SIAM J. SCI. COMPUT.
Vol. 40, No. 3, pp. C283–C304

PARALLEL-IN-SPACE-TIME, ADAPTIVE FINITE ELEMENT FRAMEWORK FOR NONLINEAR PARABOLIC EQUATIONS*

ROBERT DYJA†, BASKAR GANAPATHYSUBRAMANIAN‡, AND KRISTOFFER G. VAN DER ZEE§

Abstract. We present an adaptive methodology for the solution of (linear and) nonlinear time dependent problems that is especially tailored for massively parallel computations. The basic concept is to solve for large blocks of space-time unknowns instead of marching sequentially in time. The methodology is a combination of a computationally efficient implementation of a parallel-in-space-time finite element solver coupled with a posteriori space-time error estimates and a parallel mesh generator. While we focus on spatial adaptivity in this work, the methodology enables simultaneous adaptivity in both space and time domains. We explore this basic concept in the context of a variety of time steppers including Θ-schemes and backward difference formulas. We specifically illustrate this framework with applications involving time dependent linear, quasi-linear, and semilinear diffusion equations. We focus on investigating how the coupled space-time refinement indicators for this class of problems affect spatial adaptivity. Finally, we show good scaling behavior up to 150,000 processors on the NCSA Blue Waters machine. This conceptually simple methodology enables scaling on next generation multicore machines by simultaneously solving for a large number of timesteps, and reducing computational overhead by locally refining spatial blocks that can track localized features. This methodology also opens up the possibility of efficiently incorporating adjoint equations for error estimators and inverse design problems, since blocks of space-time are simultaneously solved and stored in memory.

Key words. parabolic problems, parallel-in-time, finite element method, adaptive mesh refinement

AMS subject classifications. 65M22, 65Y05

DOI. 10.1137/16M108985X

1. Introduction.

We describe the methodology and application examples of space-time block adaptive solutions to parabolic partial differential equations. This approach is primarily motivated by the necessity of designing computational methodologies that can scale to leverage the availability of very large computing clusters (exascale and beyond). For evolution problems, the standard approach of decomposing the spatial domain is a powerful paradigm of parallelization. However, for a fixed spatial discretization, the efficiency of purely spatial domain decomposition degrades substantially beyond a threshold—usually tens of thousands of processors—which make this approach unsuitable on larger machines.[1] To overcome this barrier,

*Submitted to the journal's Software and High-Performance Computing section August 17, 2016; accepted for publication (in revised form) November 27, 2017; published electronically May 1, 2018.
http://www.siam.org/journals/sisc/40-3/M108985.html

Funding: The work of the first and second authors was supported by NSF 1435587, NSF XSEDE resources at TACC, as well as an exploratory account on NCSA Blue Waters (with thanks to Brett Bode). The work of the third author was supported by the Engineering and Physical Sciences Research Council (EPSRC) under grant EP/I036427/1.

†Czestochowa University of Technology, 42-201 Czestochowa, Poland (robert.dyja@icis.pcz.pl).

‡Corresponding author. Iowa State University, Ames, IA 50011 (baskarg@iastate.edu).

§School of Mathematical Sciences, University of Nottingham, University Park, Nottingham NG7 2RD, UK (KG.vanderZee@nottingham.ac.uk).

[1]There are a few approaches that scale reasonably well even on very large number of processors; see, for instance, Dendro [29]. However, there is still a case to be made for space-time approaches; for example, moderate spatial problems that have to be solved over a long time horizon.

a natural approach is to consider the time domain as an additional dimension and simultaneously solve for blocks of time, instead of the standard approach of sequential time stepping [12].

This concept of solving for blocks of space-time has resulted in several promising approaches to time parallel integration that have been developed over the past century, but which are gaining increasing attention due to that availability of appropriate computing resources. Broadly[2] one can consider three types of parallelization approaches to solving a space-time problem. The first type of methods explicitly parallelizes only over time, and leaves spatial parallelism (and spatial adaptivity) undefined [21, 12]. These methods may also be considered as shooting methods [15]. The second type of methods explicitly parallelizes over space, and leaves temporal parallelism (and temporal adaptivity) undefined. These methods include wave form relaxation methods that attempt to reconcile the solution at spatial boundaries between space-time blocks [15]. The third type of methods explicitly targets parallelism (and adaptivity) in space and time. The current work seeks to advance type three methods. In the more narrow context of finite element methods, early work on type three methods was considered by Hughes and coworkers [19, 18], Tezduyar et al. [30], and Potanza and Reddy [26], while variations on this theme have recently been explored by several groups [6, 8, 22, 23, 27, 34].

The concept of solving for blocks of time simultaneously has recently gained a lot of attention to enable effective usage of exascale computing resources.[3] In addition to this obvious advantage, solving for space-time blocks also allows natural incorporation of a posteriori error estimates for mesh adaptivity, and enables the solution of inverse problems involving adjoints [14, 13]. This has several additional tangible benefits in the context of computational overhead. For evolution problems—including wave equations, and problems involving moving interfaces like bubbles and shocks—that exhibit localized behavior in space and time, solving in blocks of space-time that are locally refined to match the local behavior provides substantial computational gain [8]. Similarly, the availability of error estimates across a block of time allows optimal choices of space and time adaptivity.

Motivated by these considerations, this paper presents a methodology for the solution of time dependent linear, quasi-linear, and semi-linear diffusion equations in three dimensions. We discuss the development of a parallel adaptive framework for the solution of large blocks of space-time. We detail the development of the block space-time framework for two classes of time steppers (θ-schemes, backward difference formula (BDF)). We subsequently define *a posteriori space-time error indicators* to identify spatial regions for mesh adaption. We show representative results using problems with analytical solutions and illustrate scaling behavior up to 150,000 processors. Finally, we demonstrate that for sufficiently large problems the block space-time approach enjoys a substantial speedup over sequential time stepping.

The outline of the rest of the paper is as follows: sections 2 and 3 detail the block space-time framework for linear and nonlinear evolution equations, respectively. Section 4 discusses the space-time error estimates for these classes of problems. In section 5, we discuss implementation details. Section 6 illustrates several numerical examples of the framework and shows scaling performance and analysis. We conclude in section 7.

[2]We thank the anonymous reviewer for suggesting this classification.

[3]See, for instance, the U.S. DOE's Exascale Mathematics Working Group [17].

2. Basic space-time formulation: linear and nonlinear versions.

2.1. Space-time framework for a linear problem. Given a bounded domain $\Omega \in R^3$, and a finite time domain $[0,T]$, consider the parabolic equation that solves, for $u\colon \Omega \times [0,T] \to \mathbb{R}$,

$$\begin{cases} \partial_t u(\mathbf{x},t) - \nabla \cdot \kappa \nabla u(\mathbf{x},t) = f(\mathbf{x},t) \quad \text{in } \Omega \times [0,T]\ , \\ u(\mathbf{x},0) = u_0\ , \end{cases} \tag{1}$$

where $f\colon \Omega \times [0,T] \to \mathbb{R}$ is a smooth source function, and $\kappa > 0$. We consider, without loss of generality, that Dirichlet boundary conditions are imposed on the boundary Γ, unless otherwise specified. Considering a tessellation, $\mathcal{T} \equiv \{\Omega^1, \dots, \Omega^e, \dots\}$, of the domain Ω into elements with average size h, the weak form of this equation is given as

$$\begin{cases} \text{find} \quad u^h(\cdot,t) \in \mathcal{U}^h : \\ \qquad (w^h, \partial_t u^h(\cdot,t)) + (\nabla w^h, \kappa \nabla u^h(\cdot,t)) = (w^h, f) \quad \forall w^h \in \mathcal{V}^h\ , \end{cases} \tag{2}$$

where $(.\,,.)$ is the L_2 inner product on Ω and

$$\begin{aligned} \mathcal{U}^h &:= \left\{ u^h | u^h \in H^1(\Omega), \quad u^h \in P(\Omega^e) \quad \forall e \right\}\ , \\ \mathcal{V}^h &:= \left\{ w^h | w^h \in H^1(\Omega), \quad w^h \in P(\Omega^e) \quad \forall e \right\} \end{aligned} \tag{3}$$

with $P(\Omega^e)$ being the space of the standard polynomial finite element shape functions on element Ω^e. To obtain a fully discretized form, we employ a time-stepping technique on the above semidiscrete equation. While any time-stepping method can be used, as an example, consider the Euler backward formula that is defined on a discretization $\{0, t_1, t_2, \dots, T\}$ of the time domain:

$$\left(w^h, \frac{u^h_{n+1} - u^h_n}{\Delta t} \right) + \left(\nabla w^h, \kappa \nabla u^h_{n+1} \right) = \left(w^h, f_{n+1} \right) \quad \text{for } n = 0, 1, \dots\ , \tag{4}$$

where the subscript denotes evaluation at that discrete time, and $\Delta t = t_{n+1} - t_n$ is the time step.

