t$ processors such that each processor owns a space-time hypercube of approximately $(2^5)^2 \times 16{,}384/P_t$. Considering a smaller number of processors, sequential time stepping is both faster and uses less memory (for sequential time stepping, one has to store data from one time step only, whereas for the MGRIT approach, a whole space-time subdomain, i.e., data from several time steps, needs to be stored). On a larger number of processors, however, MGRIT is faster. The choice of which algorithm to use, therefore, depends primarily on the available computational resources. More precisely, for this particular problem, the crossover point at which it becomes beneficial to use the MGRIT algorithm is at about 256 processors. Increasing the number of processors to 4096 results in a speedup of up to a factor of 10 compared to sequential time stepping.

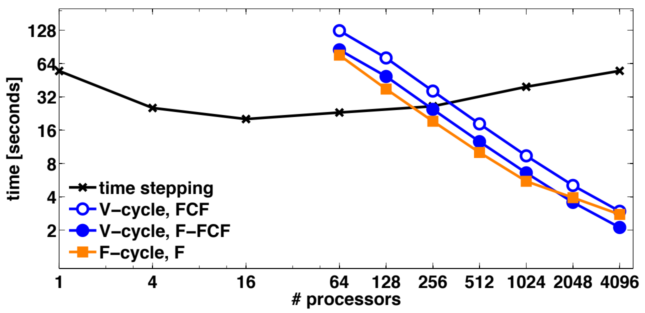


Fig. 4.5: Time to solve *Implicit2D($T = \pi^2$)* on a $129^2 \times 16{,}385$ space-time grid using sequential time stepping and three MGRIT variants.

Table 5 details the parallel efficiencies and speedups, relative to sequential time-stepping using 16 processors for spatial parallelism (which achieved the minimum time-to-solution of the time-stepping runs), for the three MGRIT variants. The parallel efficiencies for each MGRIT variant are measured relative to their runtime as time-serial processes, using 16 processors for spatial parallelism. These results show that each MGRIT variant obtains good parallel efficiency through $P_t = 64$, with some degradation for $P_t = 128$ and $P_t = 256$. All three variants show some speedup for $P_t = 64$, with F-cycles using $F-$relaxation showing best speedup there. However, the V-cycle strategies show better efficiency for larger values of P_t, with $F-FCF-$relaxation showing the best speedup with $P_t = 256$ (4096 total processors), nearly ten times faster than the fastest time with sequential time-stepping.

4.4. Parallel performance (3D space). Comparing the two time integration approaches for solving the model problem in three space dimensions with implicit time discretization, the time curves look similar to those in Figure 4.5, but the crossover point changes. Increasing the number of processors decreases the local problem size and, consequently, increases the ratio of boundary to domain or, equivalently, decreases the computation/communication ratio. For a particular computation/communication ratio, we need a much larger number of processors when considering three space dimensions instead of two since local problem sizes are larger.

P_t	1	2	4	8	16	32	64	128	256
V-cycle, FCF–relaxation									
Efficiency	100.0	95.7	94.9	83.8	83.4	82.5	80.2	74.4	63.6
Speedup	0.04	0.08	0.16	0.28	0.56	1.11	2.15	3.98	6.81
V-cycle, $F-FCF$–relaxation									
Efficiency	100.0	93.1	90.9	79.1	78.4	76.8	73.6	67.9	57.3
Speedup	0.07	0.12	0.24	0.41	0.82	1.60	3.07	5.66	9.55
F-cycle, F–relaxation									
Efficiency	100.0	92.3	88.0	89.2	87.2	83.5	76.0	53.4	37.9
Speedup	0.08	0.14	0.26	0.54	1.05	2.01	3.65	5.13	7.28

Table 5: Strong scaling efficiency and speedup of MGRIT variants. For each method, parallel efficiency is measured as $T(1)/(P_t * T(P_t))$, where $T(P_t)$ is the wall-clock time required for solution on P_t processors. Speedup is measured relative to the wall-clock time for sequential time-stepping with 16 processors used for spatial parallelism.

