

MULTIGRID REDUCTION IN TIME FOR NONLINEAR PARABOLIC PROBLEMS

R.D. FALGOUT ^{*}, T.A. MANTEUFFEL [†], B. O'NEILL [†], AND J.B. SCHRODER ^{*}

Abstract. The need for parallelism in the time dimension is being driven by changes in computer architectures, where performance increases are now provided through greater concurrency, not faster clock speeds. This creates a bottleneck for sequential time marching schemes because they lack parallelism in the time dimension. Multigrid Reduction in Time (MGRIT) is an iterative procedure that allows for temporal parallelism by utilizing multigrid reduction techniques and a multilevel hierarchy of coarse time grids. MGRIT has been shown to be effective for linear problems, with speedups of up to 50 times. The goal of this work is the efficient solution of nonlinear problems with MGRIT, where efficiency is defined as achieving similar performance when compared to an equivalent linear problem. The benchmark nonlinear problem is the p -Laplacian, where $p = 4$ corresponds to a well-known nonlinear diffusion equation, and $p = 2$ corresponds to the standard linear diffusion operator, our benchmark linear problem. When solving a linear problem using implicit methods and optimal spatial solvers, e.g. classical multigrid, the spatial multigrid convergence rate is bounded across temporal levels, despite a large variation in time step sizes. This is not the case for nonlinear problems, where the cost of a nonlinear solve increases dramatically on coarser time grids. This is the key difficulty explored by this paper. By using a variety of strategies, most importantly, an alternate initial guess for the nonlinear time-step solver and spatial coarsening, the average cost per time step evaluation is reduced over all temporal levels to a range similar to those of a corresponding linear problem. This allows for parallel scaling behavior comparable to the corresponding linear problem.

Key words. multigrid, multigrid-in-time, parabolic problems, nonlinear, reduction-based multigrid, parareal, high performance computing

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

1. Introduction. Previously, increasing clock speeds allowed for the speed-up of sequential time integration simulations of a fixed size, and allowed simulations to be refined in both space and time without an increase in overall wall-clock time. However, increases in clock speed have stagnated, leading to a sequential time integration bottleneck. Future increases in compute power will be available from more concurrency, and hence, speedups for time marching simulations must also come from increased concurrency.

By allowing for parallelism in time, much greater computational resources can be brought to bear and overall speedups can be achieved. Because of this, interest in parallel-in-time methods has grown over the last decade. Perhaps the most well known parallel in time algorithm, Parareal [23], is equivalent [13] to a two-level multigrid scheme. This work focuses on the multigrid reduction in time (MGRIT) method [10]. MGRIT is a true multilevel algorithm and has optimal parallel communication behavior, as opposed to a two-level scheme, where the size of the coarse-level limits concurrency.

Work on parallel-in-time methods actually goes back at least 50 years [30] and includes a variety of approaches. Work regarding direct methods includes [29, 32, 25, 6, 14]. There are iterative approaches, as well, based on multiple shooting, domain decomposition, waveform relaxation, and multigrid, including [21, 15, 24, 1, 16, 17, 34, 5, 35, 33, 19, 18, 23, 7, 28, 9, 36, 10]. For a gentle introduction to this history, please

^{*}Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-JRNL-692258

[†]Department of Applied Mathematics, University of Colorado at Boulder, Boulder, Colorado

see the review paper [12]. This work focuses on multigrid approaches (and MGRIT in particular) because of multigrid’s optimal algorithmic scaling for both parallel communication and number of operations. An additional attraction of MGRIT is its non-intrusive nature, where the user employs an existing sequential time-stepping routine within the context of the MGRIT implementation. This work uses XBraid [37], an open source implementation of MGRIT developed at Lawrence Livermore National Laboratory (LLNL).

MGRIT solves, in parallel, a general first-order, ordinary differential equation (ODE) and corresponding time discretization:

$$(1.1) \quad u_t = f(u, t), \quad u(0) = u_0, \quad t \in [0, T],$$

$$(1.2) \quad u(t + \delta t) = \Phi(u(t), u(t + \delta t)) + g(t + \delta t),$$

where Φ is a nonlinear operator that represents the chosen time stepping routine and g is a time dependent function that incorporates all the solution independent terms. In the linear case, the application of Φ is either a matrix vector multiplication, e.g. forward Euler, or a spatial solve, e.g. backward Euler.

Sequential time marching schemes are optimal in that they move from time $t = 0$ to $t = T$ using the fewest possible applications of Φ . By applying Φ iteratively, in comparably expensive but highly parallel multigrid cycles, MGRIT sacrifices additional computation for temporal concurrency. Both methods are optimal, i.e. $O(N)$, but the constant for MGRIT is higher. This creates a crossover point wherein the added concurrency overcomes the extra computational work. Beyond this crossover point MGRIT provides a speedup over sequential methods.

Application of MGRIT to linear parabolic problems was studied in [10]. Figure 1 shows a strong scaling study of MGRIT for linear diffusion on the machine Vulcan, an IBM BG/Q machine at LLNL. The problem size was $(257)^2 \times 16385$ (space \times time). Three data sets are presented, a standard sequential time-stepping run, a time-only parallel run of MGRIT, and a space-time parallel run of MGRIT. The space-time parallel runs used an 8×8 processor grid in space, with all additional processors added in time. Both MGRIT curves represent the use of temporal and spatial coarsening, so that the ratio of $\delta t/h^2$ is fixed on coarse time-grids, where h is the spatial mesh width. The maximum speedup achieved by the blue curve is approximately 50, and the crossover point where MGRIT provides a speedup is at about 128 processors in time. The goal of this paper is to make the overall performance (i.e., crossover point and speedup) of MGRIT for nonlinear problems similar to that for linear problems.

When considering the performance of MGRIT, the application of Φ is the dominant process. For a linear problem with implicit time stepping, each application of Φ equates to solving one linear system. When an optimal spatial solver such as classical spatial multigrid [26, 4, 31, 20] is used, the work required for a time step evaluation is independent of the time step size. However, when Φ is nonlinear, each application of Φ is an iterative nonlinear solve and when using a common method such as Newton’s, the conditioning usually depends on the time step size. To explore the effects of introducing a nonlinearity, a model nonlinear parabolic problem, known as the p -Laplacian, is considered:

$$(1.3) \quad u_t(\mathbf{x}, t) - \nabla \cdot (|\nabla u(\mathbf{x}, t)|^{p-2} \nabla u(\mathbf{x}, t)) = b(\mathbf{x}, t), \quad \mathbf{x} \in \Omega, \quad t \in [0, T],$$

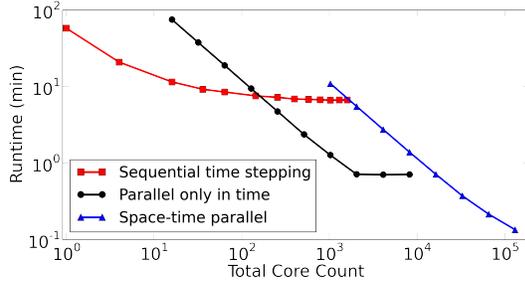


Fig. 1: Time to solve 2D linear diffusion on a $(128)^2 \times 16385$ space-time grid using sequential time stepping and two different processor decompositions of MGRIT. [10]

subject to the following Neumann boundary and initial conditions:

$$(1.4) \quad |\nabla u(\mathbf{x}, t)|^{p-2} \nabla u(\mathbf{x}, t) \cdot \mathbf{n} = g(\mathbf{x}, t), \quad \mathbf{x} \in \partial\Omega, \quad t \in (0, T],$$

$$(1.5) \quad u(\mathbf{x}, 0) = u_0(\mathbf{x}), \quad \mathbf{x} \in \Omega.$$

The p -Laplacian for $p = 4$ is well-known as a means of modeling soil erosion and transport [2] and has also found uses in image processing (denoising, segmentation and inpainting) and machine learning (see [8] for an overview and [22] for an introduction). In this paper, the model nonlinear problem corresponds to $p = 4$, while the comparable linear problem corresponds to $p = 2$, which is the standard diffusion operator.

Our study will consist of investigating parallel-in-time for equation (1.3) in the context of XBraid and MGRIT. We will balance user concerns such as overall time-to-solution, memory use, and non-intrusiveness.

To begin the study, we consider a naive application of MGRIT to (1.3), where a large increase in the cost of a nonlinear solve (for Newton’s method) on the coarser temporal grids is observed. This is caused by the relatively large time steps (compared to the finest grid) and the associated poor initial guess to the Newton solver on the coarse levels. These increases counteract the strength of multigrid, where speedup is achieved by using cheap coarse grid problems to accelerate convergence on the fine grid. Therefore, our strategy is to minimize the cost of each nonlinear solve, which is measured with a cost estimate. The goal is to ultimately achieve similar efficiencies for the nonlinear and linear versions of (1.3).

To reduce the average cost of each Φ evaluation, an improved initial guess is investigated. A commonly used approach is to use the previous time step as the initial guess for the nonlinear solver. However, for large time steps, which are found on coarse time levels, this initial guess is poor. The iterative nature of MGRIT gives us another option at no additional computational cost. Instead, the approximate solution at the corresponding time point and MGRIT level, but from the previous MGRIT iteration, can be used. This reduces the average cost of a Newton solve to below that of an equivalent sequential time integration. However, this also creates a tension between memory usage and computational efficiency, because MGRIT need not store every point in time.