Following standard FEM practice, with the tessellation of the domain resulting in k nodal values that describe spatial variation of the field u, (4) can be expressed in terms of matrix-vector products as

$$\mathbf{M}\mathbf{u}_{n+1} + \Delta t \mathbf{K}\mathbf{u}_{n+1} = \mathbf{M}\mathbf{u}_n + \Delta t \mathbf{f}_{n+1} \quad \text{for } n = 0, 1, \dots\ , \tag{5}$$

where $\mathbf{M}$ and $\mathbf{K}$ are the global mass and stiffness matrices, respectively.[4] $\mathbf{u}_{n+1}$ and $\mathbf{u}_n$ are vectors containing the nodal values of the field u at time step $n+1$ and n, respectively. Equation (5) represents the system of equations solved to get the solution for time step $n+1$. The size of vector $\mathbf{u}$ is equal to the number of nodal unknowns, k. Similarly matrices $\mathbf{K}$, $\mathbf{M}$ are sparse matrices of size $k \times k$.

Consider a blockwise division of the total time domain. Each block, B_i, consists of multiple time steps. This is schematically represented in Figure 1. Instead of sequentially solving for each time step (as in (5)), consider solving for the field variable in a complete time block, B, consisting of N time steps simultaneously, i.e., solve for $\mathbf{u}_i,\ i = 1, \dots, N$, simultaneously.

[4] With some abuse of notation, the elements of matrix $\mathbf{M}$ are equal to $M_{ij} = (w^h_i, w^h_j)$ and matrix $\mathbf{K}$ are equal to $K_{ij} = (\nabla w^h_i, \kappa \nabla w^h_j)$.

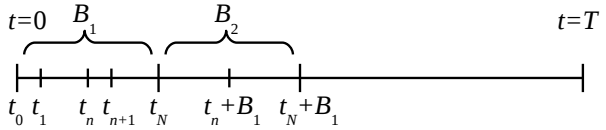


FIG. 1. *Indexing of time steps per block* (B). *Number of time steps per block is equal to N.*

This results in a block diagonal matrix of size $(N \times k)$-by-$(N \times k)$ given by

$$\begin{bmatrix} \mathbf{I} & & & & \\ -\mathbf{M} & \mathbf{M}+\Delta t\mathbf{K} & & & \\ & -\mathbf{M} & \mathbf{M}+\Delta t\mathbf{K} & & \\ & & \ddots & & \\ & & & -\mathbf{M} & \mathbf{M}+\Delta t\mathbf{K} \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_N \end{bmatrix} = \begin{bmatrix} \text{IC} \\ \Delta t\mathbf{f}_1 \\ \Delta t\mathbf{f}_2 \\ \vdots \\ \Delta t\mathbf{f}_N \end{bmatrix}, \tag{6}$$

where $\mathbf{I}$ is an identity matrix (of size k) and the IC are the imposed initial conditions. This system solves for N time steps at once with a total number of unknowns equal to $N \times k$.

Remark 1. Essentially, we convert the problem of sequentially solving for N time steps into a problem of solving for N unknowns simultaneously. By treating the unknown nodal values at different time steps as multiple degrees of freedom (DOFs) associated with each spatial node, we can leverage standard algorithmic approaches (assembly, memory usage) tailored for multiple DOFs problems. Note that by framing the temporal unknowns as multiple DOFs at each spatial point, the load distribution of the problem across processors is still based on a distribution of the spatial mesh across P processors. However, each processor now has N times more unknowns than the sequential case (5). This allows P to be much larger than for the sequential case while allowing for efficient parallel performance.[5] Many approaches in uncertainty quantification (polynomial chaos representation, spectral stochastic methods) leverage such an approach of representing field variation along additional dimensions (stochastic dimensions) as simply an additional DOF at each spatial location [25, 33, 10, 28].

2.2. Space-time framework for nonlinear problems. Extending the approach to certain nonlinear problems is straightforward. Consider the case where κ is a function of the dependent variable, u. We assume that $\kappa(u)$ satisfies appropriate smoothness and boundedness assumptions to ensure existence and uniqueness, $0 < \underline{\kappa} \le \kappa(u) \le \overline{\kappa} < \infty$. In this case the weak form for a block B is

$$\left(w^h, \frac{u^h_{n+1} - u^h_n}{\Delta t}\right) + \left(\nabla w^h, \kappa(u^h_{n+1})\nabla u^h_{n+1}\right) = \left(w^h, f_{n+1}\right) \quad \text{for } n = 0, 1, \ldots, N\,. \tag{7}$$

The solutions to such nonlinear equations are usually via (quasi-)Newton schemes. The methodology involves construction of the Jacobian and residual, which are used to compute updates. This is represented in matrix-vector terms as

$$\begin{gathered} \mathbf{J}_{\mathbf{u}^i_{n+1}} \delta\mathbf{u}^{i+1}_{n+1} = \mathbf{F}_{\mathbf{u}^i_{n+1}}, \quad \mathbf{u}^{i+1}_{n+1} = \mathbf{u}^i_{n+1} + \delta\mathbf{u}^{i+1}_{n+1} \\ \text{for } i = 1, \ldots, \text{until convergence} \;\; \text{and for } n = 0, 1, \ldots\,, \end{gathered} \tag{8}$$

[5] This is specifically illustrated in the results which show scalability for a range of different $N = 1, 10, 50, 100, 200, 500, 1000$.

where $\mathbf{J}_{\mathbf{u}_{n+1}^i}$ is the Jacobian (or linearized form), and $\mathbf{F}_{\mathbf{u}_{n+1}^i}$ is the residual of the above equation, both computed using $\mathbf{u}_{n+1}^i$. More specifically, for the nonlinear diffusion equations defined by (7), the residual, $\mathbf{F}_{\mathbf{u}_{n+1}^i}$, is given by

$$\mathbf{F}_{\mathbf{u}_{n+1}^i} = \frac{1}{\Delta t}\mathbf{M}\mathbf{u}_{n+1}^i - \frac{1}{\Delta t}\mathbf{M}\mathbf{u}_n - \mathbf{K}(\mathbf{u}_{n+1}^i)\mathbf{u}_{n+1}^i - \mathbf{f}_{n+1}\,, \tag{9}$$

where $\mathbf{K}(\mathbf{u}_{n+1}^i)$ denotes the solution dependent stiffness matrix. The Jacobian is given as

$$\mathbf{J}_{\mathbf{u}_{n+1}^i} = \frac{1}{\Delta t}\mathbf{M} + \left(\mathbf{K}(\mathbf{u}_{n+1}^i) + \frac{\mathrm{d}\mathbf{K}(\mathbf{u}_{n+1}^i)}{\mathrm{d}u}\right). \tag{10}$$

Instead of sequentially solving for each time step (as in (9)), consider solving for the field variable in a complete time block, B, consisting of N time steps simultaneously. That is,

$$\begin{bmatrix} \mathbf{J}_{\mathbf{u}_1^i} & & & \\ -\frac{1}{\Delta t}\mathbf{M} & \mathbf{J}_{\mathbf{u}_2^i} & & \\ & \ddots & & \\ & & -\frac{1}{\Delta t}\mathbf{M} & \mathbf{J}_{\mathbf{u}_N^i} \end{bmatrix} \begin{bmatrix} \delta\mathbf{u}_1^{i+1} \\ \delta\mathbf{u}_2^{i+1} \\ \vdots \\ \delta\mathbf{u}_N^{i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{\mathbf{u}_1^i} \\ \mathbf{F}_{\mathbf{u}_2^i} \\ \vdots \\ \mathbf{F}_{\mathbf{u}_N^i} \end{bmatrix} \tag{11}$$

and

$$\begin{bmatrix} \mathbf{u}_1^{i+1} \\ \mathbf{u}_2^{i+1} \\ \vdots \\ \mathbf{u}_N^{i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1^{i} \\ \mathbf{u}_2^{i} \\ \vdots \\ \mathbf{u}_N^{i} \end{bmatrix} + \begin{bmatrix} \delta\mathbf{u}_1^{i+1} \\ \delta\mathbf{u}_2^{i+1} \\ \vdots \\ \delta\mathbf{u}_N^{i+1} \end{bmatrix}.$$

Remark 2. One can alternatively ignore the off-diagonal entries of the block Jacobian to construct an approximate diagonal Jacobian. The propagation of time information is then limited to the residual on the right-hand side. We tried both approaches, with the latter approach taking more iterations to convergence, while providing substantial ease of implementation. Unless otherwise stated, all our results are based on the latter approach.

3. Space-time formulation: Higher-order time schemes. We next look at extending the space-time strategy to incorporate two families of higher-order time-steppers: Θ-scheme and BDF. We consider linear and nonlinear diffusion and, moreover, we also consider the treatment of the Allen–Cahn equation, which is a parabolic PDE with a lower-order nonlinearity, whose solution has evolving layers and for which adaptivity is particularly useful.