One possibility to benefit from the MGRIT approach at small scales is to consider a smaller spatial problem size. Figure 4.6 shows the compute times for a strong-scaling study on a $33^3 \times 4097$ space-time grid using a parallel algorithm with sequential time stepping and three MGRIT variants. Analogously to the two-dimensional case, for the sequential time stepping approach, the spatial domain is distributed evenly such that each processor holds approximately a cube in space. Considering two processors for each spatial dimension yields a reasonable spatial domain and results in close to minimum overall compute time when parallelizing only in space. Therefore, for MGRIT the space-time domain is distributed across $8P_t$ processors such that each processor owns a space-time hypercube of approximately $(2^4)^3 \times 4096/P_t$. The crossover point for which it becomes beneficial to use MGRIT for this particular problem size is at about 256 processors. Increasing the number of processors to 4096 results in a speedup of up to a factor of six compared to sequential time stepping.

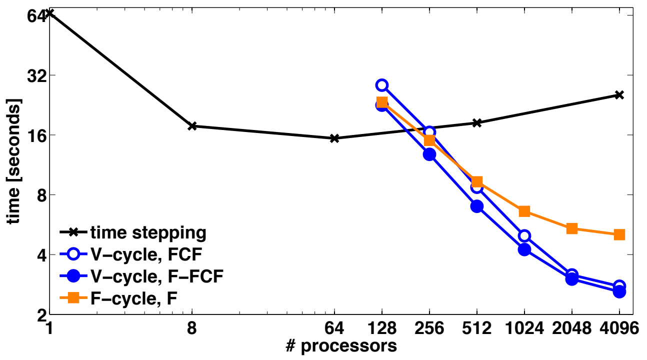


Fig. 4.6: Time to solve *Implicit3D($T = 4\pi^2$)* on a $33^3 \times 4097$ space-time grid using sequential time stepping and three MGRIT variants.

4.5. Parallel performance models. With current trends in computer architectures leading towards more processors, the MGRIT algorithm looks promising to speed up computations by adding parallelism in time. In this section, we consider three models to understand parallel performance of the algorithm. First, we consider a simple computation, comparing the number of spatial time-stepping solves required for the MGRIT algorithm, comparing to the optimal count of N_t for serial time-stepping. Since this potentially neglects serial bottlenecks in the algorithm, we next consider a more detailed model of parallel performance for parareal and MGRIT variants. Finally, we develop a full parallel performance model based on standard communication and computation models, and investigate performance for two hypothetical modern architectures.

4.5.1. Counting spatial solves. To gain insight about what level of parallelism in time is needed to break even with time stepping, we can compare the number of time-stepping routine calls for both time integration approaches. Recall that for implicit time discretizations, these function calls correspond to spatial solves and thus, are the dominant cost. Since sequential time stepping requires one spatial solve at each time step, the number of time-stepping routine calls is N_t. For MGRIT V-cycles with FCF-relaxation, we consider a hierarchy of time levels. For simplicity, we first assume that we use the same accuracy for all spatial solves on all grid levels. Since relaxation and interpolation correspond to C-relaxation and F-relaxation, respectively, the number of time-stepping routine calls in MGRIT taking the grid hierarchy into account is approximately $\nu_t(2m/(m-1)+1)N_t$, where m is the coarsening factor and ν_t denotes the number of MGRIT iterations necessary to solve to a given accuracy. Thus, for MGRIT to break even with sequential time stepping, we need about a factor of $\nu_t(2m/(m-1)+1)$ more processors to add parallelism in time. For the model problem in two space dimensions with implicit time discretization considered in Table 1, for example, the number of MGRIT iterations for factor-2 coarsening is about 10 and thus, for this particular problem we need about 50 times as many processors for MGRIT to break even with sequential time stepping.