Following this, a spatial coarsening strategy is pursued to limit $\delta t/h^2$ on coarse time grids. The goal is to maintain good conditioning of Φ fixed over all time levels so that fewer Newton iterations per Φ evaluation are required on coarse levels. In addition, the smaller problem sizes drastically reduce coarse grid compute times.

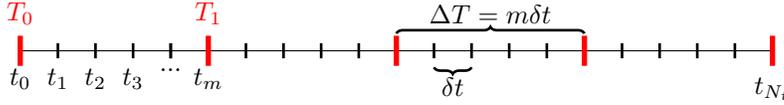


Fig. 2: Fine- and coarse-grid temporal meshes. Fine-grid points are present on only the fine-grid, whereas coarse-grid points (red) are on both the fine- and coarse-grid.

Two additional strategies include a progressive loosening of the Newton solve tolerance on coarser levels and avoiding unnecessary work on the first MGRIT cycle. Further speedups can be achieved by optimizing the number of levels in the MGRIT hierarchy and by loosening the Newton solver tolerance during the first three MGRIT iterations on all levels, where the approximate solution is still poor. Overall, the most effective strategies are spatial coarsening and the improved initial guess, but the other strategies combined have a similarly significant impact on runtime. Together, these strategies give a MGRIT algorithm for nonlinear problems that has an efficiency similar to that found for a corresponding linear problem.

In Section 2, the general MGRIT framework is discussed. In Section 3, some implementation details are given. In Section 4, our strategy for improving the performance of MGRIT is proposed and justified. In Section 5, this strategy is implemented. In sections 7.1 and 7.2, weak and strong scaling results are presented. When implemented using the strategies presented in this paper, the nonlinear implementation of the MGRIT algorithm achieves an efficiency similar to that found for linear problems.

2. MGRIT overview. First, a brief overview of the MGRIT algorithm for *time independent linear problems* is presented. The nonlinear, time dependent, extension follows in Section 2.1. Define a uniform temporal grid with time step δt and nodes t_j , $j = 0, \dots, N_t$ (non-uniform grids can easily be accommodated). Further, define a coarse temporal grid with time step $\Delta T = m\delta t$ and nodes $T_j = j\Delta T$, $j = 0, 1, \dots, N_t/m$, for some coarsening factor, m . This is depicted in Figure 2. In block triangular form, the time-stepping problem (1.2) is

$$(2.1) \quad A\mathbf{u} = \begin{bmatrix} I & & & & \\ -\Phi & I & & & \\ & & \ddots & \ddots & \\ & & & -\Phi & I \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N_t} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{N_t} \end{bmatrix} = \mathbf{g}.$$

Sequential time marching is a forward block solve of this system. MGRIT solves this system iteratively, in parallel, using a coarse-grid correction scheme based on multigrid reduction. Both are $O(N)$ methods, but MGRIT is highly concurrent. Multigrid reduction strategies are essentially approximate cyclic reduction methods and, as such, successively eliminate unknowns in the system. If the fine points are eliminated, the system becomes:

$$(2.2) \quad A_{\Delta}\mathbf{u}_{\Delta} = \begin{bmatrix} I & & & & \\ -\Phi^m & I & & & \\ & & \ddots & \ddots & \\ & & & -\Phi^m & I \end{bmatrix} \begin{bmatrix} \mathbf{u}_{\Delta,0} \\ \mathbf{u}_{\Delta,1} \\ \vdots \\ \mathbf{u}_{\Delta,N_t} \end{bmatrix} = R\mathbf{g} = \mathbf{g}_{\Delta}.$$

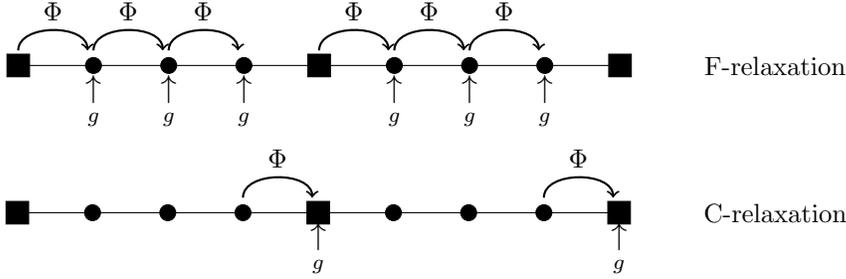


Fig. 3: F- and C-relaxation for coarsening by factor of 4

Given the reduction nature of the algorithm, the work [10] showed that the two-grid error propagator in the linear case with FCF-relaxation is

$$(2.5) \quad (I - PB_{\Delta}^{-1}RA)(P(I - A_{\Delta})R_I) = P(I - B_{\Delta}^{-1}A_{\Delta})(I - A_{\Delta})R_I.$$

This equation will be used later when spatial coarsening is introduced (see Section 5.3).

2.1. MGRIT algorithm for nonlinear problems. The linear MGRIT algorithm [10] is easily extended to the nonlinear setting using full approximation storage (FAS), a nonlinear multigrid scheme [3]. The FAS description of MGRIT first appeared in [11]. Note that the F-relaxation two-grid variant of nonlinear MGRIT is equivalent to the Parareal algorithm [13].

The nonlinear MGRIT algorithm is presented in Algorithm 1 as a two-level method, but can be used in a multilevel setting by recursively applying the algorithm at Step 4. Ideal interpolation (2.3) is carried out in two steps (7 and 8) for simplicity in implementation.

Prior to step 1, the initial guess at the fine grid C -points must be set. In general, the C -points are initialized with the best available estimate of the solution. Alternatively, the initial guess can be obtained using a full multigrid methodology. In this case, the fine grid C -points are obtained by interpolating a cheap coarse grid solution to the fine grid (see Section 6.1).

Algorithm 1 MGRIT($A, \mathbf{u}, \mathbf{g}$)

- 1: Apply F- or FCF-relaxation to $A(\mathbf{u}) = \mathbf{g}$.
 - 2: Inject the fine grid approximation and its residual to the coarse grid:
 $u_{\Delta,i} \leftarrow u_{mi}, \quad r_{\Delta,i} \leftarrow g_{mi} - (A(\mathbf{u}))_{mi}$.
 - 3: If Spatial coarsening, then
 $u_{\Delta,i} \leftarrow R_x(u_{\Delta,i}), \quad r_{\Delta,i} \leftarrow R_x(r_{\Delta,i})$.
 - 4: Solve $B_{\Delta}(\mathbf{v}_{\Delta}) = B_{\Delta}(\mathbf{u}_{\Delta}) + \mathbf{r}_{\Delta}$.
 - 5: Compute the coarse grid error approximation: $\mathbf{e}_{\Delta} \simeq \mathbf{v}_{\Delta} - \mathbf{u}_{\Delta}$.
 - 6: If Spatial coarsening, then
 $e_{\Delta,i} \leftarrow P_x(e_{\Delta,i})$.
 - 7: Correct \mathbf{u} at C -points: $u_{mi} = u_{mi} + e_{\Delta,i}$.
 - 8: If converged, then update F -points: apply F-relaxation to $A(\mathbf{u}) = \mathbf{g}$.
 - 9: Else go to step 1.
-

The reader will note that with exact arithmetic, MGRIT with FCF-relaxation

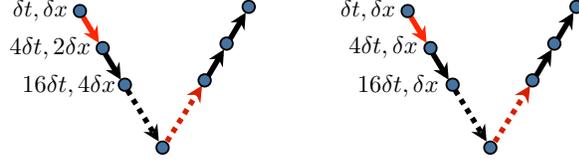


Fig. 4: Example multigrid V-cycle with space-time coarsening that holds $\delta t/h^2$ fixed on the left, and then time-only coarsening on the right.

propagates the initial condition two full coarse grid time intervals ($2\Delta t$) each cycle. Thus, MGRIT is equivalent to a sequential direct solve in $N_t/(2m)$ iterations. With F-relaxation only, the sequential solution is achieved in N_t/m iterations. The speedup comes from the fact that MGRIT converges in $O(1)$ iterations.

A variety of cycling strategies are available in multigrid (e.g., V, W, F). All results presented here use the standard V-cycle depicted in Figure 4. This corresponds to Algorithm 1 with the “Solve” step turned into a single recursive call. The recursion ends when a trivially sized grid, of, say, 5 time points, is reached. At this point a sequential solver is used. On the right in Figure 4, a V-cycle with only temporal coarsening is shown. Full spatial coarsening, depicted on the left, fixes the “parabolic” ratio, $\delta t/h^2$, on all levels, but can degrade the MGRIT convergence rate for the nonlinear problem considered here. Delayed spatial coarsening, described in Section 5.3, proved to be a more effective strategy.

The MGRIT algorithm is implemented in XBraid [37], an open source package developed at LLNL. XBraid conforms to MGRIT’s non-intrusive philosophy and requires the user to wrap an existing time-stepping routine, as well as define a few other basic operations like a state-vector norm and inner-product. The key computational kernel is the time-stepping (i.e., Φ) routine, but all the specifics are opaque to XBraid and done in user code. This allows the user to add temporal parallelism to existing time stepping codes with minimal modifications. For more details, see [10] and [37].

