MULTIGRID REDUCTION IN TIME FOR NONLINEAR PARABOLIC PROBLEMS: A CASE STUDY

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. The key difficulty encountered is that the nonlinear time-step solver becomes progressively more expensive on coarser time levels, as the time-step size increases. To overcome such difficulties, multigrid research has historically targeted an accumulated body of experience regarding how to choose an appropriate solver for a specific problem type. To that end, this paper develops a library of MGRIT optimizations and modifications, most importantly an alternate initial guess for the nonlinear time-step solver and delayed spatial coarsening, that will allow many nonlinear parabolic problems to be solved with 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 speedup 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 speedups 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 [24], is equivalent [14] 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 [33] and includes a variety of approaches. Work regarding direct methods includes [32, 36, 27, 6, 15]. There are iterative approaches, as well, based on multiple shooting, domain

*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. This work was performed under the auspices of the U.S. Department of Energy under grant numbers (SC) DE-FC02-03ER25574 and (NNSA) DE-NA0002376 and Lawrence Livermore National Laboratory under contract B600360.

decomposition, waveform relaxation, and multigrid, including [22, 16, 25, 1, 17, 18, 38, 5, 39, 37, 20, 19, 24, 7, 31, 9, 40, 10]. For a gentle introduction to this history, please 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 [41], 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:

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

$$u(t+\delta t) = \Phi(u(t), u(t+\delta t)) + g(t+\delta t), \tag{1.2}$$

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 [10], i.e. $O(N)$ where N is the total number of time-steps, 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 (square markers), a time-only parallel run of MGRIT (circle markers), and a space-time parallel run of MGRIT (triangle markers). 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 (triangle markers) is approximately 50, and the crossover point where MGRIT provides a speedup is at about 128 processors in time. Achieving a similar efficiency for nonlinear parabolic problems is the chief goal of this paper.

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 multigrid in space [28, 4, 34, 21] is used, the work required for a time-step evaluation is independent of the time-step size. However, for nonlinear problems, each application of Φ involves an iterative nonlinear solve, and when using a common method such as Newton's method, convergence can depend strongly on the initial approximation even when linear multigrid is used as the inner solver. In particular, the previous time-step is farther away on coarser grids. To explore the effects of introducing a nonlinearity

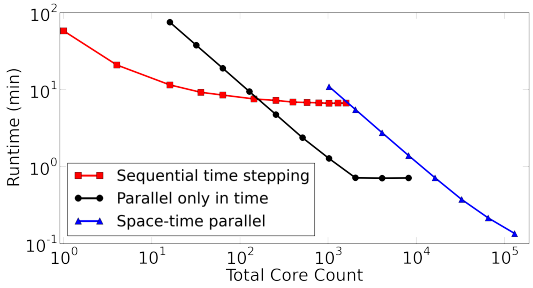

Fig. 1: Time to solve $2D$ linear diffusion on a $(128)^2 \times 16385$ space-time grid using sequential time-stepping and two different processor decompositions of MGRIT. [10]

that exhibits this key difficulty, a model nonlinear parabolic problem, known as the p-Laplacian, is considered:

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

subject to the following Neumann boundary and initial conditions:

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

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

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 [23] 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 desired result is to achieve similar efficiencies for the nonlinear and linear versions of (1.3), while also taking into account common user concerns, such as non-intrusiveness and storage costs. Ultimately, this paper's goal is to present a detailed, experimentally backed, library of optimizations and modifications that can be used to efficiently implement MGRIT for a large range of nonlinear parabolic problems. Thus, we have carefully chosen our model nonlinear parabolic problem and we note that multigrid research has historically targeted such an accumulated body of experience regarding how to choose an appropriate solver for a specific problem type.

To reduce the average cost of each Φ evaluation, an improved initial guess for each time-step is investigated. For sequential time-stepping, a commonly used approach

is to take the previous time-step as the initial guess. However, in the parallel-in-time setting this approach is flawed because on coarse levels, where the time-step size is large, this is a poor initial guess. The recent parallel-in-time works of [35, 26, 30] consider another option based on the iterative nature of many parallel-in-time algorithms, where information from previous parallel-in-time iterations is used as the initial guess. This increases storage requirements, but incurs no additional computational cost.

For MGRIT, this entails using as the initial guess the solution from the previous evaluation of Φ at each time point and on each MGRIT level. With this initial guess, as MGRIT converges, the cost of a Newton solve on each level goes to zero[1]. However, this also creates a tension between memory usage and computational efficiency because MGRIT need not store the solution from the previous Φ evaluation at all time points. Using the solution from the previous Φ evaluation as the initial guess whenever it is available allows a user with memory constraints to reap the benefits of this improved initial guess, while limiting per-processor storage costs. This reduced storage result and the importance of improved initial guesses for MGRIT are both important contributions of this work.