If we consider a specific problem and, thus, a specific time integrator, Φ, and if we choose a specific method for solving the spatial problems, we can better approximate the level of parallelism needed to break even with time stepping. For the model problem in three space dimensions with implicit time discretization, we assume that we solve the spatial problems in parallel using spatial multigrid V-cycles with coarsening by a factor of two in each dimension. Instead of counting time-stepping routine calls, we now count the number of spatial V-cycles. If we fix an accuracy for the spatial solves in the time stepping approach, the number of spatial V-cycle calls is given by the number of time steps multiplied by the number of spatial V-cycle iterations, $\nu_{x,\mathrm{ts}}$, necessary to solve to this accuracy. For the MGRIT approach, we assume that we use a fixed number of spatial V-cycles for approximating each spatial solve within relaxation and restriction on the coarse grids as well as within interpolation on all grids. If we use a fixed coarsening factor, m, on all time levels, the number of spatial V-cycle calls for all approximate spatial solves in the time grid hierarchy is about $2(m+1)/(m-1)$ multiplied by the number of time steps and the number of MGRIT iterations, ν_t, necessary to solve to a given space-time accuracy. Within relaxation and restriction on the finest grid, we consider solving the spatial problems to high accuracy. The number of spatial V-cycle calls for these spatial solves is about $2N_t$ multiplied by the number of MGRIT iterations and the number of spatial V-cycle iterations, $\nu_{x,\mathrm{MGRIT}}$, necessary to solve to the high accuracy. Thus, the number of spatial V-cycle calls for the two time integration approaches is given by

$$\nu_{x,\mathrm{ts}}N_t \qquad \text{(time stepping)}$$

and

$$\left(\nu_{x,\mathrm{MGRIT}} + \frac{m+1}{m-1}\right) 2\nu_t N_t \qquad \text{(MGRIT } V\text{-cycle with } FCF\text{-relaxation).}$$

For the problem in §4.4, we have $\nu_{x,\mathrm{ts}} = 12$, $\nu_t = 26$, $\nu_{x,\mathrm{MGRIT}} = 8$, and $m = 16$ (under some simplifying assumptions such as assuming that we coarsen by a factor of 16 on all levels). In this case, the number of spatial V-cycle calls is $12N_t$ for the time-stepping approach and about $468N_t$ for the MGRIT approach. If we denote the number of processors used for temporal parallelism in MGRIT by P_t, the number of spatial V-cycle calls per processor is about $468N_t/P_t$. Thus, the sequential cost of $12N_t$ spatial V-cycle calls in time stepping and the parallel cost of $468N_t/P_t$ spatial V-cycles in MGRIT are equal for this problem if we use about 39 times as many processors for adding parallelism in time in the MGRIT approach. The results in §4.4 show that for eight-way parallelism in space, the crossover point at which it becomes beneficial to use MGRIT is at about 256 processors, corresponding to using 32 times as many processors for parallelizing in time for each spatial subdomain. Thus, counting the spatial V-cycle calls gives a good estimate for this problem. While this is a substantial factor of extra work, the excellent strong scaling observed for the MGRIT approach shows that it can be successfully amortized over many processors, leading to real speedup.

4.5.2. Accounting for serial bottlenecks. While counting the number of timestepping solves is an important indicator of the added cost of the MGRIT approach, it offers little insight into the true parallel performance of the algorithm. For an MGRIT hierarchy with L levels, we can estimate the wall-clock time, measured in units of spatial solves, required per iteration on levels $0 \le \ell < L$ as

$$T_\ell = 2\max(m, N_t/(m^\ell P_t)),$$

where the factor of 2 bounds the cost of FCF-relaxation per temporal point, the first term corresponds to serial integration in groups of m points within that relaxation, and the second term corresponds to dividing work on N_t/m^ℓ temporal points over P_t processors. On the coarsest level, we assume a serial solution of the problem with N_t/m^L temporal points.