3. Model problem implementation. The weak form of equations (1.3)-(1.5) reads: find $u \in V^h$ such that

$$(3.1) \quad \langle u_t, v^h \rangle + \langle |\nabla u|^{p-2} \nabla u, \nabla v^h \rangle = \langle b, v^h \rangle + \langle g, v^h \rangle_{\partial\Omega}, \quad \forall v^h \in V^h,$$

where V^h is an appropriate finite element space. Discretizing in time using a backward Euler method and the temporal mesh in Figure 2 gives

$$(3.2) \quad \left\langle \frac{u_{k+1} - u_k}{\delta t}, v^h \right\rangle + \langle |\nabla u_{k+1}|^{p-2} \nabla u_{k+1}, \nabla v^h \rangle = \langle b_{k+1}, v^h \rangle + \langle g_{k+1}, v^h \rangle_{\partial\Omega}, \quad \forall v^h \in V^h,$$

where $k = 0, 1, \dots, N_t - 1$, and u_0 is the initial condition, given in (1.5), projected onto the finite element space. Define $\Psi(u)(v)$ and $f_k(v)$ to be

$$(3.3) \quad \Psi(u)(v) = \langle u, v \rangle + \delta t \langle |\nabla u|^{p-2} \nabla u, \nabla v \rangle,$$

$$(3.4) \quad f_k(v) = \langle u_k - \delta t b_{k+1}, v \rangle - \langle \delta t g_{k+1}, v \rangle_{\partial\Omega}.$$

Then, the final nonlinear weak form is: find $u_{k+1} \in V^h$ such that

$$(3.5) \quad \Psi(u_{k+1})(v^h) = f_k(v^h), \quad \forall v^h \in V^h, \quad k = 0, 1, 2, \dots, N_t - 1.$$

Each time step corresponds to the solution of this nonlinear system (i.e., the inversion of Ψ). The Fréchet derivative of $\Psi(u)(v)$, $\Psi'(u)(v)[w]$, is

$$(3.6) \quad \Psi'(u)(v)[w] = \lim_{a \rightarrow 0} \frac{\Psi(u - aw)(v) - \Psi(u)(v)}{a},$$

$$(3.7) \quad = \langle w, v \rangle + \delta t \langle [|\nabla u|^{p-2} + (p-2)(\nabla u)(\nabla u)^T] \nabla w, \nabla v \rangle.$$

Hence, Newton's method for (3.5) is

$$(3.8) \quad u_{k+1}^{j+1} = u_{k+1}^j - \delta u^j,$$

where the subscripts on u are time steps, the superscripts on u are the Newton iterations, and δu^j is the unique element of V^h such that

$$(3.9) \quad \Psi'(u_{k+1}^j)(v^h)[\delta u^j] = \Psi(u_{k+1}^j)(v^h) - f_k(v^h),$$

for every $v^h \in V^h$.

All tests were completed with $T = 4$ seconds and $\Omega = [0, 2]^2$ on a regular grid. The forcing function, $b(\mathbf{x}, t)$, was chosen such that the exact solution was

$$u(x, y) = \sin(\kappa x) \sin(\kappa y) \sin(\tau t),$$

where $\kappa = \pi$ and $\tau = (2 + 1/6)\pi$. Unless otherwise stated, the p -Laplacian was used with $p = 4$. The spatial discretization was computed using standard bi-linear quadrilateral elements and MFEM [27], a parallel finite element code.

3.1. Numerical parameters. The numerical testing parameters used throughout the paper (unless otherwise mentioned) were as follows. The Newton tolerance was fixed at 10^{-7} . The spatial solver for each Newton iteration was BoomerAMG from *hypre* 2.10.0b [20]. The BoomerAMG parameters were: HMIS coarsening (coarsen-type 10), one level of aggressive coarsening, symmetric L1 Gauss-Seidel (relax-type 8), extended classical modified interpolation (interp-type 6), and interpolation truncation equal to 4 nonzeros per row. The machine used for all numerical tests was Vulcan, an IBM BG/Q machine at LLNL.

Except for the scaling studies, the test problem size was a $(64)^2 \times 4096$ space-time grid on the domain $[0, 2]^2 \times [0, 4]$ using 1 processor in space and 64 processors in time. V-cycles and FCF-relaxation were employed in every test with a fixed stopping criteria of $10^{-9}/(\sqrt{\delta t} h)$. This allowed the same tolerance, relative to the fine-grid resolution, to be used in all cases. Note that this is an overly tight tolerance with respect to discretization error, set in large part because of the desire to investigate the algorithm's asymptotic convergence properties. The temporal coarsening factor was $m = 4$.

4. Establishing baselines. The goal of this paper is to show that, by following a few simple strategies, one can obtain an MGRIT algorithm for nonlinear problems that is as efficient as those previously seen for linear problems. To that end, an efficiency metric and two numerical baselines, through which all improvements will be measured, are now presented.

The MGRIT cost metric used here relies on $c_\ell^{(j)}$, the average cost of a Newton solve on grid level ℓ during MGRIT iteration j . This is a sensible choice because the key computational kernel for nonlinear problems is the Newton solve. Section 4.1 provides a detailed discussion of this metric and why it was chosen.

Two baseline tests were completed, the first being the sequential baseline. This baseline determined the average cost of a Newton solve when using a sequential solver, on a variety of space-time grids. An efficient implementation of MGRIT for nonlinear problems will match (or improve on) these cost estimates. The second baseline test is called the “naive” MGRIT baseline. For this baseline, MGRIT was applied to a nonlinear problem in an “out of the box” fashion. Typically, time stepping routines have a hard-coded initial guess equal to the previous time step and do not implement spatial coarsening. Hence, the “naive” MGRIT baseline mimics this. Note that this approach did, given enough temporal processors, provide a small speedup over the sequential routine (see Section 7.2); however, these speedups were much less than those seen for linear problems.

4.1. Cost Metric. We define the chosen cost metric, $c_\ell^{(j)}$, as the average cost of a Newton solve on grid level ℓ during MGRIT iteration j , i.e.,

$$(4.1) \quad c_\ell^{(j)} = a_\ell^{(j)} w_\ell,$$

where $a_\ell^{(j)}$ is the average number of Newton iterations required to take a nonlinear time step on level ℓ and iteration j , and w_ℓ is the number of spatial unknowns on level ℓ . This is the cost estimate of a single Φ application.

As motivation for this choice, let us examine a simple cost model for the entire MGRIT algorithm. Restriction from level ℓ to level $\ell + 1$ has the same cost as a C-relaxation on level ℓ . Likewise, interpolation from level ℓ to level $\ell - 1$ has the same cost as an F-relaxation on level ℓ . Hence, beyond the coarsest grid when using FCF-relaxation, each MGRIT level carries out 3 F-relaxations and 2 C-relaxations, for a total of $(2 + (m - 1)/m)N_{t_\ell}$ evaluations of Φ , where for simplicity N_{t_ℓ} , the number of time steps on level ℓ , is assumed to be an exact multiple of m . Next, consider a MGRIT V-cycle where level 0 is the finest and level L is the coarsest, ν is the number of V-cycles required for MGRIT to converge to within the residual tolerance and N_{t_ℓ} is the number of time steps at each level, again, assumed to be an exact multiple of m for simplicity. Then, the total cost of the MGRIT algorithm is

$$(4.2) \quad C(L, \nu) \approx \sum_{k=1}^{\nu} \left(N_{t_L} c_L^{(k)} + \sum_{\ell=1}^{L-1} c_\ell^{(k)} (2 + (m - 1)/m) N_{t_\ell} \right).$$

Clearly, equation (4.2) implies that minimizing the average cost of a Newton solve on each level and iteration will directly result in a reduction in the overall cost of the MGRIT cycle, and hence, $c_\ell^{(j)}$ can be used to measure the efficiency of the algorithm. Section 5 examines the two most important strategies for minimizing $c_\ell^{(j)}$, an improved initial guess for Newton and spatial coarsening.

While the metric shown here is $c_\ell^{(j)}$, another parallel performance issue is the variance in the cost of a Newton solve over the temporal domain. On the coarser grids, where a processor might own a single time step, synchronization effects imply that an F- or C-relaxation cannot complete until the slowest processor finishes. This in turn implies that the difference between the maximum and minimum cost per Newton solve would indicate synchronization problems. We therefore note that $c_\ell^{(j)}$ also tracks this spread between maximum and minimum for this problem, but since this fact is problem dependent, we draw the reader’s attention to it.

δt	h	a_ℓ	w_ℓ	c_ℓ	δt	h	a_ℓ	w_ℓ	c_ℓ
1/1024	1/64	3.5	129^2	5.82e4	1/1024	1/64	3.5	129^2	5.82e4
1/256	1/64	4.1	129^2	6.82e4	1/256	1/64	4.1	129^2	6.82e4
1/64	1/64	9.5	129^2	1.18e5	1/64	1/64	9.5	129^2	1.18e5
1/16	1/32	8.6	65^2	3.63e4	1/16	1/64	12.3	129^2	2.05e5
1/4	1/16	11.0	33^2	1.20e4	1/4	1/64	13.7	129^2	2.23e5
1	1/8	9.5	17^2	2.75e3	1	1/64	13.8	129^2	2.30e5

(a) Delayed spatial coarsening

(b) No spatial Coarsening

Table 1: Baseline Newton solver costs for sequential time-stepping.

4.2. Sequential time-stepping baseline. We now establish the sequential time-stepping baseline. Table 1 shows estimates of the cost of a Newton solve on all of the space-time grids present in the space-time grid hierarchy when MGRIT is applied to the model problem outlined in Section 3.1. These estimates, calculated using equation 4.1, are the average cost of a Newton solve when using a sequential solver.