3.1. Θ-scheme: Linear equation. The semidiscrete form of the Θ-scheme—which is a generalization of the Euler backward scheme—is as follows:

$$\begin{aligned}(w, u_{n+1}) - (w, u_n) + \Delta t\left[(1-\Theta)\left(\nabla w, \kappa\nabla u_n\right) + \Theta\left(\nabla w, \kappa\nabla u_{n+1}\right)\right] \\ = \Delta t\left[(1-\Theta)\left(w, f_n\right) + \Theta\left(w, f_{n+1}\right)\right] \quad \text{for } n = 0, 1, \ldots\,.\end{aligned} \tag{12}$$

The fully discrete matrix-vector representation is given by

$$\begin{aligned}\mathbf{M}\mathbf{u}_{n+1} - \mathbf{M}\mathbf{u}_n + \Delta t\,(1-\Theta)\,\mathbf{K}\mathbf{u}_n + \Delta t\Theta\mathbf{K}\mathbf{u}_{n+1} \\ = \Delta t\,(1-\Theta)\,\mathbf{f}_n + \Delta t\Theta\mathbf{f}_{n+1} \quad \text{for } n = 0, 1, \ldots\,.\end{aligned} \tag{13}$$

Again, it is straightforward to group and simultaneously solve for N time steps together. The corresponding matrix form is expressed as

(14)
$$\begin{bmatrix} \mathbf{I} & & & & \\ -\mathbf{M}+\Delta t(1-\Theta)\mathbf{K} & \mathbf{M}+\Delta t\Theta\mathbf{K} & & & \\ & -\mathbf{M}+\Delta t(1-\Theta)\mathbf{K} & \mathbf{M}+\Delta t\Theta\mathbf{K} & & \\ & & \ddots & & \\ & & & -\mathbf{M}+\Delta t(1-\Theta)\mathbf{K} & \mathbf{M}+\Delta t\Theta\mathbf{K} \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_N \end{bmatrix} = \begin{bmatrix} \text{IC} \\ \Delta t\left[(1-\Theta)\mathbf{f}_0+\Theta\mathbf{f}_1\right] \\ \Delta t\left[(1-\Theta)\mathbf{f}_1+\Theta\mathbf{f}_2\right] \\ \vdots \\ \Delta t\left[(1-\Theta)\mathbf{f}_{N-1}+\Theta\mathbf{f}_N\right] \end{bmatrix}.$$

Again, the global space-time matrix (15) has a block structure.

3.2. Θ-scheme: Nonlinear diffusion with variable coefficient. The corresponding weak form for this case is given as

$$\left(w^h, \frac{u^h_{n+1}-u^h_n}{\Delta t}\right) + (1-\Theta)\left(\nabla w^h, \kappa(u^h_n)\nabla u^h_n\right) + \Theta\left(\nabla w^h, \kappa(u^h_{n+1})\nabla u^h_{n+1}\right) = (1-\Theta)\left(w^h, f_n\right) + \Theta\left(w^h, f_{n+1}\right) \quad \text{for } n=0,1,\dots. \tag{15}$$

The Jacobian is

$$\mathbf{J}_{\mathbf{u}^i_{n+1}} = \mathbf{M} + \Delta t\Theta\left(\mathbf{K}(\mathbf{u}^i_{n+1}) + \frac{\mathrm{d}\mathbf{K}(\mathbf{u}^i_{n+1})}{\mathrm{d}u}\right), \tag{16}$$

while the residual is given by

(17)
$$\mathbf{F}_{\mathbf{u}^i_{n+1}} = \mathbf{M}\mathbf{u}^i_{n+1} - \mathbf{M}\mathbf{u}_n + \Delta t\Theta\left(\mathbf{K}(\mathbf{u}^i_{n+1})\mathbf{u}^i_{n+1} - \mathbf{f}_{n+1}\right) + \Delta t\,(1-\Theta)\left(\mathbf{K}(\mathbf{u}_n)\mathbf{u}_n - \mathbf{f}_n\right).$$

Again, it is straightforward to group and simultaneously solve for N time steps together. Define diagonal blocks equal to

$$D_k = \mathbf{M} + \Delta t\Theta\left(\mathbf{K}(\mathbf{u}^i_k) + \frac{d\mathbf{K}(\mathbf{u}^i_k)}{du}\right) \tag{18}$$

with the the upper index denoting, as before, Newton–Raphson iteration. The corresponding (diagonal) block Jacobian (see Remark 2) is given by

$$\begin{bmatrix} D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_N \end{bmatrix}. \tag{19}$$

3.3. Θ-scheme: Allen–Cahn equation. The Allen–Cahn equation is a semilinear diffusion equation with a nonlinear reaction term:

$$\partial_t u(\mathbf{x},t) - \Delta u(\mathbf{x},t) + \epsilon^{-2} f(u) = 0\,, \tag{20}$$

where $f(u)$ is a nonlinear function of u, usually $f(u) = u\left(u^2 - 1\right)$. The initial condition is $u(\mathbf{x}, 0) = u_0$ along with zero flux conditions in the boundaries.

The corresponding semidiscrete form is given as

$$\left(w^h, \partial_t u^h\right) + \left(\nabla w^h, \nabla u^h\right) + \epsilon^{-2}\left(w^h, f(u^h)\right) = 0\,. \tag{21}$$

Using the Θ-scheme results in the fully discrete form

$$\begin{aligned}\left(w^h, u^h_{n+1}\right) &+ \Delta t\Theta\left[\left(\nabla w^h, \nabla u^h_{n+1}\right) + \epsilon^{-2}\left(w^h, f(u^h_{n+1})\right)\right] \\ &= \left(w^h, u^h_n\right) - \Delta t\left(1-\Theta\right)\left[\left(\nabla w^h, \nabla u^h_n\right) + \epsilon^{-2}\left(w^h, f(u^h_n)\right)\right]\,.\end{aligned} \tag{22}$$

Again, it is straightforward to group and simultaneously solve for N time steps together. The corresponding (diagonal) block Jacobian is given as

$$\begin{bmatrix} \mathbf{M} + \Delta t\Theta\left(\mathbf{K}(\mathbf{u}^i_1) + \epsilon^{-2}\frac{\mathrm{d}\mathbf{f}(\mathbf{u}^i_1)}{\mathrm{d}u}\right) & & & \\ & \mathbf{M} + \Delta t\Theta\left(\mathbf{K}(\mathbf{u}^i_2) + \epsilon^{-2}\frac{\mathrm{d}\mathbf{f}(\mathbf{u}^i_2)}{\mathrm{d}u}\right) & & \\ & & \ddots & \\ & & & \mathbf{M} + \Delta t\Theta\left(\mathbf{K}(\mathbf{u}^i_N) + \epsilon^{-2}\frac{\mathrm{d}\mathbf{f}(\mathbf{u}^i_N)}{\mathrm{d}u}\right) \end{bmatrix}. \tag{23}$$

Alternative schemes for the Allen–Cahn equation and other phase-field models are described in, e.g., [16].

3.4. BDF-based time steppers. BDF-based time steppers of order s utilize the solution at s previous time steps to construct the solution at the next time step. A general s order BDF scheme is given as

$$\sum_{k=0}^{s} \alpha_k u_{n+k} = \Delta t\beta g_{n+s}\,, \tag{24}$$

where the left-hand side is the BDF scheme representation of the time derivative, $\frac{\partial u}{\partial t}$, in terms of the solution u_i at time point i, and the right-hand side collects all other terms. Here, α and β are known BDF coefficients [7]. A first-order ($s = 1$) BDF scheme is identical to the Euler backward scheme described earlier. The simplest multistep scheme is for $s = 2$ and for the linear diffusion equation it is given as

$$\left(w^h, u^h_{n+2}\right) - \frac{4}{3}\left(w^h, u^h_{n+1}\right) + \frac{1}{3}\left(w^h, u^h_n\right) + \frac{2}{3}\Delta t\left(\nabla w^h, \kappa\nabla u^h_{n+2}\right) = \frac{2}{3}\Delta t\left(w^h, f_{n+2}\right)\,. \tag{25}$$

Again, it is straightforward to group and simultaneously solve for N time steps together.[6] The corresponding space-time block equations are as follows:

$$\begin{bmatrix} \mathbf{I} & & & & & \\ -\mathbf{M} & \mathbf{M}+\Delta t\mathbf{K} & & & & \\ \frac{1}{3}\mathbf{M} & -\frac{4}{3}\mathbf{M} & \mathbf{M}+\frac{2}{3}\Delta t\mathbf{K} & & & \\ & & & \ddots & & \\ & & \frac{1}{3}\mathbf{M} & -\frac{4}{3}\mathbf{M} & \mathbf{M}+\frac{2}{3}\Delta t\mathbf{K} \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_N \end{bmatrix} = \begin{bmatrix} \mathrm{IC} \\ \Delta t\mathbf{f}_1 \\ \frac{2}{3}\Delta t\mathbf{f}_2 \\ \vdots \\ \frac{2}{3}\Delta t\mathbf{f}_N \end{bmatrix}. \tag{26}$$

As expected, higher-order multistep methods produce block matrices that have a larger bandwidth. It is clear that nonlinear problems can be treated similarly.