For a two-grid method, such as parareal, $L = 1$, and we have a total cost of

$$T_{TG} = \nu_t \left(\max(m, N_t/P_t) + N_t/m\right),$$

where we discard the factor of 2 in T_ℓ above since F−relaxation alone is sufficient for two-level convergence. Assuming ν_t is independent of m, the optimal choice of m in the two-level context would make the coarse-grid problem as small as possible without adding a sequential bottleneck on either the fine or coarse grids, giving $m = \max(\sqrt{N_t}, N_t/P_t)$, and

$$T_{TG} = \nu_t \left(\max(\sqrt{N_t}, N_t/P_t) + \max(\sqrt{N_t}, P_t)\right). \tag{4.1}$$

This suggests the optimal choice of P_t in the two-grid context is $P_t = \sqrt{N_t}$, balancing all terms and giving

$$T_{TG} = 2\nu_t\sqrt{N_t}.$$

Note that when $P_t = \sqrt{N_t}$, then $m = \sqrt{N_t}$ as well, suggesting the optimal two-grid approach is to parallelize over $\sqrt{N_t}$ processors in the temporal direction, coarsening to a single point per processor on the coarse grid, just as in classical two-level domain decomposition.

For a full multilevel method, with $L = \log_m N_t$, the sequential phase of relaxation becomes a parallel bottleneck on levels where $m > N_t/(m^\ell P_t)$, or $\ell > \log_m(N_t/P_t) - 1$. Defining $K = \log_m(N_t/P_t)$, the full multigrid time comes from summing T_ℓ for $0 \le \ell \le L$, giving

$$\begin{aligned} T_{MG} &= \nu_t \left(2 \sum_{\ell=0}^{K-1} N_t/(m^\ell P_t) + 2 \sum_{\ell=K}^{L-1} m + 1 \right) \\ &\le \nu_t \left(4N_t/P_t + 2m \log_m P_t + 1 \right). \end{aligned}$$

Now, the optimal choice of m minimizes $m \log_m P_t$ which, for any P_t occurs when $\ln m = 1$; for practical purposes, we round this to $m = 3$, giving

$$T_{MG} \le \nu_t \left(4N_t/P_t + 6 \log_3 P_t + 1 \right). \tag{4.2}$$

Optimizing the choice of P_t in this expression gives $P_t = (2\ln(3)/3)N_t$, giving the bound

$$T_{MG} \le \nu_t \left(6 \log_3 \left((2\ln(3)/3)N_t \right) + 6/\ln(3) + 1 \right) \approx 6\nu_t \log_3 N_t.$$

Comparing (4.1) and (4.2), using the optimal coarsening factors for both two-grid and multi-grid variants, we see that we can make a direct comparison between work on the finest grid(s), where we expect the total work term, N_t/P_t, to dominate, and the coarsest grids, where sequential bottlenecks develop. If $N_t/P_t < \sqrt{N_t}$, then the finest-scale work is comparable, with the extra relaxation and levels in the MGRIT hierarchy contributing no more than a factor of 4 more work than the two-grid method with F-relaxation. On coarse scales, however, the sequential solve in the two-level scheme requires at least P_t work per iteration, while only $6 \log_3 P_t$ work is required per iteration in the multilevel approach. For even moderate values of P_t, this clearly benefits the multilevel variants. If we further assume enough parallel resources to optimally parallelize the two approaches, then the multilevel schemes can effectively use larger numbers of processors, and the ultimate wall-clock time bound grows like only $\log_3 N_t$ for the multilevel scheme, compared to $\sqrt{N_t}$ for the two-level approach. If parallel resources are limited, then the two-level method may be faster, such as in the case where $P_t = \sqrt{N_t}$, where $T_{MG} \approx 2T_{TG}$.