Following this, a spatial coarsening strategy is pursued to limit $\delta t/h^2$ on coarse time grids. The goal is to reduce the number of Newton iterations required on coarse levels by reducing the spatial variability of the nonlinearity while also improving the condition number of the inner linear problems. Recognizing this specific importance of spatial coarsening when controlling the cost of nonlinear time-step solvers on coarse time grids is an important contribution of this work. In addition, the smaller problem sizes drastically reduce coarse grid compute times and coarse grid storage costs.

Two additional strategies include avoiding unnecessary work on the first MGRIT cycle and optimizing the number of levels in the MGRIT hierarchy. Further speedups can be obtained 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 produce an 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.

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 $t_j,\ j = 0, \ldots, N_t$ (non-uniform grids can easily be accommodated). Further, define a coarse temporal grid with time-step $\Delta T = m\delta t$ and nodes $T_j = j\Delta T,\ j = 0, 1, \ldots, N_t/m$, for some coarsening factor, m. This is depicted in Figure 2. In block

[1] Assuming the Newton solver measures nonlinear convergence with a fixed absolute tolerance.

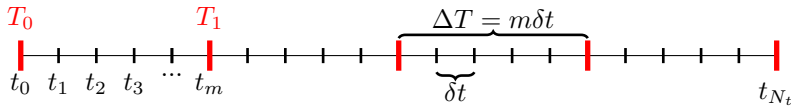

Fig. 2: Fine- and coarse-grid temporal meshes. Fine-grid points (black) are present on only the fine-grid, whereas coarse-grid points (red) are on both the fine- and coarse-grid.

triangular form, the time-stepping problem (1.2) is

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

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 a variation of cyclic reduction methods and, as such, successively eliminate unknowns in the system. If the fine points are eliminated, the system becomes:

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

"Ideal" restriction, R, and interpolation, P, are defined as in [10], so that the system (2.2) can be formed in a multigrid fashion. Let,

$$R = \begin{bmatrix} I & & & & & & & & \\ & \Phi^{m-1} & \dots & \Phi & I & & & & \\ & & & & & \ddots & & & \\ & & & & & & \Phi^{m-1} & \dots & \Phi & I \end{bmatrix}, \tag{2.3a}$$

$$P^T = \begin{bmatrix} I & \Phi^T & \dots & \Phi^{m-1,T} & & & & \\ & & & \ddots & & & & \\ & & & & I & \Phi^T & \dots & \Phi^{m-1,T} \end{bmatrix}. \tag{2.3b}$$

The interpolation injects at coarse points before extending those values to fine points, i.e., it is injection from the coarse- to fine-grid followed by F-relaxation (defined below).

With this, the "ideal" coarse-grid operator is $A_\Delta = RAP$.[2] This is referred to as ideal because the solution of (2.2) yields an exact solution at the coarse points. This

[2]We note that $RAP = R_I AP$, where R_I is injection at the coarse points. Thus for efficiency, injection is always used to map to the coarse-level (like Parareal). The exception is the *spatial coarsening* option where spatial restriction and interpolation functions, $R_x()$ and $P_x()$, are used to coarsen in space as well as time.

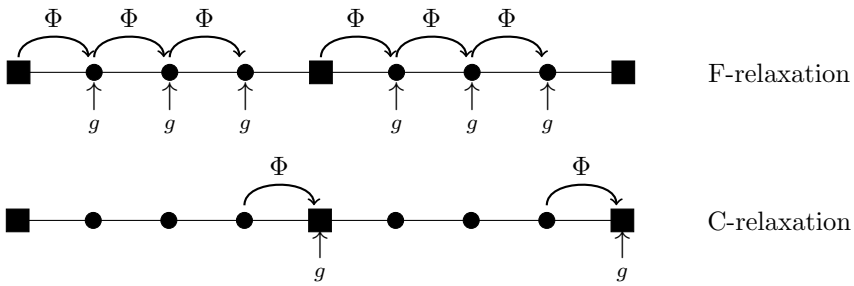

Fig. 3: F- and C-relaxation for coarsening by factor of 4

coarse grid is essentially a compressed version of the problem. If this is followed by interpolation, then the exact solution is also available at fine points. The limitation of this exact reduction method is that the coarse-grid problem is, in general, as expensive to solve as the original fine-grid problem (because of the Φ^m evaluations). Multigrid reduction methods address this by approximating A_Δ with B_Δ, where

$$B_\Delta = \begin{bmatrix} I & & & \\ -\Phi_\Delta & I & & \\ & \ddots & & \\ & & -\Phi_\Delta & I \end{bmatrix}, \tag{2.4}$$

and Φ_Δ is an approximate coarse-grid time-step operator. One obvious choice for defining Φ_Δ is to re-discretize the problem on the coarse grid so that a coarse-grid time-step is roughly as expensive as a fine-grid time-step. This paper makes that choice. For instance, with backward Euler, one simply uses a larger time-step size. Convergence of MGRIT is governed by the approximation, $A_\Delta \approx B_\Delta$, and this choice of using a re-discretization of Φ with $\Delta T = m\delta t$ has proved effective [10, 11]. It is important to note that, while the definition of this algorithm relies upon Φ and Φ_Δ, the internals of these functions need not be known. This is the non-intrusive aspect of MGRIT. The user defines the time-step operator and can wrap existing codes to work within the MGRIT framework. The coarse grid is used to compute an error correction based on the residual equation (see Algorithm 1). Relaxation, a local fine-grid process, is used to resolve fine-scale behavior. Figure 3 shows the actions of F- and C-relaxation on a temporal grid with $m = 4$. F-relaxation propagates the solution forward in time from each coarse point to the neighboring F-points. Overall, relaxation is highly parallel. Each interval of F-points can be updated independently during F-relaxation. Every C-point update (during C-relaxation) is similarly independent.