[6] Note that the first time step is approximated using a backward Euler time stepper as the second-order BDF scheme requires knowledge of the solution at two previous time steps.

4. Adaptive meshing for the block space-time method: Residual-based error estimator. A central idea of this work is to develop a block space-time methodology that can be integrated with mesh adaptivity. This will enable targeted refinement of regions that exhibit variations in the corresponding block of time. Mesh adaptivity requires the definition of an indicator function that determines which regions of the space require refinement/coarsening. In this work, we build on prior work and utilize standard residual-based error indicators (see, e.g., [4, 32, 31]) and construct space-time analogues by averaging or taking the maximum across the time block. Alternative duality-based indicators for nonlinear parabolic problems are described in, e.g., [8, 9, 11].

Residual-based error indicators η_e are constructed for each spatial element, Ω_e, and consist of two terms—an interior residual, r_{int} and a jump residual, r_{jump} [32]:

$$\eta_e^2 = h_e^2 \|r_{int}^h\|^2_{L^2(\Omega_e)} + h_e \|r_{jump}^h\|^2_{L^2(\partial\Omega_e)}\,, \tag{27}$$

where h_e is the size of element, Ω_e. For example, for the linear and quasi-linear diffusion equation, the detailed derivation of the interior and jump residual is available in the work of Verfürth [32]. We refer the interested reader to that work and only show the key results here. Basically, the two terms are constructed (as the name suggests) from the definition of the residual:

$$\mathcal{R}^h(w) = (w, f) - \left(w, \partial_t u^h\right) - \left(\nabla w, \kappa(u^h)\nabla u^h\right)\,. \tag{28}$$

This residual is decomposed into elementwise terms as

$$R^h(w) = \sum_e \left(\int_{\Omega_e} w r^h_{\text{int}}\, \mathrm{d}\Omega + \int_{\Gamma_e} w r^h_{\text{jump}}\, \mathrm{d}\Gamma \right)\,, \tag{29}$$

where

$$r^h_{\text{int}} = f - \partial_t u^h + \nabla \cdot \kappa(u^h)\nabla u^h \tag{30}$$

and

$$r^h_{\text{jump}} = \begin{cases} 0 & \text{on} \quad \partial\Omega_e \cap \Gamma_h\,, \\ \kappa(u^h)\nabla u^h_+ \cdot \mathbf{n}_+ + \kappa(u^h)\nabla u^h_- \cdot \mathbf{n}_- & \text{on} \quad \partial\Omega_e \setminus \Gamma_h\,, \end{cases} \tag{31}$$

where Γ_h is that part of the boundary with Dirichlet conditions imposed.

The interior and jump residuals for the Allen–Cahn equations are similarly defined as

$$r^h_{\text{int}} = \epsilon^{-2} f(u^h) - \partial_t u^h + \Delta u^h \tag{32}$$

and

$$r^h_{\text{jump}} = \begin{cases} 0 & \text{on} \quad \partial\Omega_e \cap \Gamma_h\,, \\ \nabla u^h_+ \cdot \mathbf{n}_+ + \nabla u^h_- \cdot \mathbf{n}_- & \text{on} \quad \partial\Omega_e \setminus \Gamma_h\,. \end{cases} \tag{33}$$

Recall that we are using error indicators defined over a block of time. We extend the concept of error indicator defined at one time step to the notion of an error indicator defined over a block of time steps. There are several choices of error indicators with two of them being the average value of the error indicator across the block of time for an element e,

$$\eta_{e,avg} = \sum_{n=1}^{N} \frac{\eta_{e,n}}{N}\,, \tag{34}$$

and the maximum value in the block of time for an element e,

$$\eta_{e,max} = \max_{n=1,2,\ldots,N} \eta_{e,n} \,. \tag{35}$$

The former approach is advantageous for slow variations in the block and avoids frequent migrating of elements between processors, which can negatively affect scalability. The latter approach is advantageous when there are rapid changes across a few time steps. In this case the error estimator is not "diffused" by time steps where the solution is changing slowly.

Remark 3. In this work, we approximate the semidiscrete term, $\frac{\partial u}{\partial t}$, in terms of its finite difference representation (Θ or BDF scheme). An implicit assumption is that the time steps are small enough that this approximation is valid. Ideally, one would choose a consistent representation in both space and time, i.e., using a finite element representation for time variations [26, 19]. This provides several advantages in terms of mathematical elegance. We defer this development to a subsequent paper.

5. Implementation details. We utilize our in-house scalable, parallel finite element method (FEM) framework that is optimized for distributed memory computing. The FEM software library is implemented in C++ and uses object oriented software principles. Linear algebra, parallel matrix, and vector storage are all performed by the PETSc library [5]. PETSc modules (KSP, SNES) are used to solve (non)linear equations. Specifically, we use the hierarchical GMRES solvers as part of the PETSc software suite (KSP construct) for solving the linear system. These solvers have been shown to scale exceptionally well to hundreds of thousands of processors [24]. We use the block Jacobi preconditioner in conjunction with the GMRES routine. The FEM library is dynamically linked to the parallel hierarchical grid (PHG) library [2] which is a three dimensional (3D) parallel mesh refinement framework with inbuilt load balancing. PHG uses a bisection-type algorithm [35], specifically newest vertex bisection to refine/coarsen elements.[7] PHG operates on simplex elements and produces conforming meshes after refinement. We remind the reader that we are able to utilize parallel tools for 3D mesh adaptivity due to formulating our space-time problem as a spatial problem with N DOFs. Additional implementation details are provided in the appendix.

We perform scaling studies on two machines. Preliminary scaling was performed on the TACC Stampede [3]. Stampede consists of 6400 compute nodes each equipped with two Intel E5-2680 8-core processors. We also performed scaling on NCSA Blue Waters [1]. Blue Waters consists of 22,640 nodes, each consisting of two AMD 6276 Interlagos processors for a total of 362,240 computing cores.

6. Numerical examples. In this section we illustrate our adaptive parallel-in-space-time framework on a variety of linear and nonlinear diffusion equations. We show how the framework scales with increasing DOFs as well as increasing the size of the time-blocks, i.e., increasing N.

6.1. Problem A: Linear diffusion. We first consider the linear case. We set $\kappa = 1$ in (1), and consider a time horizon of $T = 1$. We use the method of manufactured solutions to construct the forcing term in (1) to ensure an analytical solution, u:

$$u(x, y, z, t) = \exp\left(-\alpha(x, y, z, t)\right) \,, \tag{36}$$

[7] In the newest vertex bisection, the edge that lies opposite to the newest node is divided.

where $\alpha(x, y, z, t)$ is equal to

$$\alpha(x, y, z, t) = \frac{(x - x_0(t))^2 + (y - y_0(t))^2 + (z - z_0(t))^2}{d^2} \tag{37}$$

and

$$\begin{aligned} x_0(t) &= a\cos(\omega t) + b\,, \\ y_0(t) &= a\sin(\omega t) + b\,, \\ z_0(t) &= b \end{aligned}$$

with $a = 0.2$, $b = 0.5$, $\omega = 2\pi$, $d = 0.1$. Thus, u is a rotating exponential hill centered at the midplane and rotating on a circle with a radius of 0.2 and with an angular speed of 2π (and hence a period of 1). This manufactured solution is constructed by the following forcing term

$$\begin{aligned} f(\mathbf{x}, t) = -\Bigg(&\frac{4((x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2)}{d^4} - \frac{6}{d^2} \\ &- 2a\omega\left(\frac{\cos(t\omega)(y - y_0) - \sin(t\omega)(x - x_0)}{d^2}\right)\Bigg)\exp\left(-\alpha(x, y, z, t)\right). \end{aligned} \tag{38}$$

The equation is solved in the unit cube $[0, 1] \times [0, 1] \times [0, 1]$. Every boundary face of the region has an essential boundary condition with prescribed value of u equal to the value computed from (36).

We first investigate the comparative scaling performance of the space-time approach with a sequential approach. We use a time step of 0.01 and solve for a block of 100 time steps. The space-time approach solves for this block of 100 time steps simultaneously, while the sequential approach steps through the time steps, solving for one time step at a time. Figure 2(top) shows scaling[8] on TACC Stampede for three sets of spatial discretization, 32^3, 64^3, and 128^3 trilinear elements. While this corresponds to $0.04M$, $0.3M$, and $2.1M$ DOFs for the sequential approach (that is then solved 100 times), the space-time approach corresponds to $3.6M, 27.5M$, and $214.7M$ DOFs (since they solve for all 100 time steps simultaneously). We can clearly see that the space-time approach exhibits better scalability over a larger range of processor counts, with the sequential approach tapering off at a much lower processor count. For instance, for the 128^3 discretization, the space-time approach shows good scaling behavior across the full range of processors investigated, while the sequential approach exhibits its best performance at 1024 processors.