Table 1a depicts baseline estimates for MGRIT using delayed spatial coarsening, while Table 1b provides baseline estimates for MGRIT with no spatial coarsening. An efficient implementation of MGRIT will match (or improve on) these cost estimates across all levels. Table 1 indicates that the average cost of a Newton iteration is highly dependent on the space-time grid, with there being a considerable advantage to coarsening simultaneously in space and time. This is because spatial coarsening reduces both $a_\ell^{(j)}$ and w_ℓ . The decrease in $a_\ell^{(j)}$ is explained by considering a standard backward Euler time step,

$$(4.3) \quad \left(I - \frac{\delta t}{h^2} G \right) (u_{k+1}) = f(u_k),$$

where G is a nonlinear diffusion operator. As $\delta t/h^2$ increases, the nonlinear operator moves away from the identity, becoming more expensive to solve. Coarsening in space and time bounds this ratio, making the nonlinear solve needed when solving equation (4.3) cheaper on the coarse grid. A detailed investigation into using spatial coarsening is given in Section 5.3.

The other important strategy explored here to reduce the cost of Newton is to improve the initial guess (see Section 5.2). This importance is also evident in Table 1, where a_ℓ continues to increase with dt , even when spatial coarsening is used. This is because δt is increasing, thus making the initial guess to Newton (the previous time step) progressively worse. On the coarse levels, where δt is large, this is clearly a poor approximation to the solution. In Section 5.2, an alternate initial guess for Newton is pursued, not available to standard sequential time stepping, where the initial guess is the solution at that MGRIT level from the previous MGRIT iteration.

4.3. Naive MGRIT baseline. Next, the so called “naive” MGRIT baseline is presented. The naive implementation is an unmodified, “out of the box” type implementation that wraps a sequential time stepping routine without any MGRIT optimizations. In particular, the previous time step is used as the initial guess and

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	1.11e5	1.11e5	1.41e5	2.36e5	2.85e5	2.75e5
1	6.11e4	7.12e4	1.40e5	2.05e5	2.66e5	3.00e5
2	5.77e4	6.87e4	1.39e5	2.13e5	2.80e5	3.16e5
3	5.74e4	6.76e4	1.41e5	2.15e5	2.71e5	3.20e5
4	5.74e4	6.76e4	1.40e5	2.13e5	2.71e5	3.20e5
5	5.74e4	6.76e4	1.40e5	2.15e5	2.71e5	3.20e5

Table 2: Cost estimates, $c_\ell^{(j)}$, for the naive MGRIT baseline (solver 0). The cost estimates are given in raw, unscaled form across each temporal level and MGRIT iteration.

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1	Total ($m = 4$)
0	8.88e5	2.22e5	2.82e5	4.72e5	5.70e5	5.50e5	2.98e6
1	4.89e5	1.42e5	2.80e5	4.11e5	5.32e5	6.00e5	2.45e6
2	4.61e5	1.37e5	2.79e5	4.26e5	5.60e5	6.32e5	2.50e6
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

Table 3: For processes active on each temporal level, estimated average cost to carry out FCF-relaxation, based on Table 2.

spatial coarsening is not implemented. Table 2 depicts the cost estimates, $c_\ell^{(j)}$, for this setting inside an MGRIT cycle. Each δt (column) value corresponds to a temporal level, while the rows represent different XBraid iterations.

Due to increasing Newton iteration counts, the cost of a coarse grid Newton solve is, in some places, 5 times more expensive than a fine grid solve. This can lead to an inefficient method in parallel. Consider the case where each MPI process owns m points in time on the finest level, i.e., one CF-interval. In [10] it was shown that this is an efficient decomposition. On coarser levels, each process then owns at most one point in time. Table 3 gives, for the processes active on each temporal level, the estimated cost of an FCF-relaxation. On the finest grid, these numbers are $2m$ times the cost estimates in Table 2 because FCF-relaxation (for larger m) involves roughly $2m$ time step evaluations. Then, on coarser grids, the cost estimate is simply twice what is in Table 2 because each processor owns at most one point, and all F -points are relaxed twice.

In this setting, it is immediately clear how expensive Newton solves on coarse levels can dominate the cost of a V-cycle because the levels must be traversed in order on each processor (i.e., one can sum each row in Table 3 for a cost estimate of that V-cycle). Since the dominant cost of a V-cycle is relaxation, this row-sum given in the final column is then a cost estimate for each V-cycle. In contrast, for the linear setting (when using an optimal spatial solver), the work required would be independent of time step size. For instance, the last row of the Table 3 would read similarly to $4.61e5$, $4.61e5/m$, $4.61e5/m$, ... for a total of $1.04e6$ ($m = 4$). When targeting an MGRIT efficiency similar to a linear problem, this will be a key issue.

In conclusion, our goal is to minimize the average cost of a Newton solve, focusing

on the coarse grids. Controlling any growth in the cost of a Newton solve across temporal levels will be key to matching the results seen with MGRIT for linear problems. By itself, the naive application of MGRIT scales very poorly when placed alongside the comparable linear problem (see sections 7.1 and 7.2).

5. Efficient MGRIT for the model nonlinear problem. In this section a variety of approaches designed to make MGRIT more efficient for the chosen nonlinear model problem (1.3) are investigated. Improvements are measured by comparing to the naive MGRIT baseline of Section 4.3.

5.1. Solver ID table. Given the number of options considered, and to make discussion easier, each solver is given a numerical ID. Table 4 presents each solver considered and its runtime for the chosen test problem size. Each solver option is discussed in more detail in the following subsections. However, a brief description of each solver is presented here for the reader’s convenience.

Solver 0 refers to the naive MGRIT approach from Section 4.3. The column heading “Spatial Grids” refers to the number of spatial grids used and is the option introduced in Section 5.3. A value of 1 for “Spatial Grids” indicates no spatial coarsening, while 4 means that the finest spatial grid is coarsened 3 times for a total of 4 grids. The finest grid is 64×64 , so coarsening further in space is not advantageous. The difference between solvers 1 and 2 is that solver 1 delays spatial coarsening so that it begins on the fourth temporal grid. Solver 2 begins spatial coarsening immediately on the first coarse time grid. Remember, for this problem with 4096 time steps and $m = 4$, there are only 6 temporal grids. Given the bad effects on convergence visible from “no delay” in solver 2, unless otherwise mentioned, spatial coarsening is always delayed. See Section 5.3 for more details.

Solvers 3, 4, 5 and 6 correspond to the improved initial guess introduced in Section 5.2. Here, “PMI” means that the previous MGRIT iteration is used as the initial guess to each Newton solve. This happens either at every point, or at all the coarse grid points and the fine grid C -points, depending on whether all the points or only the C -points are stored.

Solvers 7 and 8 correspond to turning on the “Skip” option introduced in Section 6.1. There, the concept of skipping unnecessary work during the first MGRIT down cycle is introduced.

Solvers 9 and 10 correspond to having a “fixed” or “scaled” Newton tolerance on coarse grids, as introduced in Section 6.2. Essentially, the Newton tolerance is either fixed on all levels, or relaxed on coarse grids.

Solvers 11 and 12 correspond to having “Cheap first three iters” as introduced in Section 6.3. Here, the Newton tolerance is further relaxed during the first three MGRIT iterations.

Solvers 13, 14, 15 and 16 reduce the number of levels in the hierarchy by setting a larger “Coarsest grid size” as introduced in Section 6.4. For instance with $m = 4$, using a coarsest grid size of 16 instead of 4 removes the coarsest level in the hierarchy. This can improve the time-to-solution by avoiding cycling between very small grids.

5.2. MGRIT with an improved initial guess for Newton’s method. In Section 4.2, it was suggested that $c_\ell^{(j)}$ increases with the time step size (and hence MGRIT level) because the initial guess becomes increasingly inaccurate. Recall that using the previous time step as the initial guess for the nonlinear solver is a common approach. On coarse time grids this is a poor approximation to the solution. After MGRIT completes one iteration, the user has two choices; the previous time step, and

Solver ID	Spatial grids	PMI	Skip	Newton Tol	Cheap first 3 iters	Coarsest grid size	Runtime per iter	Iterations	Runtime
0	1	Never	no	fixed	no	4	211s	9	1898s
1	4	Never	no	fixed	no	4	135s	9	1215s
2	4 no delay	Never	no	fixed	no	4	74s	35	2577s
3	1	C points	no	fixed	no	4	142s	9	1278s
4	4	C points	no	fixed	no	4	106s	9	953s
5	1	Always	no	fixed	no	4	132s	9	1193s
6	4	Always	no	fixed	no	4	96s	9	864s
7	1	Always	yes	fixed	no	4	103s	8	826s
8	4	Always	yes	fixed	no	4	79s	9	708s
9	1	Always	yes	scaled	no	4	93s	8	746s
10	4	Always	yes	scaled	no	4	76s	9	681s
11	1	Always	yes	scaled	yes	4	80s	8	636s
12	4	Always	yes	scaled	yes	4	64s	9	574s
13	1	Always	yes	scaled	yes	16	85s	8	682s
14	4	Always	yes	scaled	yes	16	60s	9	536s
15	1	Always	yes	scaled	yes	64	111s	8	890s
16	4	Always	yes	scaled	yes	64	74s	9	668s

Table 4: Overall runtimes, iteration counts and average time per iteration for the various solver options, with a $(64)^2 \times 4096$ space-time grid.

the solution from the previous MGRIT iteration. As MGRIT converges, the solution from the previous iteration becomes an ever improving initial guess.²

The main drawback of using the previous MGRIT iteration is that the solution must be stored at all points in time. To limit memory usage, one can alternatively choose to store the solution at the fine grid C -points only and to then use the previous MGRIT iteration (PMI) as the initial guess whenever it is available. In this way, the PMI is used as the initial guess during all time steps, except those completed during fine grid F-relaxation, where the previous time step is used. This approach, referred to as PMI at C-Points, is used by solvers 3 and 4. Overall, storing all points as opposed to just C -points requires storing (roughly) a factor of m more points in time per processor. This creates a tension between memory usage and computational efficiency.