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 [14].

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.

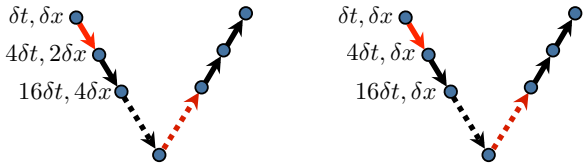

Fig. 4: Example multigrid V-cycle with space-time coarsening that holds $\delta t/h^2$ fixed on the left, and then time-only coarsening on the right.

Prior to running the algorithm, initial values for $\mathbf{u}$ 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 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, as shown in [10].

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 [41], 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 [41].

2.2. Storage Costs. The reader should also note the storage costs associated with MGRIT. At a minimum, the solution values u_i must be stored at each C-point in the temporal hierarchy (the first term in the storage cost model below). In addition, some number k of auxiliary vectors must be stored at all points on the coarse levels (the second term below). Let L be the number of temporal levels, p be the number of temporal processors, and s_l be the cost to store a vector on level l. Then the storage cost of MGRIT (per temporal processor) is as follows:

$$S_C(k) \approx \sum_{i=0}^{L-1} \left\lceil \frac{N_t}{m^{i+1}p} \right\rceil s_i + \sum_{i=1}^{L-1} k \left\lceil \frac{N_t}{m^i p} \right\rceil s_i. \tag{2.5}$$

By default, MGRIT stores two auxiliary vectors on coarse grids ($k = 2$): a restricted copy of the fine-grid solution and the right-hand-side for the coarse FAS system. However, in Section 5.2 we show how storing one additional auxiliary vector, specifically the most recent solution from the corresponding Φ evaluation, allows the user to dramatically reduce the overall runtime.

Equation (2.5) will be used to compute a memory multiplier that gives an estimate of the per processor increase in memory usage when compared to sequential time-stepping. The memory multiplier used is defined as $S_C(k)/s_0$.

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

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

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

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

where $k = 0, 1, \ldots, N_t - 1$, 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

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

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

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

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

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

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

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

Hence, Newtons method for (3.5) is

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

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

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

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

3.1. Numerical parameters. 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 [29], a parallel finite element code.

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 [21]. 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 4 processors in space and 128 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 develop experience with general strategies that one can use to obtain an MGRIT algorithm for nonlinear problems that is as efficient as those previously seen for linear problems. To that end, an efficiency metric for computational cost is now presented. Two numerical baselines, through which all improvements will be measured, are also given.

The MGRIT cost metric used here relies on $c_\ell^{(j)}$, the average cost of a Newton solve[3] 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

[3]By Newton solve, we mean the whole cost to take a nonlinear time step with Φ using the Newton solver, including its multiple inner Newton iterations.

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 the model 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 cost of a Newton solve (Φ application) on grid level ℓ during MGRIT iteration j, averaged over all time-steps on that level, during that MGRIT iteration, i.e.,

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

where $a_\ell^{(j)}$ is the average number of Newton iterations required to take a nonlinear time-step on level ℓ and MGRIT iteration j, and w_ℓ is the number of spatial unknowns on level ℓ. Thus, $c_\ell^{(j)}$ 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 an MGRIT V-cycle where level 0 is the finest and level L is the coarsest, and ν is the number of V-cycles required for MGRIT to converge to within the residual tolerance. Then, the total cost of the MGRIT algorithm is

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

Clearly, equation (4.2) implies that minimizing the average cost of a Newton solve on each level and MGRIT 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.

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

δt	h	a_ℓ	w_ℓ	c_ℓ
1/1024	1/32	2.13	65^2	9.00e3
1/256	1/32	2.85	65^2	1.20e4
1/64	1/32	4.18	65^2	1.77e4
1/16	1/16	6.52	33^2	7.10e3
1/4	1/8	10.8	17^2	3.12e3
1/1	1/4	6.5	9^2	5.26e2

(a) Delayed spatial coarsening

δt	h	a_ℓ	w_ℓ	c_ℓ
1/1024	1/32	2.13	65^2	9.00e3
1/256	1/32	2.85	65^2	1.20e4
1/64	1/32	4.18	65^2	1.77e4
1/16	1/32	7.77	65^2	3.28e4
1/4	1/32	13.4	65^2	5.66e4
1/1	1/32	9.0	65^2	3.80e4

(b) No spatial Coarsening

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

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,

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

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 from 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 δt, 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, where the initial guess is the solution obtained during a previous evaluation of Φ at the current time-point, and on the current MGRIT level, is pursued.