We next look at the *total time to solve* for the full time horizon $[0, T]$. We plot this in Figure 2(bottom). As anticipated from the scaling studies, the space-time approach gives consistent performance improvements for a larger range of processor counts. This translates into consistent reduction of total time to solve with increasing processor count. While the sequential approach has lower total time to solve for smaller processor counts, beyond a certain processor count threshold the space-time approach outperforms the sequential approach.[9] This is about 2048 processors for the largest spatial discretization (128^3) considered. This bodes well for making a total

[8] The minimum number of processors used for any problem in this work was 16. Thus all scaling results are compared with the 16 processor result.

[9] One of the reviewers pointed out that Figures 2(bottom) and 5(bottom) that explore strong scaling are similar to those produced by the parareal [21] (Lions, Maday, and Turinici) and multigrid reduction in time [12] (Falgout et al.) methods. It is interesting to see qualitatively similar results using different approaches to parallel-in-time.

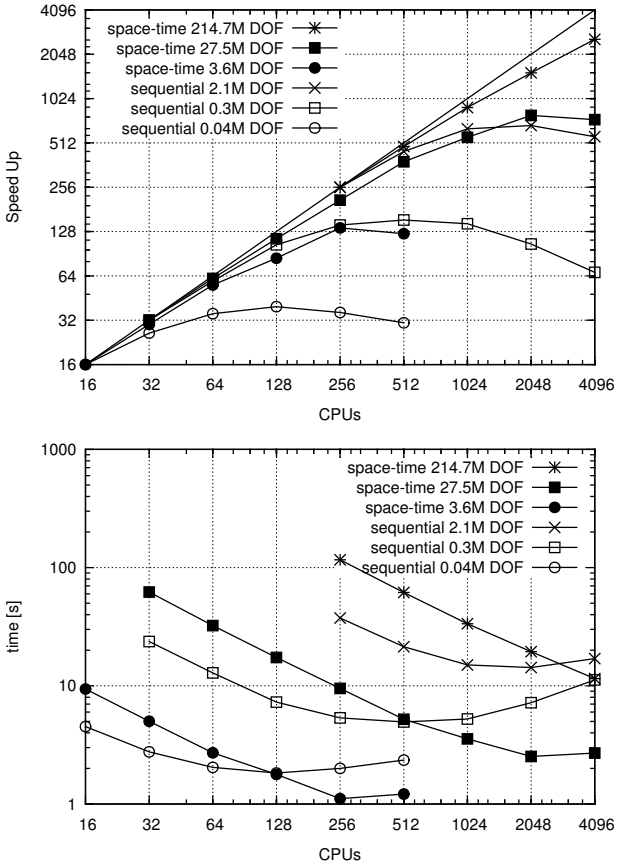


FIG. 2. *Scaling studies on TACC Stampede. Comparison between sequential time stepping and the space-time approach. The equation being solved is the linear diffusion equation with* $N = 100$ *time steps in a time block and discretized using the Euler backward scheme. The spatial discretizations used are* 32^3*,* 64^3*, and* 128^3*. Top plot shows speedup comparisons while bottom plot shows time to solution.*

time to solve argument when using large processor counts (especially, as will be shown later in this section, using $\sim$ 100,000 processors on the NCSA Blue Waters machine). Furthermore, the total time to solve metric for the space-time approach will become even more competitive for more complex problems involving complex geometries and multiple remeshing that will require I/O.

Remark 4. In some sense, by solving problems in space-time instead of marching forward in time, we are essentially making a computationally cheaper problem (of solving a spatial problem at one time step) more expensive. However, as we make the case,[10] there are several mitigating factors that warrant this approach. These factors include (a) hitting the limits of scaling when using purely sequential approaches, (b) the ability to store and solve the full time horizon with significant implications to solving adjoint problems, (c) the ability to only perform spatial adaptivity in blocks greatly simplifying storage, and, finally, (d) as a first step to simultaneous adaptivity in space and in time [19], which would prove very significant.

[10]Several other researchers have also made this case [12, 6, 20].

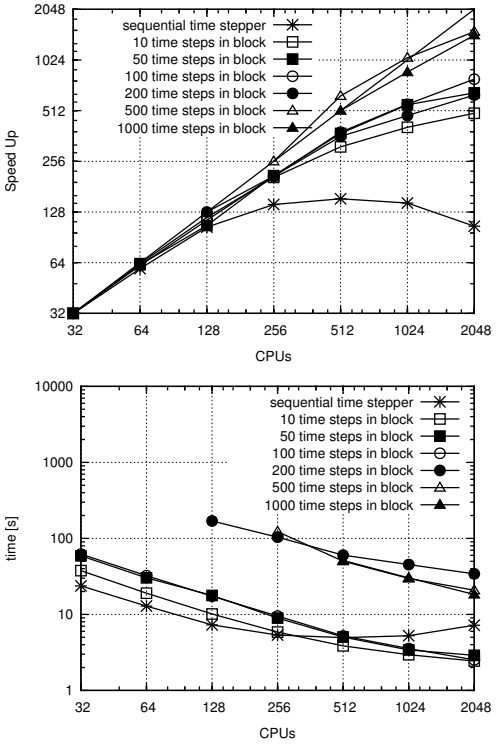


FIG. 3. *Speedup results on TACC Stampede for different sizes of N. The equation being solved is the linear diffusion equation with $N = 1, 10, 50, 100, 200, 500, 1000$ time steps in a time block and discretized using the Euler backward scheme. The spatial discretization is 64^3.*

We next look at how increasing the number of time steps, N, in a time block impacts performance of the space-time approach. Increasing N essentially increases the DOFs per spatial node in the space-time framework. We again consider the time interval $[0, T = 1]$. We investigate scaling behavior for a wide range of N ranging from $N = 1$ (the sequential case) to $N = 1000$. Figure 3(top) plots the scaling results on TACC Stampede using up to 2048 processors. Notice that as N increases the curves tend towards the ideal scaling line. This is consistent with earlier results in Figure 2 that show that increasing total DOFs results in better scaling performance. This is especially encouraging as it indicates that utilizing larger blocks of time (which translates to large total DOFs) results in lower total time to solve compared to a sequential approach. This is clearly seen in Figure 3(bottom) which plots the total time to solve over the full horizon $[0, T]$. The processor count at which the space-time approach beats the sequential approach shows an increasing trend with increasing N, with $N = 10$ taking less time than the sequential approach beyond just 256 processors, and the $N \geq 200$ potentially required more than 4096 processors to take less time than the sequential approach.

We next explore the number of GMRES iterations required to reach a relative tolerance ($-ksp\_rtol$) of 10^{-8} as a function of increasing N. Table 1 lists the number of iterations of the GMRES solver taken for various N's with a spatial discretization into 64^3 trilinear elements when run on 64 processors. Interestingly, the number of iterations saturates around 200 for $N \geq 50$. We emphasize that we have not tried to

TABLE 1
Number of GMRES iterations needed to solve one space-time block of given size. In all cases the block Jacobi preconditioner was used. The spatial discretization is 64^3.

N	1 (seq)	10	50	100	200	500	1000
Iterations	54	116	206	211	211	207	203

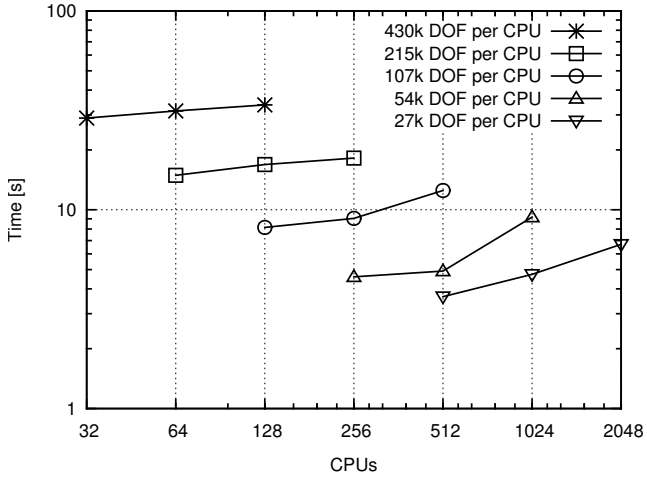


FIG. 4. *Weak scalability results on TACC Stampede for different numbers of DOFs per CPU. The same DOFs per CPU was achieved by using different numbers of time steps per block.*

optimize the linear algebra portion (solver + preconditioner) of the method, relying completely on PETSc modules. We expect that performance improvement may be achieved by carefully choosing the solver and preconditioners as well as including the full Jacobian (see Remark 2). An alternative question that we were unable to explore in this work was to understand how the number of GMRES iterations changes as the space-time system (and not just the time block, N) is refined.

While Figures 2 and 3 illustrate the *strong scalability* of the space-time approach, we also test the *weak scalability* of the space-time framework. We consider five different starting problem sizes (i.e., a starting spatial discretization and N), which are represented in terms of different DOFs per processor ($\frac{DOFs}{proc} = 27K, 54K, 107K, 215K, 430K$). For each starting problem size, we consider two doublings of the total DOFs while keeping the DOFs per processor fixed (i.e., double the total problem size, but also double the number of processors the problem is solved on). Figure 4 plots these weak scalability results. Ideal weak scalability is said to be exhibited when the total time to solution remains unchanged with increasing total problem size. Here, we see that for low DOFs per processor ($27K$, and $54K$ DOFs per processor) weak scaling is quite poor, but improves significantly for higher DOFs per processor scenarios.