Table 5 shows $c_\ell^{(j)}$ over all levels and iterations, scaled entry-by-entry with the corresponding entry from the naive MGRIT baseline from Table 2. For example, entry

²As an implementation note, after some MGRIT iterations, the initial guess to Newton can become so good that it satisfies the Newton solver’s tolerance. However, MGRIT needs relaxation to update the solution, or face stagnation. Therefore, the Newton solver is forced to iterate at least until the nonlinear residual is reduced.

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	1.3298	1.2447	0.9988	0.7112	0.7719	0.8484
1	1.2724	0.9088	0.5683	0.5227	0.4925	0.6388
2	0.9769	0.7506	0.4066	0.4437	0.4869	0.5789
3	0.7304	0.5221	0.3412	0.4162	0.5079	0.5989
4	0.5855	0.4926	0.3392	0.4164	0.5202	0.5833
5	0.5797	0.4926	0.3388	0.4124	0.5092	0.5833

Table 5: Relative cost estimates, $c_\ell^{(j)}/c_{\ell,naive}^{(j)}$, for MGRIT with the improved initial guess at all points (solver 5), across each temporal level and MGRIT iteration. The solver setup is otherwise identical to that used for Table 2. Cost estimates are scaled entry-by-entry with $c_{\ell,naive}^{(j)}$, the naive MGRIT baseline costs from Table 2.

(2,3) in Table 5 has been scaled by entry (2,3) from Table 2. Values less than one indicate that the cost per Newton solve was reduced by using the PMI, while values greater than one indicate it increased. A large reduction in $c_\ell^{(j)}$ is seen across most temporal grids and iterations.

The only exception to this was on the finest two grids, during the first two iterations. In those cases, using the improved initial guess led to an increase in cost due to an increase in Newton iterations. This is not surprising. During the first few iterations the solution from the previous MGRIT iteration is either inaccurate or unavailable.³

Based on these results, unless an accurate initial guess is known, the previous time step should be used as the initial guess at all time points during the first iteration. On all subsequent iterations, whenever possible, the previous MGRIT iteration should be used.

Table 4 validates our strategy. Solvers 3 and 5, where the PMI is used as the initial guess to the Newton solver at all points and only at coarse points (*C*-points), show large reductions in overall runtime, a direct consequence of the cost reductions seen in Table 5. Moreover, there is no degradation in MGRIT convergence. Solvers 4 and 6, where the PMI is used in conjunction with spatial coarsening, are discussed in Section 5.4.

In conclusion, these results indicate that using the PMI as the initial guess is very beneficial, reducing the runtime by 37% when comparing solvers 0 and 5 in Table 4. For users with memory limitations, using the PMI only at *C*-points is a good option.

5.3. MGRIT with spatial coarsening. Motivated by results seen in Section 4.2 and Table 1a, spatial coarsening is added to the naive solver from Section 4, as indicated in Algorithm 1. In general, the user’s code defines the separate spatial interpolation and restriction functions $P_x()$ and $R_x()$.⁴ Here, the natural finite element spatial restriction operator (and its transpose) is used to interpolate between regularly refined spatial grids. This operator is provided by MFEM. Spatial interpolation

³ During the first iteration, at *C*-Points, the user supplied initial guess is returned as the solution from the previous MGRIT iteration. The user does not define an initial guess at *F*-Points, so the previous time step must be used.

⁴MGRIT semi-coarsens in time and is agnostic to the spatial discretization, thus spatial coarsening is an *extra* option available to the user.

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	1.0000	1.0090	1.0248	0.2574	0.05931	0.01241
1	1.0081	1.0070	0.9976	0.2456	0.05562	0.01177
2	1.0028	1.0000	1.0000	0.2241	0.05180	0.01078
3	1.0000	1.0000	1.0023	0.2184	0.05339	0.01067
4	1.0000	1.0000	0.9988	0.2201	0.05379	0.01067
5	1.0000	1.0000	1.0000	0.2184	0.05379	0.01067

Table 6: Relative cost estimates, $c_\ell^{(j)}/c_{\ell,naive}^{(j)}$, for MGRIT with spatial coarsening (solver 1), across each temporal level and MGRIT iteration. Spatial coarsening begins on the fourth time level, $\delta t = 1/16$. The previous time step is used as the initial guess at all points. The solver setup is otherwise identical to that used for Table 2. Cost estimates are scaled entry-by-entry with $c_{\ell,naive}^{(j)}$, the naive MGRIT baseline costs from Table 2.

is the scaled (by 1/4) transpose of restriction so that $R_x P_x \approx I$. The choice of spatial interpolation operators is an area of active research; however, it is not surprising that scaling so that RP resembles an oblique projection helps MGRIT convergence. A better choice here could lead to improved results below.

Using coarse spatial grids on the coarse time levels drastically reduces $c_\ell^{(j)}$. The trade-off is that the coarse grid solves are less accurate, which, in turn, can reduce the MGRIT convergence rate.

When used without spatial coarsening, one benefit of the two-grid MGRIT algorithm is that, given an exact coarse grid solution, interpolation yields an exact fine grid solution. Error introduced by restriction and interpolation between spatial meshes removes this property. Without this exactness, any error modes introduced, specifically error modes in the null space of the spatial restriction operator, must be damped solely by FCF-relaxation. In many cases this causes a degradation of the MGRIT convergence rate. More precisely, with spatial restriction and prolongation, the temporal two-grid error propagation operator from (2.5) becomes

$$(5.1) \quad P(I - P_x B_\Delta^{-1} R_x A_\Delta)(I - A_\Delta) R_I,$$

where B_Δ is now on a coarse spatial grid. Table 6 depicts the results with this idea and scales each $c_\ell^{(j)}$ with the corresponding cost estimates found using the naive MGRIT baseline in Table 2. To highlight the cost saving benefits of spatial coarsening, this table shows results that use the previous time step as the initial guess for the Newton solve. Section 5.4 discusses the combined effect of spatial coarsening and the improved initial guess, introduced in Section 5.2.

In addition to reducing Newton iterations on the coarse grids, spatial coarsening also reduced the number of spatial unknowns on those levels. The effect of this is highlighted in Table 6. On the coarsest grid, the average cost per Newton Solve is 100 times smaller than the cost to complete the same solve using the naive baseline. The benefit of these dramatic cost reductions is validated by the improved timings in Table 4. Here, solvers 0, 1 and 2 apply varying levels of spatial coarsening.

Solver 2, where spatial coarsening begins on the first coarse grid, leads to a degradation in MGRIT convergence. This degradation is still a subject of active research

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	1.3298	1.2477	1.0035	0.2538	0.0417	0.0084
1	1.2942	0.9439	0.5850	0.1246	0.0271	0.0077
2	1.0000	0.7748	0.4150	0.1067	0.0257	0.0073
3	0.7391	0.5197	0.3423	0.1054	0.0262	0.0072
4	0.5826	0.4926	0.3392	0.1067	0.0262	0.0072
5	0.5797	0.4926	0.3388	0.1054	0.0262	0.0072

Table 7: Relative cost estimates, $c_\ell^{(j)}/c_{\ell,naive}^{(j)}$, for MGRIT when using spatial coarsening and the improved initial guess (solver 6), across each temporal level and MGRIT iteration. The solver setup is otherwise identical to that used for Table 2. Cost estimates are scaled entry-by-entry with $c_{\ell,naive}^{(j)}$, the naive MGRIT baseline costs from Table 2. Spatial coarsening again begins on the fourth time level, $\delta t = 1/16$.

and likely relates to the discussion above regarding the scaling of spatial interpolation, and the information lost when moving to coarse levels as described in equation (5.1). For practical purposes, this solver is unusable as the convergence degradation continues for larger problems. This is unfortunate given that this approach has a much smaller runtime per iteration.

In solver 1, spatial coarsening is delayed until the fourth temporal level, limiting any degradation in the MGRIT convergence rate, while still allowing for a dramatic reductions in $c_\ell^{(j)}$ on the coarse grids. This leads to a large reduction in the overall cost and run-time. This is the strategy pursued in this paper and it has proven to be robust in our tests. Compared to the naive baseline (solver 0), delayed spatial coarsening (solver 2) provides a 36% percent improvement in overall runtime.

5.4. Combining the improved initial guess and spatial coarsening. Sections 5.2 and 5.3 introduced an improved initial guess and spatial coarsening. Individually these improvement reduced the overall runtime by 37% and 36%, respectively.

Table 7 gives a cost analysis of the MGRIT algorithm when these strategies are combined, using 4 levels of spatial coarsening and the PMI at all points. In this table each cost estimate is again scaled by the corresponding cost estimate calculated using the naive MGRIT baseline.

Solvers 4 and 6, from Table 4, show the effect of using both of these options. When compared to the “naive” baseline, solver 4, where the PMI is used only at C -points, gives a 50% reduction in runtime. Solver 6 uses the PMI at all points, giving a 54% reduction in runtime, the best result so far. There is no degradation in MGRIT convergence.