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 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, 6 times more expensive than a fine grid solve. This can lead to

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	2.17e4	2.10e4	2.20e4	3.16e4	5.86e4	4.92e4
1	1.08e4	1.24e4	1.76e4	3.52e4	5.98e4	5.41e4
2	8.81e3	1.19e4	1.70e4	3.35e4	6.06e4	5.73e4
3	8.77e3	1.17e4	1.72e4	3.38e4	5.94e4	5.73e4
4	8.72e3	1.17e4	1.72e4	3.39e4	5.94e4	5.73e4
5	8.72e3	1.17e4	1.72e4	3.39e4	5.98e4	5.73e4

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
0	1.73e5	4.20e4	4.39e4	6.32e4	1.17e5	9.83e4	5.38e5
1	8.62e4	2.48e4	3.51e4	7.05e4	1.20e5	1.08e5	4.44e5
2	7.05e4	2.38e4	3.41e4	6.71e4	1.21e5	1.15e5	4.31e5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

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

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. When targeting an MGRIT efficiency similar to a linear problem, this will be a key issue.

In conclusion, our goal is to present a library of MGRIT optimizations and modifications that can be extended to most nonlinear parabolic problems. These optimizations and modifications are developed with the goal of minimizing the average cost of a Newton solve, focusing on the coarse grids, while addressing common user concerns of non-intrusiveness and memory usage. 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. Also given is the per processor "memory multiplier" that gives an estimate of the per processor increase in memory usage (see Section 2.2), when compared to sequential time-stepping. The multiplier is given for the number of temporal processors $p = 128$ and $p = 4096$, which is the largest possible p for this test problem size. For $p = 128$, the multiplier is quite large, but the addition of more resources can make this number relatively small.

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, "IIG" means that the improved initial guess, specifically the solution from a previous evaluation of Φ, is used as the initial guess for 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. Regarding the memory multiplier, the use of IIG requires an additional vector stored on coarse levels, hence $k = 3$ (instead of 2) in Equation 2.5.

Solvers 7, 8, 9 and 10 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 11 and 12 correspond to having "Cheap first three iters" as introduced in Section 6.2. Here, the Newton tolerance is 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 6.3. 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. Section 4.2 showed that $c_\ell^{(j)}$ increases with the time-step size (and hence MGRIT level) because using the previous time-step as the initial guess for Newton's methods becomes increasingly inaccurate. Thus, this section explores an improved initial guess

Solver ID	Spatial grids	IIG	Skip	Cheap first 3 iters	Coarsest grid size	Memory mult., $p = 128 \to 4096$	Runtime per iter	Iterations	Runtime
0	1	Never	No	No	4	$40 \to 16$	$162s$	7	$1135s$
1	4	Never	No	No	4	$31 \to 7$	$101s$	7	$712s$
2	4*	Never	No	No	4	$12 \to 2$	$57s$	30	$1717s$
3	1	C points	No	No	4	$53 \to 21$	$91s$	7	$638s$
4	4	C points	No	No	4	$42 \to 10$	$82s$	7	$579s$
5	1	Always	No	No	4	$84 \to 21$	$92s$	7	$647s$
6	4	Always	No	No	4	$73 \to 10$	$84s$	7	$591s$
7	1	C points	Yes	No	4	$53 \to 21$	$64s$	7	$453s$
8	4	C points	Yes	No	4	$42 \to 10$	$58s$	7	$410s$
9	1	Always	Yes	No	4	$84 \to 21$	$61s$	7	$429s$
10	4	Always	Yes	No	4	$73 \to 10$	$55s$	7	$391s$
11	1	Always	Yes	Yes	4	$84 \to 21$	$56s$	7	$395s$
12	4	Always	Yes	Yes	4	$73 \to 10$	$51s$	7	$360s$
13	1	Always	Yes	Yes	16	$80 \to 17$	$58s$	7	$408s$
14	4	Always	Yes	Yes	16	$73 \to 10$	$50s$	7	$354s$
15	1	Always	Yes	Yes	64	$76 \to 13$	$63s$	8	$506s$
16	4	Always	Yes	Yes	64	$73 \to 10$	$47s$	8	$383s$

Table 4: Overall runtimes, storage costs, iteration counts and average time per iteration for the various solver options, with a $(64)^2 \times 4096$ space-time grid. A * indicates no delay in spatial coarsening.

(IIG), which uses as the initial guess for Newton's method at a specific time point and MGRIT level, the solution from the previous evaluation of Φ at that point and level. As XBraid converges, this value becomes an ever improving initial guess.

The main drawback of using the IIG is that an extra auxiliary vector must be stored at each point on every coarse grid[4]. The extra storage at coarse levels is required because the FAS solution, already stored there, is not equal to the result of the previous Φ evaluation.