4.5.3. Parallel communication and computation model. The above arguments allow us to estimate the level of parallelism in time needed to break even with time stepping or a two-grid approach, but neglect important details such as message latency and network bandwidth. To give more precise estimates, we now develop parallel performance models based on the standard communication and computation models

$$T_{\text{comm}} = \alpha + n\beta, \qquad T_{\text{comp}} = n\gamma, \tag{4.3}$$

where α and β represent communication costs and γ is computation cost on a given machine. The numbers in [13, Table 2] can be used as the basis for choosing two parameter sets characterizing modern machines: a “computation dominant” set consisting of the parameters

$$\alpha = 1\ \mu\text{s}, \quad \beta = 10\ \text{ns/double}, \quad \gamma = 8\ \text{ns/flop}, \tag{4.4}$$

and a “communication dominant” set defined by

$$\alpha = 1\ \mu\text{s}, \quad \beta = 0.74\ \text{ns/double}, \quad \gamma = 0.15\ \text{ns/flop}. \tag{4.5}$$

The ratios α/β and α/γ are assumed to be “small” in the computation dominant set and “large” in the communication dominant set. To define the parameter sets (4.4) and (4.5), we have set $\alpha = 1\ \mu$s and chosen β and γ such that the ratios α/β and α/γ are equal to the minimum or maximum ratios from [13, Table 2], respectively.

Based on the two parameter sets (4.4) and (4.5), we compare the two time integration approaches. Analogously to the numerical experiments in §4.4, for the sequential time-stepping approach, we assume that the spatial domain is equally distributed such that each processor holds approximately a cube in space, and that the number of processors is increased only up to the point at which the spatial subdomain consists of about 2^3 points per processor. We consider a domain refinement of the problem in §4.4 meaning that instead of a space-time grid of size $33^3 \times 4097$ we assume a space-time grid of size $65^3 \times 16{,}385$. Assuming that $n_x = 2^4$ is a reasonable local problem size in each space dimension, we assume that the space-time domain is distributed across $64P_t$ processors such that each processor owns a space-time hypercube of about $(2^4)^3 \times 16{,}384/P_t$. Furthermore, we assume that we use a fixed coarsening factor on all levels that depends on the local number of time points on the finest grid as follows: if $16{,}384/P_t \geq 16$, we assume coarsening by a factor of 16, otherwise we consider factor-2 coarsening.

Figure 4.7 shows the predicted time to solve *Implicit3D($T = 4\pi^2$)* on a $65^3 \times 16{,}385$ space-time grid using sequential time stepping parallelized only in space and the predicted time to solution for applying MGRIT as functions of the number of processors used for the computations. The left plot shows the expected behavior based on the computation dominant parameters (4.4), and the right plot presents the expected behavior based on the communication dominant parameters (4.5). The time curves for both parameter sets show similar trends to those in numerical experiments, but as Figure 4.7 demonstrates, the expected crossover point and expected speedup depend on the parameter choices and, hence, the type of machine being used. In the computation dominant regime, the model predicts a speedup of up to a factor of about three, whereas in the communication dominant regime it is up to a factor of about 27. This result is attractive since, on future architectures, we expect the parameters to be most likely in the more communication dominant regime. Furthermore, comparing model predictions to numerical results for the problem in §4.4, the communication dominant model corresponds better to numerical results than the computation dominant model. We note that since the MGRIT approach relies on extra computation over time-stepping that is amortized in parallel, it is very sensitive to slow network performance relative to the computational work. Large increases in latency or decreases in bandwidth can certainly lead to situations where this work can no longer be effectively amortized, and the approach offers less value in these limits.

5. Conclusions.

With current trends in computer architectures leading towards systems with more, but not faster, processors, faster compute speeds must come from increased concurrency. Motivated by this challenge, a non-intrusive, optimal-scaling,