Having established that the space-time approach is well suited for deploying on larger numbers of processors, we next look at scaling performance on the NCSA Blue Waters machine. The NCSA Blue Waters machine allowed us to test the space-time framework for a wide range of processor counts ranging from $2^4 = 32$ processors up to $2^{17} = 131{,}072$ processors. We evaluate strong scaling under two campaigns of simulations. The first campaign considers $N = 10$, and four spatial discretizations

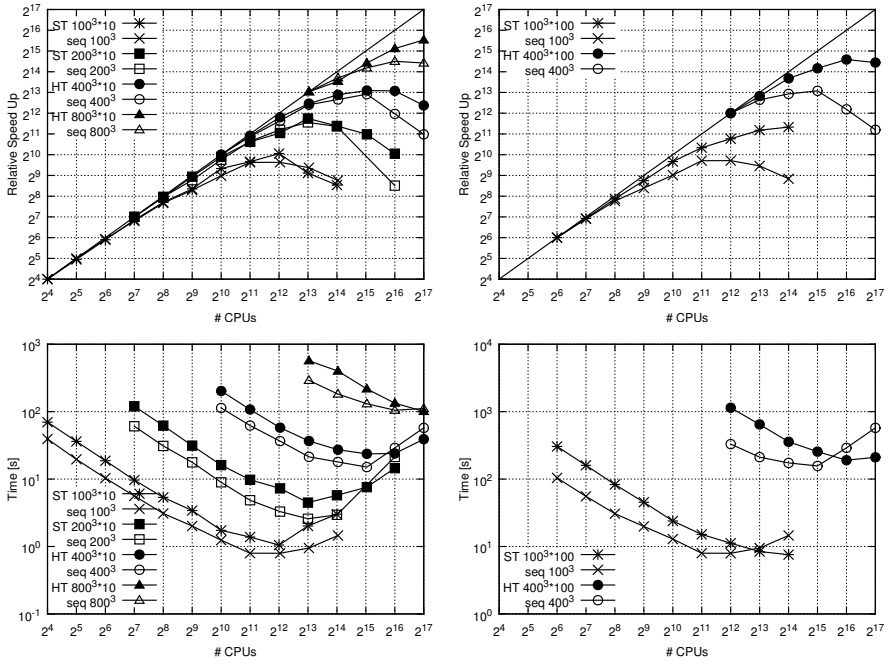


FIG. 5. *Speedup on NCSA Blue Waters for linear diffusion equation with time block* $N = 10$ *on the left and* $N = 100$ *on the right.*

of the domain into 100^3, 200^3, 400^3, and 800^3 trilinear elements. The results of the scaling studies are plotted in Figure 5(left top). The space-time approach produces consistent performance across a wide range of processors, with some degradation of performance at very high processor counts. Figure 5(left bottom) plots the total time to solve of the full time horizon $[0, T]$ for this set of simulations. This plot clearly shows that there is a processor count (especially for the larger problems 200^3, 400^3, and 800^3) beyond which the space-time approach outperforms the sequential approach. These trends in scalability as well as the time to solve are even more apparent in the second campaign that considers a larger time block of $N = 100$ (i.e., more DOFs per processor) and spatial discretizations of the domain into 100^3 and 400^3 trilinear elements. The scalability results are plotted in Figure 5(right top), and total time to solve results are plotted in Figure 5(right bottom).

We next look at the effect of adaptive meshing on the solution. Note that the analytical solution to the PDE is a rotating exponential hill centered at the midplane. We deploy the space-time approach with a residual-based error estimator considering a time horizon of $[0, T = 1]$ using a time step of 0.01. We consider a time block consisting of $N = 100$, which is one full rotation of the exponential hill in the midplane. Figure 6 shows part of the refined spatial mesh after 20 refinement iterations. We remind the reader that each iteration consists of solving the space-time problem, constructing the time-averaged elemental refinement indicators using (34), and then refining the 3D mesh according to the indicators. Given that this is a moving source problem, this refinement is clearly around the location where the peak of the hill passes (centered 0.2 away from the midpoint with a thickness of 0.1).

We finally compare spatial convergence rates for three implementations: (a) sequential time stepping with no spatial adaptivity, (b) space-time implementation with

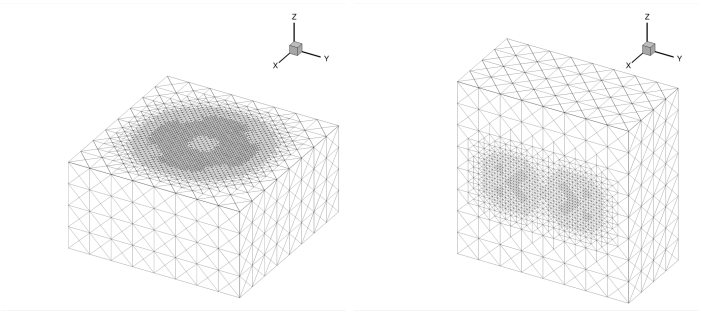


FIG. 6. *Refined mesh: view from top (left) and front (right).*

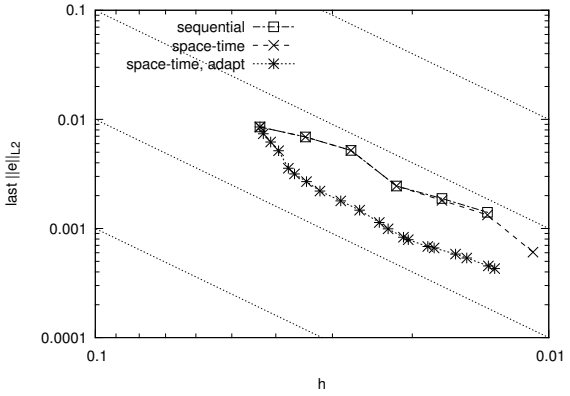


FIG. 7. *Spatial convergence for linear diffusion equation with time-step* 0.01. *Error is* $\|u-u_h\|_2$ *at the last time-step plotted as a function of the average element size* $\bar{h}$.

no spatial adaptivity, and (c) space-time implementation with spatial adaptivity. We plot convergence in Figure 7, where the error is $\|u-u_h\|_2$. The first two implementations (obviously) overlap, with the adaptive mesh implementation showing a reduced error. All three curves show a slope of 2, which is to be expected. Figure 8 shows time-step convergence, with expected slopes of 1 and 2 for backward Euler and BDFs, respectively.

6.2. Problem B: Nonlinear diffusion. For the nonlinear case, we set the coefficient $\kappa(u) = 1 + 10u^2$ and we choose an exact solution such that $|u| \leq 1$, thus bounding κ. As before, we choose our analytical solution to be given by (36). The forcing term, f, is consequently

$$
\begin{aligned}
(39)\quad & f(\mathbf{x},t) \\
&= \frac{80u}{d^4}\left((x-x_0)^2+(y-y_0)^2+(z-z_0)^2\right)\exp\left(-2\alpha(x,y,z,t)\right) \\
&+ \left(1+10u^2\right)\left(\frac{4\left((x-x_0)^2+(y-y_0)^2+(z-z_0)^2\right)}{d^4}-\frac{6}{d^2}\right)\exp\left(-\alpha(x,y,z,t)\right) \\
&- 2a\omega\frac{\cos(t\omega)\,(y-y_0)-\sin(t\omega)\,(x-x_0)}{d^2}\exp\left(-\alpha(x,y,z,t)\right).
\end{aligned}
$$

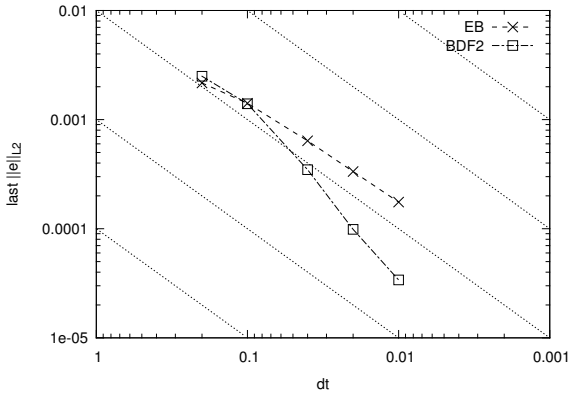


FIG. 8. *Time convergence for linear diffusion equation. Error is* $\|u - u_h\|_2$ *at the last time step plotted versus time step.*