6. Further Cost Savings. While the improved initial guess and spatial coarsening are, by far, the most effective strategies presented, several more strategies for reducing the cost of MGRIT are now considered. The effect of these strategies is significantly smaller than that for spatial coarsening and the improved initial guess; thus we will now normalize the cost analysis tables with respect to the just previously considered solver configuration. For example, the next table in Section 6.1 is scaled entry-by-entry by the raw cost estimates corresponding to Table 7 from Section 5.4. The table after that in Section 6.2 is scaled entry-by-entry by the raw cost estimates corresponding to the table from Section 6.1, and so on. An entry greater than 1

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	0	2.2623	2.7090	1.5563	2.0275	2.6500
1	0.5747	0.7648	0.7601	2.9966	2.5188	1.8750
2	0.8530	0.9343	1.0778	0.1877	1.3303	1.0937
3	1.0470	1.1421	1.0724	1.0447	1.0076	0.9687
4	1.0945	1.0200	1.0069	0.9962	1.0000	1.0000
5	1.005	1.0000	1.0000	1.0000	1.0000	1.0000

Table 8: Relative cost estimates, $c_\ell^{(j)}/c_{\ell,7}^{(j)}$, for MGRIT when skipping work during the down-cycle during iteration 0 (solver 8), across each temporal level and MGRIT iteration. The solver setup is otherwise identical to that used for Table 7 (solver 6). Cost estimates are scaled entry-by-entry with $c_{\ell,7}^{(j)}$, the raw cost estimates used to generate Table 7.

represents a deterioration relative to the previous solver configuration, while a value less than 1 represents an improvement.

6.1. Skipping Unnecessary Work. Even with the improvements so far, $c_i^{(j)}$ remains large during the first three MGRIT iterations. Consider iteration 0 and the down-cycle and up-cycle parts of Figure 4. For the model problem, where no prior knowledge of the solution is available, it is clear that relaxation during the down cycle of iteration 0 provides no benefit. The first time that global information is propagated is during the coarse grid solve, and the subsequent up cycle, of iteration 0. Therefore, all relaxation, and in fact work of any kind, during the down-cycle of iteration 0 can be omitted. In this setting, the solution on the finest-grid is injected to the coarsest-grid and then serially propagated. Then, the solution is interpolated back to the finest-grid. This strategy is similar to full multigrid methods. In general, there may be times when this work should not be skipped. For example, when some a priori knowledge of the solution is available, and a useful initial guess is available.

Table 8 shows the cost analysis for this approach (solver 8) when it is added to the solver strategy from Section 5.4 (solver 6), where the PMI is used as the initial guess, along with four levels of spatial coarsening. The cost analysis is similar if spatial coarsening is not used. To better discern the improvements, $c_\ell^{(j)}$ is scaled by the corresponding raw cost estimates generated when using solver 6. Interestingly, skipping work actually increases the cost of a Newton solve in most places during the first few iterations. However, the corresponding solvers 7 and 8 in Table 4 show that this strategy reduces the overall runtime with no degradation in MGRIT convergence. The reduced runtime is due to the fact that the time-stepping routine is called far fewer times during iteration 0, despite being more expensive when it is called. For instance, the “0” denotes the fact that there is no work done on the first level for iteration 0, by far the most expensive level. On coarse levels, during iteration 0, only interpolation (F-relaxation) is performed.

6.2. Newton solver accuracy. Here, further updates are made to the solver from the last section. In this case, the Newton solver tolerance is loosened on the coarse grids. Reduction in the accuracy of the coarse grid solves proved to be an effective way of minimizing overall run times for linear problems [10]. In that case, the number of spatial V-cycles allowed by the coarse grid linear solver was capped.

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	0	0.9946	0.9869	0.9864	0.9683	0.9669
1	1	0.9320	0.8983	0.9779	0.9580	0.9333
2	1	0.9364	0.8796	9.4752	0.8997	0.8857
3	1	0.9419	0.8617	0.9053	0.8586	0.9032
4	1	0.9852	0.8472	0.9253	0.8759	0.9062
5	1	1.0000	0.8496	0.9253	0.8759	0.9062

Table 9: Relative cost estimates, $c_\ell^{(j)}/c_{\ell,8}^{(j)}$, for MGRIT when using a scaled Newton tolerance (solver 10) across each temporal level and MGRIT iteration. The solver setup is otherwise identical to that used for Table 8 (solver 8). Cost estimates are scaled entry-by-entry with $c_{\ell,8}^{(j)}$, the raw cost estimates used to generate Table 8.

Here, varying the solver tolerance across temporal levels is investigated, rather than capping iteration counts (see Remark 6.1).

One way to reduce work on the coarse grids is to loosen the Newton solver tolerance on a per-level basis, with looser tolerances corresponding to coarser grids. The Newton solver tolerance on the coarse grid scales as

$$(6.1) \quad tol = \min \left(m^\ell \left(\frac{h_\ell}{h_0} \right)^{2\ell} tol_f, 0.001 \right),$$

where $\ell = 0$ is the finest level, tol_f is the desired Newton tolerance on the finest grid and h_ℓ is the spatial mesh width on level ℓ . In this case, $tol_f = 1 \times 10^{-7}$. The scaling with m and h allows one to consistently compare Newton residual norms across spatially and temporally coarsened grids (the overall method is $O(\delta t, h^2)$).

Table 9 shows the cost analysis for this approach (solver 10) when it is added to the solver strategy from Section 6.1 (solver 8). The cost analysis is similar if spatial coarsening is not used. The cost estimates, $c_\ell^{(j)}$, are scaled by the corresponding raw cost estimates generated using the previous solver 8. This allows one to easily see the benefit of only this strategy. Examination of Table 4 (solvers 9 and 10) then validates the cost estimate heuristic, where the across the board reduction in $c_\ell^{(j)}$ leads to a decrease in runtime and no degradation in MGRIT convergence.

REMARK 6.1. *An alternative to the variable tolerance is to simply cap the number of Newton iterations on the coarse grids in a manner similar to that pursued for linear problems in [10]. With this strategy, the number of Newton iterations is allowed to vary on the fine grid, until the tolerance is met. But on coarse grids, the iteration count is capped. However, this is more dangerous in the nonlinear setting because each Newton iteration is not guaranteed to reduce the residual by a fixed amount, as opposed to the linear case in [10]. Indeed, our experiments have shown that this approach easily leads to degraded MGRIT convergence and must be tuned to each individual grid size. Therefore, it is not considered any further.*

6.3. Cheap initial iterates. The initial three iterates are the most expensive, with $c_\ell^{(j)}$ significantly higher on all levels. The solution at this point is still inaccurate, so another obvious modification is to reduce the Newton tolerance during the first three iterations. Our simple strategy here is to use $tol_f = 10^{-3}$ (as opposed to 10^{-7})

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	0	0.9411	0.9515	0.9541	0.9719	0.9658
1	0.7399	0.7291	0.6577	0.9435	0.9437	1.0000
2	0.7195	0.7535	0.6474	0.8578	0.8860	1.0000
3	1.0299	1.0088	1.0074	1.0000	1.0000	1.0000
4	1.0045	1.0000	1.0081	1.0000	1.0000	1.0000
5	1.0000	1.0000	1.0082	1.0000	1.0000	1.0000

Table 10: Relative cost estimates, $c_\ell^{(j)}/c_{\ell,9}^{(j)}$, for MGRIT when using the cheap initial iterate strategy (solver 12) across each temporal level and MGRIT iteration. The solver setup is otherwise identical to that used for Table 9 (solver 10). Cost estimates are scaled entry-by-entry with $c_{\ell,9}^{(j)}$, the raw cost estimates used to generate Table 9.

for the Newton tolerance strategy from equation (6.1) during the first three iterations. Thereafter, tol_f returns to 10^{-7} .

Table 10 shows the cost analysis for this approach (solver 12) when it is added to the previous solver strategy from Section 6.2 (solver 10). The cost analysis is similar if spatial coarsening is not used. The cost estimates, $c_\ell^{(j)}$, are scaled by the corresponding raw cost estimates generated using the previous solver 10, so that values less than 1 represent an improvement over Table 9. These scaled values do show the expected drop in the cost estimate during iterations 0, 1 and 2. This drop is especially large on the finest grid, where the decrease is by about 20-25%. Table 4 (solvers 11 and 12) validates this by showing a decreased runtime and no degradation in overall MGRIT convergence.

REMARK 6.2. *One other solver modification for improved performance is capping the number of inner linear iterations on coarse grids, similar to [10]. Here, the BoomerAMG solver is used for each Newton iteration and the number of BoomerAMG iterations can be capped. These tests use a cap of 8; however, this number can be lowered further (our experiments went as low as 5 with no noticeable degradation), but to avoid over-tuning the problem, 8 was used. For solvers 11 and 12 (and all higher numbers) this feature has been added. Given it's small impact, a solver ID number was not devoted to this change.*

6.4. Setting the coarsest grid size. The final algorithmic enhancement considered is the size of the coarsest grid. Given how relatively expensive the Newton solver is on coarse grids, the question naturally arises whether truncating the number of levels in the hierarchy can be beneficial. However, this involves a trade-off. When a level is truncated from the hierarchy, the expensive Newton solves are no longer done at that level and the communication involved with visiting that level is avoided. However, the sequential part of the algorithm increases because the coarsest grid size has now increased. Thus, the best coarsest grid size is naturally problem and machine dependent. So far, the coarsest grid size has been 4, but here, coarsest grids of size 16 and 64 as now also considered. Since changing the coarsest grid size changes the cost estimates very little, that data is omitted.