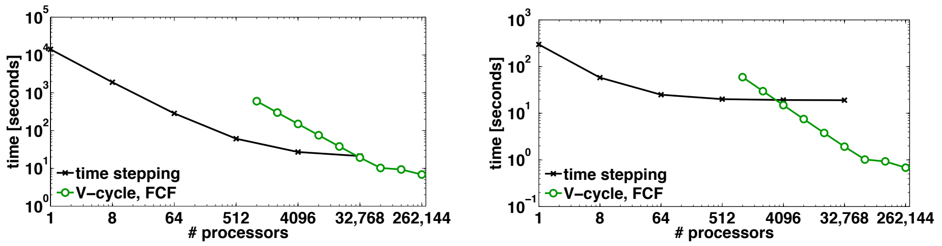


Fig. 4.7: Predicted time to solve *Implicit3D(T = $4\pi^2$)* on a $65^3 \times 16,385$ space-time grid using sequential time-stepping or MGRIT. At left, expected behavior based on the computation dominant parameters (4.4) and at right, expected behavior based on the communication dominant parameters (4.5).

time-parallel method is proposed. Being a non-intrusive approach which only uses an existing time propagator, this multigrid-reduction-in-time algorithm easily allows one to exploit substantially more computational resources than standard sequential time stepping. This is particularly important when moving to exascale, but it also enables benefits on smaller scales. For example, for problems that involve a large number of time steps, effective speedup is limited when allowing only spatial parallelism. Numerical results show that already on modern machines adding parallelism in time can sometimes significantly speedup computations when sufficient computational resources are available.

The parareal time integration method is equivalent to an optimal two-level variant of our MGR-based algorithm, MGRIT. However, the obvious generalization of parareal to multiple levels does not produce an optimal method. The MGR viewpoint makes it possible to see that replacing F-relaxation in the two-level parareal method with FCF-relaxation and applying the resulting method recursively produces an optimal multilevel algorithm.

For explicit time discretization schemes, stability issues on coarse grids can be circumvented while preserving optimality and non-intrusiveness by using an implicit discretization on coarse grids. However, a more general approach to time parallelization is to consider the space and time variables together by thinking of time as just another dimension of the problem. The drawback of space-time multigrid is that it will be more intrusive compared to MGRIT. On the other hand, space-time methods should have better performance properties and smaller memory requirements.

In this paper, only problems that yield constant-coefficient one-step time discretization methods are considered meaning that the time integration operator $\Phi_i(\cdot)$ in (2.2) corresponds to a matrix-vector product with a fixed matrix, $\Phi_i(\mathbf{u}_{i-1}) = \Phi\mathbf{u}_{i-1}$. Future work includes considering methods for variable-coefficient problems such that $\Phi_i(\mathbf{u}_{i-1}) = \Phi_i\mathbf{u}_{i-1}$. The generalization to this case is straightforward, but coefficient variability often creates additional difficulties for multigrid solvers. We will investigate the sensitivity of our methods to variations in Φ_i.

If the problem (2.1) is nonlinear, MGR ideas can be generalized to the full approximation storage (FAS) setting [6]. Future work also includes extending the MGR approach to the nonlinear setting. In fact, the parallel implementation of MGRIT is already based on the FAS approach. One interesting problem area we will consider is moving and/or adaptive meshes for which the F-cycle variant of MGRIT should prove useful.

Finally, future work includes broadening the applicability of MGRIT to hyperbolic problems. To this end, we have explored the simple linear advection equation in onedimensional space cross time. Here, initial results are promising when running a study analogous to Table 1, where we observe slowly growing iteration counts for F-cycles and FCF-relaxation. Future work will focus on scalability and more difficult problems, e.g., shocks.

REFERENCES

[1] *hypre: High performance preconditioners. http://www.llnl.gov/casc/hypre/.*

[2] P. Amodio and L. Brugnano, *Parallel solution in time of ODEs: some achievements and perspectives*, Appl. Numer. Math., 59 (2009), pp. 424–435.

[3] S. F. Ashby and R. D. Falgout, *A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations*, Nuclear Science and Engineering, 124 (1996), pp. 145–159. UCRL-JC-122359.

[4] R. Bank and K. Smith, *The incomplete factorization multigraph algorithm*, SIAM J. Sci. Comput., 20 (1999), pp. 1349–1364.

[5] ———, *An algebraic multilevel multigraph algorithm*, SIAM J. Sci. Comput., 23 (2002), pp. 1572–1592.