This creates a tension between memory usage and computational efficiency. To limit memory usage, one can alternatively choose to store the solution at the fine grid C-points only. In this case, the IIG is available at all time steps except those completed during fine grid F-relaxation, where the previous time-step is used. This approach, referred to as IIG at C-points, is used by solvers 3, 4, 7 and 8.

[4]The solution from the previous Φ evaluation and the current FAS solution are equivalent on the fine-grid, so the IIG is already available at stored fine-grid points.

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	1.33270	1.20858	1.12873	0.63990	0.21818	0.46833
1	1.61597	1.14191	0.86014	0.36279	0.11507	0.10455
2	1.37209	0.85911	0.59375	0.23077	0.06196	0.04464
3	1.02336	0.58947	0.34524	0.14528	0.04021	0.02679
4	0.64319	0.34491	0.22029	0.09203	0.01724	0.02679
5	0.49765	0.32491	0.20167	0.04976	0.01596	0.01786

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.

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 (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 improved initial guess, while values greater than one indicate it increased. A large reduction in $c_\ell^{(j)}$ is seen across most temporal grids and iterations.

The exception to this is on the fine-grid, where costs increase during early iterations when using the IIG. This is not surprising. During the first few iterations the IIG is either inaccurate or unavailable.[5] A consequence of these increased costs is seen in Table 4, where solver 3, which uses the IIG everywhere except the fine grid F-Points, was actually slightly faster than when the IIG was used at all points.

Despite the increased costs on the fine grid, solvers 3 and 5, where the IIG 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 IIG is used in conjunction with spatial coarsening, are discussed in Section 5.4.

In conclusion, these results indicate that using the IIG as the initial guess is very beneficial. Using the IIG only at C-points is an excellent option, in this case reducing the runtime by 44%, with only a 32% increase in storage costs over the naive approach, when comparing solvers 0 and 3 in Table 4. Using the IIG as the initial guess at all points doubled the storage costs of the method, but only reduced the runtime by 43%.

Optionally, a user could opt to experimentally determine which initial guess to use on each level and iteration. Such a strategy would likely result in further cost savings, but be highly problem specific. Instead, we will pursue more general strategies that, when used in conjunction with the improved initial guess, will reduce the cost of those first three iterations in Section 6.

Either way, the improved initial guess is an effective, non-intrusive strategy that will provide a large reduction in runtime for most nonlinear parabolic implementations of MGRIT. This ability to effectively use reduced storage only at C-points and

[5] During the first iteration, at C-points, the user supplied initial guess is returned as the solution from a previous Φ evaluation. The user does not define an initial guess at F-points, so the previous time-step must be used.

the overall importance of improved initial guesses for nonlinear MGRIT are both important contributions of this work.

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 $R_x()$. At first glance spatial coarsening seems intrusive, however, MGRIT semi-coarsens in time and is agnostic to the spatial discretization, thus spatial coarsening is an *extra* option that can easily be implemented independently of the time-integration scheme.

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 equals 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.

The benefit of spatial coarsening is validated by the improved timings in Table 4. Here, solvers 0, 1 and 2 apply varying levels of spatial coarsening.

Beginning spatial coarsening on the first coarse grid (solver 2) leads to a degradation in MGRIT convergence. 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.

It is important to note that this degradation is not restricted to nonlinear problems. For example, if we replace the nonlinear co-efficient of the p-Laplacian with the exact, known solution, then the resulting linear problem still suffers from the same MGRIT degradation due to spatial coarsening. This degradation is still an area of active research, however recent analysis suggests that it occurs because the eigenvalues of the time-integration operators (Φ and Φ_Δ) are not bounded away from zero.

Our goal is to develop a spatial coarsening strategy that can be applied to all nonlinear problems. As such, we omit a detailed, problem specific convergence analysis and instead provide a simple strategy that allows spatial coarsening to be used with some nonlinear problems that exhibit this phenomenon. A similar strategy was used in [13] to make use of spatial coarsening inside a full space-time multigrid method.

The strategy, referred to as "delayed" spatial coarsening, is to limit spatial coarsening to the coarsest time grids. For example, solver 1 uses four spatial grids with spatial coarsening delayed until the fourth temporal level. Table 6 presents a cost analysis of this strategy, scaling 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 delayed 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.

Delayed spatial coarsening reduced both the number and cost of Newton iterations on the coarse grids, In fact, 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. Recognizing the specific importance of spatial coarsening when controlling the nonlinear time-step solver iterations on coarse time grids is an important contribution.

The benefit of these dramatic cost reductions is seen in Table 4. Solver 1, where spatial coarsening is delayed until the fourth temporal level, simultaneously minimized

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	1.00000	1.00390	1.00933	0.23705	0.05026	0.01139
1	1.00380	0.99670	0.99534	0.21163	0.04837	0.01036
2	0.99535	0.99656	1.00000	0.21612	0.04814	0.01004
3	1.00000	1.00000	0.99762	0.21459	0.04914	0.01004
4	1.00000	1.00000	1.00000	0.21437	0.04914	0.01004
5	1.00000	1.00000	1.00000	0.21437	0.04880	0.01004

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.