For the case of a coarsest grid size of 16 (solvers 13 and 14) Table 4 shows a significant speedup of 44s when using spatial coarsening, but a slow down of 19s for the case of no spatial coarsening. This is because spatial coarsening makes the spatial

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4
0	0	2.6385	2.5492	0.3723	0.07642
1	0.5501	0.4838	0.2543	0.2349	0.04057
2	0.6027	0.4965	0.2418	0.0807	0.02092
3	0.7595	0.5317	0.2985	0.0972	0.02310
4	0.6233	0.4923	0.2888	0.0983	0.02310
5	0.5798	0.4923	0.2888	0.0974	0.02310

Table 11: Relative cost estimates, $c_\ell^{(j)}/c_{\ell,naive}^{(j)}$, for MGRIT when using the most effective combination of the suggested improvements (solver 14), across each temporal level and MGRIT iteration. Cost estimates are scaled entry-by-entry with $c_{\ell,naive}^{(j)}$, the raw cost estimates used to generate Table 2.

grid on the coarsest grid much smaller, and hence, the sequential solve required on the coarsest level is much cheaper. In other words the penalty for a larger coarsest grid size is much smaller for the case of spatial coarsening.

For the case of a coarsest grid size of 64, (solvers 15 and 16 in Table 4), the extra work from the larger sequential component of the solve on the coarsest level swamps any benefit and the run times increase substantially for both solvers 15 and 16. Given that solver 14 is the best overall performing solver, a maximum coarse grid size of 16 is used in scaling studies below.

6.5. Most effective improvements. Not surprisingly, the largest overall reduction in runtime is seen when spatial coarsening and the improved initial guess where used in conjunction with the four improvements outlined in sections 6.1-6.4.

Table 11 gives a cost analysis of this approach (solver 14). In this table cost estimates are scaled entry-by-entry with the naive MGRIT baseline from Table 2. Levels 1 and 2 of the first iteration are the only entries for which $c_\ell^{(j)}$ increases. However, because the time-stepping routine is called far fewer times during iteration 0, despite being more expensive when it is called (see Section 6.1), the cost to complete iteration 0 here is far less than the cost to complete the first iteration of the naive baseline. Comparing to Table 7, so that the overall effect of the improvements from sections 6.1-6.4 can be measured, shows that most of the work saved is on the first two temporal levels (note that there is one less level in Table 11.) Regarding runtime, Table 4 validates our strategy. Solver 14 is 3.54 times faster than the naive baseline, a direct consequence of the across the board cost savings seen in Table 11.

While many improvements have been outlined, Table 4 makes it clear that two of the improvements, the improved initial guess and spatial coarsening, are the most important. This can be seen by comparing solver 0 to solver 5 (to see the impact of the improved initial guess), which shows a 37% improvement. Then, when solver 5 and solver 6 are compared (to see the impact of spatial coarsening), a further 28% improvement is seen. The other four strategies in concert combine for another 48% improvement (compare solver 6 with solver 14). The subsequent scaling studies, therefore, focus on solvers 0, 1, 6 and 14.

7. Scaling Studies. Previous sections have focused on producing the most efficient Newton solver as a proxy for MGRIT efficiency. Here, in an effort to validate that heuristic, parallel scaling studies are presented.

ID \ Grid:	$16^2 \times 256$	$32^2 \times 1024$	$64^2 \times 4096$	$128^2 \times 16384$	$256^2 \times 65536$
0	4	7	9	10	10
1	6	8	9	11	10
6	6	8	9	11	10
14	6	8	9	11	10

Table 12: Weak scaling study: MGRIT iteration counts.

7.1. Optimal multigrid scaling. In this subsection, to test the optimality of MGRIT for the chosen model problem, a domain refinement study is presented. This was completed in a manner similar to the way in which spatial multigrid optimality is tested experimentally. That is, the space-time domain was fixed ($[0, 2]^2 \times [0, 4]$) while the spatial and temporal resolution were scaled up, keeping $\delta t/h^2$ fixed. For runs using spatial coarsening, the number of levels of spatial coarsening was increased on each subsequent test, resulting in 4 levels of spatial coarsening on the largest space-time grid of $256^2 \times 65536$. The solvers considered are 0, 1, 6, 14, so that (respectively) the effects of spatial coarsening, the improved initial guess for the Newton solver, and all the other enhancements, can be examined.

Table 12 shows the number of MGRIT iterations required for each solver to reduce the MGRIT residual to within a fixed tolerance, across the range of weakly scaled space-time grids described above. In general, observed iteration counts appear bounded independently of problem size for all the solver options considered. Unfortunately, using this experiment for weak scaling timings requires more processors than our machine provides (131K processors are available). For example, consider a weak scaling experiment where the smallest problem size involves 8 compute nodes, with 16 processors per node, for a total of 128 processors. A base test case that involves some off-node communication is needed, hence the choice of 8 compute nodes. To maintain a constant problem size per node and a constant $\delta t/h^2$, the number of time points must quadruple as the spatial problem size is doubled. This corresponds to increasing the node count by a factor of 16. Thus, to obtain four data points, $128 * 16^3 = 524288$ processors would be required.

7.2. Strong scaling. Both MGRIT and sequential time stepping are $O(N)$ optimal, but the constant for MGRIT is larger. On the other hand, MGRIT allows for temporal parallelism. This leads to a crossover point, after which MGRIT is beneficial. To illustrate this, a strong scaling study of MGRIT, for the space-time grid of $(128)^2 \times 16384$, was completed. Figures 5a and 5b show the results. The plot for “ID=14, linear problem” corresponds to the comparable linear problem of $p = 2$ in equation (1.3). This allows one to compare MGRIT’s scaling for the nonlinear problem with the scaling for the corresponding linear problem. To generate the data for “ID=14, linear problem”, we simply set $p = 2$ in the code and make no other optimizations, e.g., the spatial matrix and solver are still built during every application of Φ , as is required in the nonlinear case, so that the comparison is fair.⁵

The other plots are for the sequential (“Time-stepping”) time-stepping code and

⁵The spatial matrix and solver actually only need to be built once as in [10], because $p = 2$ is a constant coefficient heat equation. But, this is not done here for the purposes of a fair comparison between the linear and nonlinear cases.

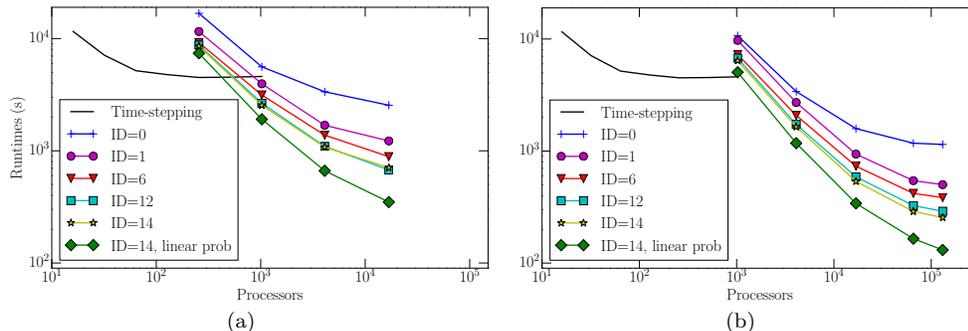


Fig. 5: Strong scaling study for a $(128)^2 \times 16385$ space-time grid, Left: 16 processors in space, $m = 16$, Right: 32 processors in space, $m = 4$.

solver ID’s 0, 1, 6, 12 and 14. This allows for a comparison of naive MGRIT (ID=0) with the effects of spatial coarsening (ID=1), the improved initial guess (ID=6) and the effects of all the other improvements (ID’s=12, 14).

Figure 5 depicts the results for $m = 16$ with 16 processors in space on the left, while the right shows the results for 32 processors in space and $m = 4$. The coarsening factor, m , changes to indicate that this is an important user parameter affecting the speedup observed and the overall number of processors used. For the $m = 16$ case, the crossover point at which MGRIT is beneficial is well below 1024 processors, or about 35 processors in time. The maximum speedup attained is a factor of 6.7 at 16384 processors. For the $m = 4$ case, the crossover point is at about 2000 processors, or about 500 processors in time. Yet, this smaller coarsening factor allows one to use more of the machine, and at 130K processors, the speedup is 18. The data points in each plot end when there are m points in time per processor, i.e., when using more processors in time provides no benefit. This is because FCF-relaxation is sequential over each interval of m points.

Figure 5 highlights another important aspect of the MGRIT algorithm. Increasing processors in space allows for greater scaling potential, but, given a fixed number of processors, it is often beneficial to bias the processor distribution towards temporal processors until m equals the number of time points per processor. A last important point is that, for the tests presented here, MGRIT converged in 9 iterations, but discretization error was reached after 5 iterations. Thus, the speedups presented here are largely understated. Halting MGRIT in a more automatic fashion after discretization error is reached is a topic of further research.

Compared to the strong scaling for the linear problem presented in Figure 1, these results are not as good. Optimal scaling would be represented by “straighter” lines, however, achieving this is difficult. The dataset for the linear heat equation (“ID=14, linear problem”) is very similar to the experiment in Figure 1. In fact, the only differences are: (1) bi-linear finite elements on a regular grid were used in space, as opposed to finite differencing; (2) the BoomerAMG solver in hypre was used, as opposed to the more efficient geometric-algebraic solver PFMG in hypre; and (3) the spatial discretization and spatial multigrid solver was built during every time step. However, in these tests the data set for the linear heat equation shows less than linear strong scaling. Improving strong scaling here will require addressing each of the three

differences described above. Difference (1) can be discounted because it does not change the sparsity pattern of the spatial operator, nor does it qualitatively change the convergence rate of the spatial multigrid solver. Thus, the most likely culprits for the strong scaling degradation are the parallel finite-element matrix assembly and the BoomerAMG setup-phase, although BoomerAMG is known to be an efficient spatial multigrid code. This is a topic for future research.