[6] A. Brandt, *Multi–level adaptive solutions to boundary–value problems*, Math. Comp., 31 (1977), pp. 333–390.

[7] J. Bulin, *Large-scale time parallelization for molecular dynamics problems*, Master's thesis, Royal Institute of Technology, 2013.

[8] A. Christlieb, R. Haynes, and B. Ong, *A parallel space-time algorithm*, SIAM J. Sci. Comput., 34 (2012), pp. C233–C248.

[9] A. J. Christlieb, C. B. Macdonald, and B. W. Ong, *Parallel high-order integrators*, SIAM J. Sci. Comput., 32 (2010), pp. 818–835.

[10] X. Dai and Y. Maday, *Stable parareal in time method for first- and second-order hyperbolic systems*, SIAM J. Sci. Comput., 35 (2013), pp. A52–A78.

[11] R. D. Falgout and J. E. Jones, *Multigrid on massively parallel architectures*, in Multigrid Methods VI, E. Dick, K. Riemslagh, and J. Vierendeels, eds., vol. 14 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, 2000, pp. 101–107. Proc. of the Sixth European Multigrid Conference held in Gent, Belgium, September 27-30, 1999. UCRL-JC-133948.

[12] H. Foerster, K. Stüben, and U. Trottenberg, *Nonstandard multigrid techniques using checkered relaxation and intermediate grids*, in Elliptic Problem Solvers, M. Schulz, ed., Academic, New York, 1981, pp. 285–300.

[13] H. Gahvari, A. Baker, M. Schulz, U. M. Yang, K. Jordan, and W. Gropp, *Modeling the Performance of an Algebraic Multigrid Cycle on HPC Platforms*, in 25th ACM International Conference on Supercomputing, Tucson, AZ, 2011.

[14] M. J. Gander and S. Güttel, *PARAEXP: a parallel integrator for linear initial-value problems*, SIAM J. Sci. Comput., 35 (2013), pp. C123–C142.

[15] M. J. Gander and E. Hairer, *Nonlinear convergence analysis for the parareal algorithm*, in Domain Decomposition Methods in Science and Engineering XVII, U. Langer, M. Discacciati, D. E. Keyes, O. B. Widlund, and W. Zulehner, eds., vol. 60 of Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 2008, pp. 193–200.

[16] M. J. Gander and S. Vandewalle, *Analysis of the parareal time-parallel time-integration method*, SIAM J. Sci. Comput., 29 (2007), pp. 556–578.

[17] I. Garrido, B. Lee, G. E. Fladmark, and M. S. Espedal, *Convergent iterative schemes for time parallelization*, Math. Comp., 75 (2006), pp. 1403–1428.

[18] C. W. Gear, *Parallel methods for ordinary differential equations*, Calcolo, 25 (1988), pp. 1–20.

[19] W. Hackbusch, *Parabolic multigrid methods*, in Computing methods in applied sciences and engineering, VI (Versailles, 1983), North-Holland, Amsterdam, 1984, pp. 189–197.

[20] G. Horton and R. Knirsch, *A space-time multigrid method for parabolic partial differential equations*, Parallel Computing, 18 (1992), pp. 21–29.

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

[22] K. R. Jackson, *A survey of parallel numerical methods for initial value problems for ordinary differential equations*, IEEE Trans. Magnetics, 27 (1991), pp. 3792–3797.

[23] D. Kamowitz and S. V. Parter, *On MGR[ν] multigrid methods*, SIAM J. Numer. Anal., 24 (1987), pp. 366–381.

[24] Z. Li, Y. Saad, and M. Sosonkina, *pARMS: A parallel version of the algebraic recursive*

multilevel solver, Numer. Linear Algebra Appl., 10 (2003), pp. 485–509.

[25] J. L. Lions, Y. Maday, and G. Turinici, *Résolution d'EDP par un schéma en temps pararéel*, C.R.Acad Sci. Paris Sér. I Math, 332 (2001), pp. 661–668.