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	1.33270	1.20858	1.12873	0.14734	0.01058	0.00488
1	1.63118	1.15512	0.88112	0.07907	0.00651	0.00118
2	1.37674	0.87629	0.60096	0.04945	0.00387	0.00098
3	1.05140	0.59298	0.34762	0.03269	0.00251	0.00042
4	0.64319	0.35053	0.22768	0.02213	0.00137	0.00042
5	0.50235	0.32421	0.19905	0.01498	0.00107	0.00042

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, $\delta t = 1/16$.

the MGRIT convergence rate and the coarse grid computational costs, leading to a 37 % reduction in run-time when compared to solver 0. This is the strategy pursued in this paper and it has proven to be robust in our tests.

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 improvements reduced the overall runtime by 43% and 37%, respectively.

Table 7 gives a cost analysis of the MGRIT algorithm when these strategies are combined, using 4 levels of spatial coarsening and the IIG 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 IIG is used only at C-points, gives a 49% reduction in runtime with only a 5% increase in storage, when compared to solver 0.

Overall, delayed spatial coarsening, and spatial coarsening in general, is an effective strategy that can be applied to most, if not all, nonlinear parabolic imple-

mentations of MGRIT. Clearly, no delay in spatial coarsening is best, however, in cases where this causes a large degradation in the convergence rate, delayed spatial coarsening can still be used to balance the computational savings and convergence degradation. Mitigating this degradation in the convergence rate is an area of active research.

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 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 initial condition on the finest-grid is injected to the coarsest-grid and then serially propagated. Then, the solution is interpolated back to the finest-grid. Because we have no knowledge of the solution, the previous time step is used as the initial guess throughout this process. This strategy, similar to those found in full multigrid methods, is a non-intrusive modification of the basic MGRIT algorithm.

If some a priori knowledge of the solution is available to initialize the finest-grid for MGRIT, then this work should not be skipped. In all other cases, this is an effective, cheap, approach to determining an initial guess at the fine-grid C-points.

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 IIG 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. Skipping work can actually increase the cost of a Newton solve on some levels 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 "-1.0000" 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. 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

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	−1.0000	0.49677	0.77355	1.43736	4.58678	1.73333
1	0.50583	1.02000	1.15344	1.35294	1.63158	1.62000
2	0.89527	0.86667	0.92000	0.98765	1.03599	0.57143
3	0.81333	0.82840	0.88356	0.98148	1.11492	1.00000
4	0.94891	0.96296	0.96855	1.13643	1.41956	1.00000
5	0.94393	0.97727	0.96163	1.05847	1.00000	0.66667

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 10), 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.

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	−1.0000	0.66234	0.72650	0.82569	0.90090	0.88462
1	0.58525	0.66387	0.66514	0.68478	0.68548	0.61728
2	1.01887	1.01357	1.00870	1.01250	1.05263	1.25000
3	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
4	1.00000	1.00104	1.00108	1.00000	1.00000	1.00000
5	1.00000	1.00000	1.00000	0.99238	1.00000	1.00000

Table 9: Relative cost estimates, $c_\ell^{(j)}/c_{\ell,8}^{(j)}$, 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 8 (solver 10). Cost estimates are scaled entry-by-entry with $c_{\ell,8}^{(j)}$, the raw cost estimates used to generate Table 8.

three iterations on all levels. Our simple strategy here is to use $tol = 10^{-3}$ (as opposed to 10^{-7}) as the Newton tolerance during the first three iterations. Thereafter, tol returns to 10^{-7}. Some tuning of these parameters will be required when applying this strategy to another nonlinear problem.

Table 9 shows the cost analysis for this approach (solver 12) when it is added to the previous solver strategy from Section 6.1 (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 8. 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 40-45%. Table 4 (solvers 11 and 12) validates this by showing a decreased runtime and no degradation in overall MGRIT convergence.

REMARK 6.1. *A similar optimization that works well in the linear setting of [10] is to fix the number of iterations taken in the coarse-level implicit solves. However, this was not effective in the nonlinear case, apparently because the coarse Newton estimates were not accurate enough to maintain good convergence.*

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4
0	−1.0000	0.39376	0.61754	0.17001	0.04318
1	0.47909	0.77888	0.67599	0.07355	0.00591
2	1.25581	0.76289	0.54808	0.04792	0.00583
3	0.83645	0.47018	0.28810	0.02939	0.00404
4	0.59624	0.33228	0.21742	0.02503	0.00350
5	0.46667	0.31649	0.19045	0.01486	0.00188

Table 10: 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.

6.3. 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 small speedup of $6s$ when using spatial coarsening, but a slow down of $13s$ for the case of no spatial coarsening. This is because spatial coarsening makes the spatial 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 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.