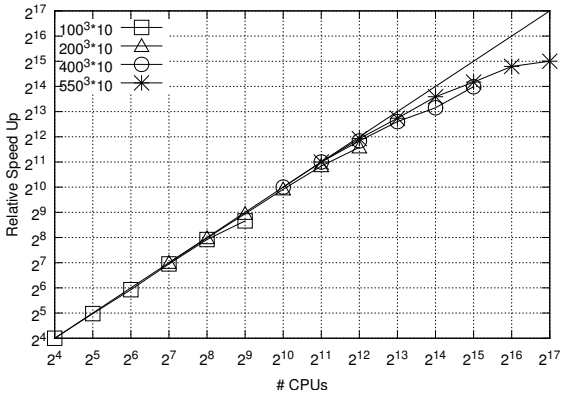
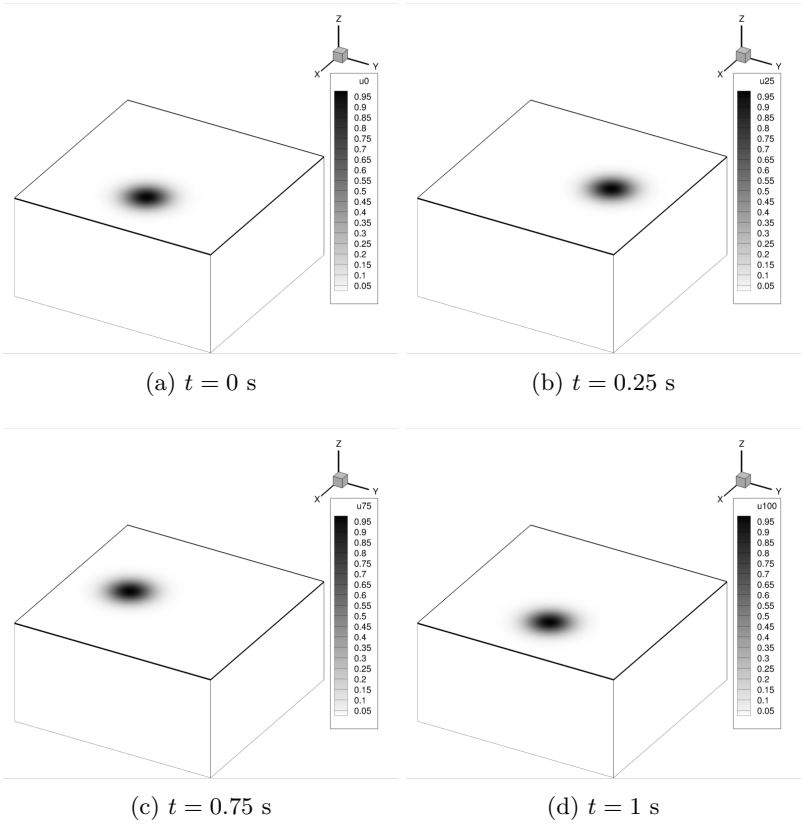


FIG. 9. *Speedup on NCSA Blue Waters for the nonlinear diffusion equation.*

The spatial and temporal domain over which the problem is solved remain unchanged from the linear case. Similarly to the linear case, we investigate scaling performance of the space-time approach. We however limit scaling studies to the NCSA Blue Waters machine, which provides a larger range of processor counts. We set $N = 10$, and consider four different spatial discretizations, 100^3, 200^3, 400^3, and 550^3. Figure 9 plots the scaling behavior of these problems across a wide range of processor counts.[11] As before, the space-time approach exhibits very promising and consistent scaling. While we do not show total time to solve analysis, our results indicate that comparison with the sequential time-stepping approach will show similar trends here as was shown for the case of the linear problem in Figure 5(left bottom).

We next look at the effect of adaptive meshing on the solution. We deploy the space-time approach with residual based error estimator for a time horizon of $[0, T = 1]$ using a time step of 0.01. We consider a time block consisting of $N = 100$, which is one full rotation of the exponential hill in the midplane. Figure 10 plots several time snapshots of the moving nonlinear source problem, with the solution accurately

[11] We use the PETSc SNES construct with a line search option. We set $-ksp\_rtol$ and $-snes\_rtol$ to 10^{-8} for all runs.



(a) $t = 0$ s (b) $t = 0.25$ s

(c) $t = 0.75$ s (d) $t = 1$ s

FIG. 10. *Snapshots of the solution at different time points on the midplane.*

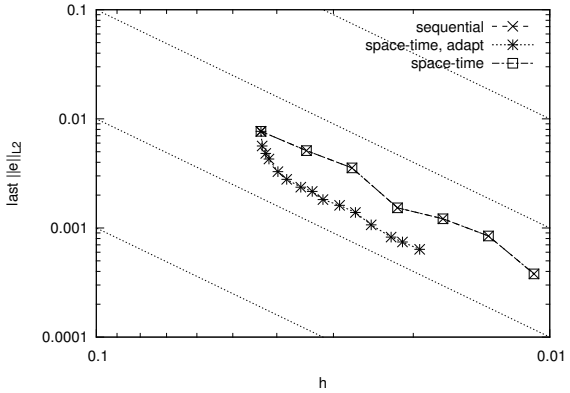


FIG. 11. *Spatial convergence for nonlinear heat equation with timestep* 0.01. *Error is* $\|u - u_h\|_2$ *at the last time step plotted as a function of the average element size* $\bar{h}$.

tracking the moving source. In Figure 11, we plot spatial convergence for three implementations: (a) sequential time stepping with no spatial adaptivity, (b) space-time implementation with no spatial adaptivity, and (c) space-time implementation with

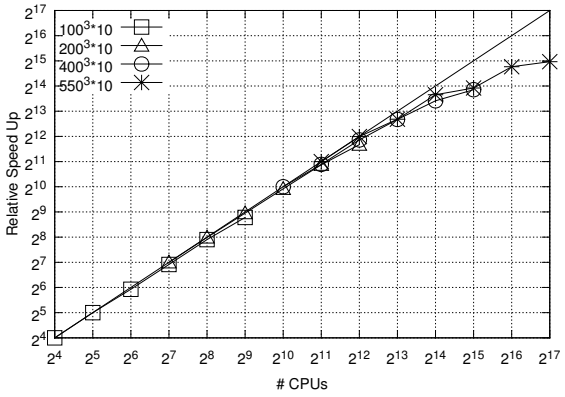


FIG. 12. *Speedup on NCSA Blue Waters for Allen–Cahn problem.*

spatial adaptivity (plotted with the average element size). Convergence rates follow along expected lines with a slope of 2.

6.3. Problem C: Allen–Cahn. In this final example, we solve the modified Allen–Cahn problem. The key physics which is described by this nonlinear equation essentially occurs in a highly localized region of the domain on a surface of codimension 1 which is evolving in time, thus representing a moving interface. Adaptive refinement and coarsening has been a very effective approach to accurately resolve this localized region, and a space-time approach with spatial adaptivity represents a very interesting approach to solving this problem. The equation is given as $\frac{\partial u}{\partial t} = -D\left(f(u) - C_n^2 \nabla^2 u\right)$, where

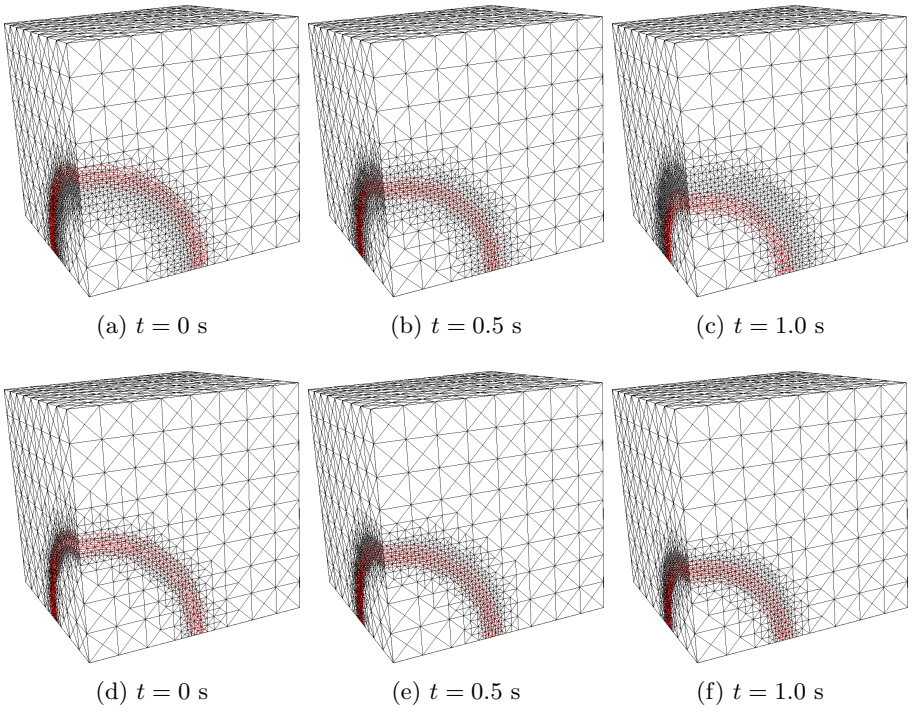
$$f(u) = 2Au\left(1 - 3u + 2u^2\right) - k \tag{40}$$

and $D = 1$, $C_n = 0.1$, $A = 16$, $k = 0.1$. The initial conditions are

$$u(x) = 0.5 + 0.5 \tanh\left(\frac{r - 0.5}{\sqrt{\frac{2}{A}} C_n}\right), \qquad r = \sqrt{x^2 + y^2 + z^2}\,, \tag{41}$$

with zero flux conditions on all boundaries. This represents an initial solid of radius, $r = 0.5$, that is melting. Using symmetry arguments, we consider a single octant of the space $[0:1] \times [0:1] \times [0:1]$. Similarly to the previous cases, we investigate scaling performance of the space-time approach. We again limit scaling studies to the NCSA Blue Waters machine, which provides a larger range of processor counts. We set $N = 10$ and use a time step of $\delta t = 0.02$, and consider four different spatial discretizations of 100^3, 200^3, 400^3, and 550^3 trilinear elements. Figure 12 plots the scaling behavior of these problems across a wide range of processor counts. As before, the space-time approach exhibits very promising and consistent scaling.