The goal of this paper is to show that, by implementing several simple strategies, an efficient implementation of the MGRIT algorithm for nonlinear problems is possible. As a measure of that efficiency, the MGRIT algorithm was compared against a similar implementation of a linear problem. This linear implementation, represented by the “ID=14, linear prob” plot, represents the strong scaling one would expect to see if, when using solver 14, each and every Newton iteration converged in exactly two iterations. The overall scaling behavior is similar, with deterioration for both at larger processor counts. However, MGRIT for the nonlinear problem does scale somewhat less well. This is due in large part to the phenomenon discussed with Table 3, where the relatively more expensive coarse grid solves for the nonlinear problem reduce scalability and increase the overall cost. The chief strategy under research now is to improve spatial coarsening so that it can begin on the first coarse grid, which will significantly reduce this effect.

It is important to note that, even at it’s worst, MGRIT for the nonlinear problem is only about 3 times slower than for the linear problem. This is a good result, considering MGRIT takes at least 2, but often many more, linear solves per time step (depending on level and iteration).

8. Conclusions. The MGRIT algorithm effectively adds temporal parallelism to existing sequential solvers and has been shown to be effective for linear problems [10]. However, when moving to the nonlinear setting, the relatively large time-step sizes on coarse grids make the application of MGRIT nontrivial. The proposed measures allow MGRIT to achieve similar performance to a comparable linear problem.

In summary, after the first iteration, the user should always use the solution from the previous MGRIT iteration as the initial guess to the nonlinear time-stepping routine (here, a Newton solver). When memory constraints inhibit this approach, using the previous MGRIT iteration whenever it is available still provides dramatic speedups. Thus, there is a tension between memory usage and computational efficiency. Secondly, spatial coarsening should be used whenever possible. For the linear example in Figure 1, spatial coarsening was implemented on all levels effectively. Tests showed that for the nonlinear model problem this was not the best strategy. However, delaying spatial resolution until the fourth temporal level both dramatically reduced the cost of a Newton solve on the coarser grids and limited degradation of the MGRIT convergence rate. These two changes gave the largest speedup. The other changes, when combined, also effected a significant speedup.

Weak scaling results showed that MGRIT is a scalable algorithm for nonlinear problems, with iteration counts bounded independently of problem size. Strong scaling showed the benefit of MGRIT, with an up to 18 times speedup seen over the corresponding sequential time stepping routine. The scaling is not as ideal as in [10], but with the modifications given here, similar performance was attained when compared to the corresponding linear problem.

REFERENCES

- [1] P. BASTIAN, J. BURMEISTER, AND G. HORTON, *Implementation of a parallel multigrid method for parabolic partial differential equations*, in Parallel Algorithms for PDEs, Proc. 6th GAMM Seminar Kiel, January 19-21, 1990, W. Hackbusch, ed., Braunschweig, 1990, Vieweg Verlag, pp. 18–27.
- [2] B. BJORN AND J. ROWLETT, *Mathematical models for erosion and the optimal transportation of sediment*, International Journal of Nonlinear Sciences and Numerical Simulation, 14 (2013), pp. 323–337.
- [3] A. BRANDT, *Multi-level adaptive computations in fluid dynamics*, 1979. Technical Report AIAA-79-1455, AIAA, Williamsburg, VA.
- [4] A. BRANDT, S. F. MCCORMICK, AND J. W. RUGE, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D. J. Evans, ed., Cambridge Univ. Press, Cambridge, 1984, pp. 257–284.
- [5] P. CHARTIER AND B. PHILIPPE, *A parallel shooting technique for solving dissipative ODEs*, Computing, 51 (1993), pp. 209–236.
- [6] ANDREW J. CHRISTLIEB, COLIN B. MACDONALD, AND BENJAMIN W. ONG, *Parallel high-order integrators*, SIAM Journal on Scientific Computing, 32 (2010), pp. 818–835.
- [7] H. DE 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.
- [8] A. ELMOATAZ, M. TOUTAIN, AND D. TENBRINCK, *On the p -laplacian and ∞ -laplacian on graphs with applications in image and data processing*, SIAM Journal on Imaging Sciences, 8 (2015), pp. 2412–2451.
- [9] M. EMMETT AND M. L. MINION, *Toward an efficient parallel in time method for partial differential equations*, Commun. Appl. Math. Comput. Sci., 7 (2012), pp. 105–132.
- [10] R. D. FALGOUT, S. FRIEDHOFF, Tz. V. KOLEV, S. P. MACLACHLAN, AND J. B. SCHRODER, *Parallel time integration with multigrid*, SIAM Journal on Scientific Computing, 36 (2014), pp. C635–C661.
- [11] R. D. FALGOUT, A. KATZ, Tz.V. KOLEV, J. B. SCHRODER, A. WISSINK, AND U. M. YANG, *Parallel time integration with multigrid reduction for a compressible fluid dynamics application.*, Lawrence Livermore National Laboratory Technical Report, LLNL-JRNL-663416, (2015).
- [12] M. J. GANDER, *50 years of time parallel time integration*, in Multiple Shooting and Time Domain Decomposition, T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, eds., Springer, 2015, pp. 69–114.
- [13] M. J. GANDER AND S. VANDEWALLE, *Analysis of the parareal time-parallel time-integration method.*, SIAM Journal on Scientific Computing, 29 (2007), pp. 556–578.
- [14] STEFAN GÜTTEL, *A parallel overlapping time-domain decomposition method for ODEs*, in Domain decomposition methods in science and engineering XX, vol. 91 of Lect. Notes Comput. Sci. Eng., Springer, Heidelberg, 2013, pp. 459–466.
- [15] W. HACKBUSCH, *Parabolic multigrid methods*, in Computing methods in applied sciences and engineering, VI (Versailles, 1983), North-Holland, Amsterdam, 1984, pp. 189–197.
- [16] G. HORTON, *The time-parallel multigrid method*, Comm. Appl. Numer. Methods, 8 (1992), pp. 585–595.
- [17] G. HORTON AND R. KNIRSCH, *A time-parallel multigrid-extrapolation method for parabolic partial differential equations*, Parallel Comput., 18 (1992), pp. 21–29.
- [18] G. HORTON AND S. VANDEWALLE, *A space-time multigrid method for parabolic partial differential equations*, SIAM J. Sci. Comput., 16 (1995), pp. 848–864.
- [19] G. HORTON, S. VANDEWALLE, AND P. WORLEY, *An algorithm with polylog parallel complexity for solving parabolic partial differential equations*, SIAM J. Sci. Comput., 16 (1995), pp. 531–541.
- [20] *HYPRE: High performance preconditioners*. <http://www.llnl.gov/CASC/hypre/>.
- [21] H. B. KELLER, *Numerical methods for two-point boundary-value problems*, Blaisdell Publishing Co. Ginn and Co., Waltham, Mass.-Toronto, Ont.-London, 1968.
- [22] P. LINDQVIST, *Notes on the p -laplace equation*, tech. report, Department of Mathematics, Ohio State University, 2006.
- [23] 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.
- [24] C. LUBICH AND A. OSTERMANN, *Multigrid dynamic iteration for parabolic equations*, BIT, 27 (1987), pp. 216–234.
- [25] YVON MADAY AND EINAR M. RØNQUIST, *Parallelization in time through tensor-product space-time solvers*, Comptes Rendus Mathématique. Académie des Sciences. Paris, 346 (2008), pp. 113–118.

- [26] S. F. MCCORMICK AND J. W. RUGE, *Multigrid methods for variational problems*, SIAM J. Numer. Anal., 19 (1982), pp. 924–929.
- [27] *MFEM: Modular finite element methods*. www.mfem.org.
- [28] 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. 388–391.
- [29] WILLARD L. MIRANKER AND WERNER LINIGER, *Parallel methods for the numerical integration of ordinary differential equations*, Math. Comp., 21 (1967), pp. 303–320.
- [30] J. NIEVERGELT, *Parallel methods for integrating ordinary differential equations*, Comm. ACM, 7 (1964), pp. 731–733.
- [31] J. W. RUGE AND K. STÜBEN, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., Frontiers Appl. Math., SIAM, Philadelphia, 1987, pp. 73–130.
- [32] DONGWOO SHEEN, IAN H. SLOAN, AND VIDAR THOMÉE, *A parallel method for time discretization of parabolic equations based on Laplace transformation and quadrature*, IMA Journal of Numerical Analysis, 23 (2003), pp. 269–299.
- [33] S. VANDEWALLE AND G. HORTON, *Fourier mode analysis of the multigrid waveform relaxation and time-parallel multigrid methods*, Computing, 54 (1995), pp. 317–330.
- [34] S. VANDEWALLE AND R. PIESSENS, *Efficient parallel algorithms for solving initial-boundary value and time-periodic parabolic partial differential equations*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1330–1346.
- [35] S. G. VANDEWALLE AND E. F. VAN DE VELDE, *Space-time concurrent multigrid waveform relaxation*, Ann. Numer. Math., 1 (1994), pp. 347–360. Scientific computation and differential equations (Auckland, 1993).
- [36] 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.
- [37] *XBraid: Parallel multigrid in time*. <http://11nl.gov/casc/xbraid>.