6.4. Most effective improvements. Not surprisingly, the largest overall reduction in runtime is seen when spatial coarsening and the improved initial guess were used in conjunction with the three improvements outlined in sections 6.1-6.3.

Table 10 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. Level 0 of iteration 2 is the only entries for which $c_\ell^{(j)}$ increases. Comparing to Table 7, so that the overall effect of the improvements from sections 6.1-6.3 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 10.) Regarding runtime, Table 4 validates our strategy. Solver 14 is 3.2 times faster than the naive baseline, a direct consequence of the across the board cost savings seen in Table 10.

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

Table 11: Weak scaling study: MGRIT iteration counts.

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 47% improvement. Then, when solver 5 and solver 6 are compared (to see the impact of spatial coarsening), a further 8% improvement is seen. The other four strategies in concert combine for another 40% improvement (compare solver 6 with solver 14). The subsequent scaling studies, therefore, focus on solvers 0, 1, 4, 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.

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 $256^2 \times 65\,536$. The solvers considered are 0, 1, 4, 6 and 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 11 shows the number of MGRIT iterations required for each solver to reduce the MGRIT residual to within the 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, $128 * 16^3 = 524\,288$ 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 bene-

ficial. 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.[6]

The other plots are for the sequential ("Time-stepping") time-stepping code and solver ID's 0, 1, 4, 6 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's=4, 6) and the effects of all the other improvements (ID=14).

Figure 5a depicts the results for $m = 16$ with 16 processors in space, while Figure 5b shows the results for 32 processors in space and $m = 4$. For the sequential solver, the bottleneck occurs at around 256 spatial processors, although the best wall clock time of 2737 seconds is at 1024 spatial processors. Two different coarsening factors, m, and two different spatial processor counts are shown in order to indicate that these are now important user parameters affecting the speedup. The data points in each plot end when there are m points in time per processor on the finest-level, i.e., when using more processors in time provides no benefit. This is because FCF-relaxation is sequential over each interval of m points.

Observing the results for the $m = 16$ case, the crossover point at which MGRIT is beneficial is around 1024 processors, or about 64 processors in time. The maximum speedup over the sequential solver was a factor of 6.4 at 16384 processors. Note that delayed spatial coarsening (solver 1) does not show an improvement because the larger coarsening factor reduces the number of levels in the temporal hierarchy. 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 over the sequential solver was $21\times$.

In both cases solvers 4 and 6 scaled identically, this is a key result of this paper. Storing the solution at fine grid C-points reduces the storage costs of MGRIT without affecting the overall scaling of the algorithm. 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.

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 investigating each of the

[6] The 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.

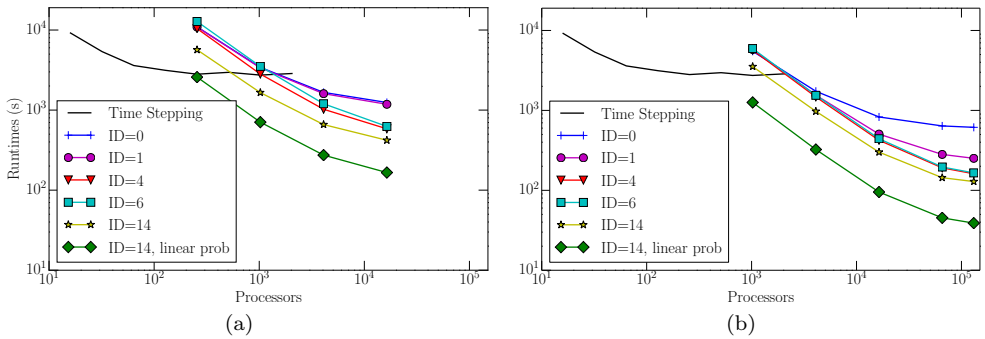

Fig. 5: Strong scaling study for a $(128)^2 \times 16385$ space-time grid, Left: 16 processors in space, $m = 16$, Right: 32 processors in space, $m = 4$.

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 develop a library of non-intrusive optimizations and modifications that make an efficient implementation of the MGRIT algorithm for nonlinear problems possible. As a measure of that efficiency, the MGRIT algorithm was compared against a similar implementation of a linear problem. When comparing to the linear problem in Figure 5, the lines for solver ID's "14" and "14, linear prob" are essentially parallel, which is a good result, indicating equivalent scaling. Yet, the line for "14, linear prob" is noticeably lower, indicating its faster runtime of about $3\times$. This is not surprising because for the linear problem, the Newton solver need only iterate once, while for the nonlinear problem, the Newton solver takes on average 2–3 iterations on the finest-level, meaning that the problem is itself 2–3 times more expensive. 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.

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.