[26] S. MacLachlan, T. Manteuffel, and S. McCormick, *Adaptive reduction-based AMG*, Numer. Linear Algebra Appl., 13 (2006), pp. 599–620.

[27] S. MacLachlan and Y. Saad, *Greedy coarsening strategies for nonsymmetric problems*, SIAM J. Sci. Comput., 29 (2007), pp. 2115–2143.

[28] Y. Maday, *The "parareal in time" algorithm*, in Sub-Structuring Techniques and Domain Decomposition Methods, F. Magoulès, ed., Computational Science, Engineering & Technology, Saxe-Coburg Publications, Stirlingshire, UK, 2010, ch. 2, pp. 19–44.

[29] C. Mense and R. Nabben, *On algebraic multi-level methods for non-symmetric systems—comparison results*, Linear Algebra Appl., 429 (2008), pp. 2567–2588.

[30] ———, *On algebraic multilevel methods for non-symmetric systems—convergence results*, Electron. Trans. Numer. Anal., 30 (2008), pp. 323–345.

[31] M. L. Minion, *A hybrid parareal spectral deferred corrections method*, Comm. App. Math. and Comp. Sci., 5 (2010), pp. 265–301.

[32] M. L. Minion and S. A. Williams, *Parareal and spectral deferred corrections*, in Numerical Analysis and Applied Mathematics, T. E. Simos, ed., no. 1048 in AIP Conference Proceedings, AIP, 2008, pp. 388â"–391.

[33] J. Nievergelt, *Parallel methods for integrating ordinary differential equations*, Comm. ACM, 7 (1964), pp. 731–733.

[34] Y. Notay, *Algebraic multigrid and algebraic multilevel methods: a theoretical comparison*, Numer. Linear Algebra Appl., 12 (2005), pp. 419–451.

[35] S. V. Parter, *On an estimate for the three-grid MGR multigrid method*, SIAM J. Numer. Anal., 24 (1987), pp. 1032–1045.

[36] M. Ries and U. Trottenberg, *MGR-ein blitzschneller elliptischer löser*, Tech. Rep. Preprint 277 SFB 72, Universität Bonn, 1979.

[37] M. Ries, U. Trottenberg, and G. Winter, *A note on MGR methods*, Linear Algebra Appl., 49 (1983), pp. 1–26.

[38] D. Ruprecht and R. Krause, *Explicit parallel-in-time integration of a linear acoustic-advection system*, Comput. & Fluids, 59 (2012), pp. 72–83.

[39] Y. Saad and B. Suchomel, *ARMS: an algebraic recursive multilevel solver for general sparse linear systems*, Numer. Linear Algebra Appl., 9 (2002), pp. 359–378.

[40] R. Speck, D. Ruprecht, M. Emmett, M. Bolten, and R. Krause, *A space-time parallel solver for the three-dimensional heat equation*, in Parallel Computing: Accelerating Computational Science and Engineering (CSE), M. Bader, A. Bode, H.-J. Bungartz, M. Gerndt, G. Joubert, and F. Peters, eds., vol. 25 of Advances in Parallel Computing, IOS Press, 2014, pp. 263–272.

[41] R. Speck, D. Ruprecht, M. Emmett, M. Minion, M. Bolten, and R. Krause, *A multi-level spectral deferred correction method*, arXiv preprint, (2013), pp. 1–30.

[42] H. D. Sterck, T. A. Manteuffel, S. F. McCormick, and L. Olson, *Least-squares finite element methods and algebraic multigrid solvers for linear hyperbolic PDEs*, SIAM J. Sci. Comput., 26 (2004), pp. 31–54.

[43] T. Weinzierl and T. Köppl, *A geometric space-time multigrid algorithm for the heat equation*, Numer. Math. Theory Methods Appl., 5 (2012), pp. 110–130.

[44] D. E. Womble, *A time-stepping algorithm for parallel computers*, SIAM J. Stat. Comput., 11 (1990), pp. 824–837.