If a good initial guess is not available for the first MGRIT iteration, the user should build an FMG-like initial guess by skipping work on the first down cycle and using the same initial guess as in the corresponding sequential solver. After that, the IIG should be used during all coarse-grid time steps. The accuracy of the IIG is poor during the first few MGRIT iterations, but gets better as MGRIT converges. Thus the best choice for an initial guess on the fine grid will vary with MGRIT iteration. This behavior is problem specific, hence, we suggest the user run a few small numerical tests to determine on which MGRIT iteration the fine-grid IIG should be introduced. If memory usage is a concern, the user can reduce storage costs by not storing the

finest grid F-points with little to no reduction in the overall runtime. In this scenario the initial guess from the corresponding sequential solver is used at each fine grid F-point. This is an important result as memory usage is a common user concern.

Spatial coarsening should also be used whenever possible. For the linear example in Figure 1, spatial coarsening was implemented on all levels effectively. For the nonlinear model problem, tests showed that a better strategy was to delay spatial coarsening until the fourth temporal grid, which dramatically reduced the cost of coarse Newton solves and limited degradation of the MGRIT convergence rate. This together with improved initial guesses gave the largest speedup. The other changes, when combined, also produced a significant speedup.

Weak scaling results showed that MGRIT is a scalable algorithm for the nonlinear parabolic problem considered, with iteration counts bounded independently of problem size. Strong scaling showed the benefit of MGRIT, with an up to 21 times speedup seen over the corresponding sequential time-stepping routine. Similar performance was attained when compared to the corresponding linear problem.

In summary, this paper has begun the process of developing a library of effective and numerically-backed optimizations and modifications that can be applied, with minimal tuning, to MGRIT for nonlinear parabolic problems.

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 solutions to boundary–value problems*, Math. Comp., 31 (1977), pp. 333–390.

[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] Martin J. Gander and Martin Neumüller, *Analysis of a new space-time parallel multigrid algorithm for parabolic problems*, SIAM Journal on Scientific Computing, 38 (2016),

pp. A2173–A2208.
[14] 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.
[15] S. 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.
[16] W. Hackbusch, *Parabolic multigrid methods*, in Computing methods in applied sciences and engineering, VI (Versailles, 1983), North-Holland, Amsterdam, 1984, pp. 189–197.
[17] G. Horton, *The time-parallel multigrid method*, Comm. Appl. Numer. Methods, 8 (1992), pp. 585–595.
[18] G. Horton and R. Knirsch, *A time-parallel multigrid-extrapolation method for parabolic partial differential equations*, Parallel Comput., 18 (1992), pp. 21–29.
[19] G. Horton and S. Vandewalle, *A space-time multigrid method for parabolic partial differential equations*, SIAM J. Sci. Comput., 16 (1995), pp. 848–864.
[20] 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.
[21] *HYPRE: High performance preconditioners.* `http://www.llnl.gov/CASC/hypre/`.
[22] H. B. Keller, *Numerical methods for two-point boundary-value problems*, Blaisdell Publishing Co. Ginn and Co., Waltham, Mass.-Toronto, Ont.-London, 1968.
[23] P. Lindqvist, *Notes on the p-laplace equation*, tech. report, Department of Mathematics, Ohio State University, 2006.
[24] 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.
[25] C. Lubich and A. Ostermann, *Multigrid dynamic iteration for parabolic equations*, BIT, 27 (1987), pp. 216–234.
[26] Yvon Maday, Olga Mula, and Mohamed-Kamel Riahi, *Toward a fully scalable balanced parareal method: Application to neutronics.* working paper or preprint, Aug 2015.
[27] Y. Maday and E.M. Rønquist, *Parallelization in time through tensor-product space-time solvers*, Comptes Rendus Mathématique. Académie des Sciences. Paris, 346 (2008), pp. 113–118.
[28] S. F. McCormick and J. W. Ruge, *Multigrid methods for variational problems*, SIAM J. Numer. Anal., 19 (1982), pp. 924–929.
[29] *MFEM: Modular finite element methods.* `www.mfem.org`.
[30] M. L. Minion, R. Speck, M. Bolten, M. Emmett, and D. Ruprecht, *Interweaving pfasst and parallel multigrid*, SIAM J. on Sci. Comput., 37 (2015), pp. S244–S263.
[31] 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.
[32] W. L. Miranker and W. Liniger, *Parallel methods for the numerical integration of ordinary differential equations*, Math. Comp., 21 (1967), pp. 303–320.
[33] J. Nievergelt, *Parallel methods for integrating ordinary differential equations*, Comm. ACM, 7 (1964), pp. 731–733.
[34] 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.
[35] J. B. Schroder, R. D. Falgout, and B. O'Neill, *Multigrid reduction in time (MGRIT): A flexible and non-intrusive method.* 4th Workshop on Parallel-in-Time Integration, Dresden, Germany, archived at Lawrence Livermore as LLNL-PRES-671059, May 2015.
[36] D. Sheen, I.H. Sloan, and V. 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.
[37] S. Vandewalle and G. Horton, *Fourier mode analysis of the multigrid waveform relaxation and time-parallel multigrid methods*, Computing, 54 (1995), pp. 317–330.
[38] 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.
[39] 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).
[40] 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.
[41] *XBraid: Parallel multigrid in time.* `http://llnl.gov/casc/xbraid`.