As a final result, we compare the adaptive mesh generated by the space-time framework with the adaptive mesh generated by the sequential approach. We consider the time domain $[0, 1]$ during which the initial sphere shrinks to about 70% of its original volume. We choose a time step of $\delta t = 0.02$ and consider $N = 50$. We remind the reader that each iteration consists of solving the space-time problem, constructing the time-averaged elemental refinement indicators using (34), and then refining the



(a) $t = 0$ s (b) $t = 0.5$ s (c) $t = 1.0$ s

(d) $t = 0$ s (e) $t = 0.5$ s (f) $t = 1.0$ s

FIG. 13. *Mesh after* 20 *refinement iterations with superimposed solution at times* 0 *s,* 0.5 *s,* 1 *s using the space-time adaptive approach (top). For comparison, view of adaptive mesh from the sequential solution (bottom).*

3D mesh according to the indicators. In contrast, the sequential problem consists of solving for each time step, constructing the refinement indicators, and then refining the 3D mesh according to the indicators at each time step.[12] Figure 13 illustrates the adaptive mesh refinement across the time block with the top row representing the space-time approach mesh and solution, and the bottom row representing the sequential approach mesh and solution. Notice that in the sequential approach the band of refined elements is moving with the moving front across the snapshots plotted. In contrast, the refinement of the space-time mesh is more "smeared" across the region that the moving front traverses in this time block. This results in an interplay between a slightly increased number of degrees of freedom that are being solved for, versus the computational overhead involved in repeated remeshing.

7. Conclusion. We present formulation, implementation details, and representative examples of a parallel-in-space-time-based adaptive methodology for the solution of (linear and) nonlinear time dependent problems. This is based on the simple concept of solving for large blocks of space-time unknowns instead of marching sequentially in time, which results in a block lower triangular system. This is analogous to recent approaches like parareal and MGRIT that solve for large space-time blocks. For the nonlinear problems, MGRIT (and parareal) use full approximation storage multigrid methods; in contrast, the approach taken here linearizes the entire space-time block and solves the linearization inside of a global space-time Newton solve.

[12] For computational efficiency, this is usually done every few time steps.

We emphasize that this approach is also nonintrusive, allowing the user to keep their existing space-time discretization. This serves as a first step towards a full space-time (intrusive) parallel adaptive formulation.

The approach is a combination of a computationally efficient implementation of a parallel-in-space-time finite element solver coupled with a posteriori space-time error estimates and a parallel mesh generator. We specifically address how a posteriori spatial error estimates can be extended to the space-time case. We illustrate how this implementation is especially tailored for massively parallel computations. We show good scaling behavior up to 150,000 processors on the Blue Waters machine. We make the case that this strategy is especially useful for large time blocks on computing clusters with very large numbers of processors.

This work opens up several avenues of future work. An open question is identifying which of (or under what conditions one of) the several approaches (including the current work) for solving space-time problems are optimal, especially for nonlinear problems. Another open question is to investigate how the space-time refinement compares with an optimal sequential refinement. A further question is to identify the trade-off between using the approximate Jacobian versus the full Jacobian in terms of memory versus performance (Remark 2). A fourth avenue of research is to efficiently incorporate adjoint equations for error estimators and inverse design problems, since blocks of space-time are simultaneously solved and stored in memory. A fifth avenue is to extend the space-time framework to utilize finite element basis functions in time (which enables formal derivation of space-time a posteriori error estimates), and subsequently implementing four dimensional finite elements to enable simultaneous space and time adaptivity.

Appendix A. Implementation details. While our existing implementation was reasonably optimized, several standard software engineering principles had to be implemented to ensure efficient execution of the block space-time problems.[13] We made software engineering decisions to ensure that the space-time implementation is compatible with our existing sequential FEM frameworks (see Remark 1 in main text).

Memory interlacing and matrix bandwidth: We rearrange the vector of unknowns $\mathbf{u}$ to enumerate time points before looping over space, i.e., $\mathbf{u} = \{u_1^1, u_2^1, \ldots, u_N^1, u_1^2, u_2^2, \ldots, u_N^2, \ldots, u_N^k\}$, where subscript refers to time and superscript denotes space. This allows for more efficient assembly, because all data (coefficients in system of equation) for a specific element share memory locality, thus preventing cache misses. Moreover, this approach is compatible with existing frameworks for FEM, because problems with multiple DOFs are supported in existing FEM frameworks, so we can use standard procedures for system assembling or imposing boundary conditions. This has the additional advantage of reducing the matrix bandwidth. Table 2 enumerates the bandwidth for space-time formulation (two dimensional mesh with 100×100 quad elements, linear basis function) using a Euler backward formulation. As expected, the bandwidth linearly increases with an increasing number of time steps.

Matrix access and storage: Careful identification of the nonzero pattern results in much larger savings. We utilize the "no-zero" approach where we avoid

[13] Some of the simpler changes that had substantial impact for the space-time formulation *but had minimal impact on the existing, standard iterative formulation* are (a) moving calculations out of loops whenever possible; (b) executing calculations and storing results in array before loops; (c) conversion to 64-bit-based integer variables for storing matrix indices, which allows going beyond 4 billion unknowns; (d) using local values instead of values obtained through pointer or reference.

TABLE 2
The bandwidth size for space-time formulation.

	Number of time steps				
	Sequential	5	10	25	50
Row size	10201	51005	102010	255025	510050
Bandwidth	205	1023	2043	5103	10203

inserting zero elements into the global matrix. The memory requirement for this case is also substantially minimized (close to two orders of magnitude reduction in memory requirements).

Acknowledgment. The authors thank the two anonymous reviewers for excellent suggestions that helped improve this paper.

REFERENCES

[1] *Blue Waters: System Summary*, https://bluewaters.ncsa.illinois.edu/hardware-summary (2016).
[2] *PHG: Parallel Hierarchical Grid*, http://lsec.cc.ac.cn/phg/ (2016).
[3] *TACC Texas Advanced Computing Center: Stampede*, https://www.tacc.utexas.edu/resources/hpc/stampede-technical (2016).
[4] M. AINSWORTH AND J. T. ODEN, *A Posteriori Error Estimation in Finite Element Analysis*, Wiley, New York, 2000.
[5] S. BALAY, W. D. GROPP, L. C. MCINNES, AND B. F. SMITH, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools for Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser, Boston, 1997, pp. 163–202, https://doi.org/10.1007/978-1-4612-1986-6_8.
[6] M. BEHR, *Simplex space-time meshes in finite element simulations*, Internat. J. Numer. Methods Fluids, 57 (2008), pp. 1421–1434, https://doi.org/10.1002/fld.1796.
[7] J. C. BUTCHER, *Numerical Methods for Ordinary Differential Equations*, Wiley, West Sussex, England, 2008.
[8] V. CAREY, D. ESTEP, A. JOHANSSON, M. LARSON, AND S. TAVENER, *Blockwise adaptivity for time dependent problems based on coarse scale adjoint solutions*, SIAM J. Sci. Comput., 32 (2010), pp. 2121–2145, https://doi.org/10.1137/090753826.
[9] G. ŞIMŞEK, X. WU, K. VAN DER ZEE, AND E. VAN BRUMMELEN, *Duality-based two-level error estimation for time-dependent PDEs: Application to linear and nonlinear parabolic equations*, Comput. Methods Appl. Mech. Engrg., 288 (2015), pp. 83–109, https://doi.org/10.1016/j.cma.2014.11.019.
[10] B. J. DEBUSSCHERE, H. N. NAJM, P. P. PÉBAY, O. M. KNIO, R. G. GHANEM, AND O. P. LE MAITRE, *Numerical challenges in the use of polynomial chaos representations for stochastic processes*, SIAM J. Sci. Comput., 26 (2004), pp. 698–719, https://doi.org/10.1137/S1064827503427741.
[11] K. ERIKSSON, C. JOHNSON, AND A. LOGG, *Adaptive computational methods for parabolic problems*, in Encyclopedia of Computational Mechanics, Fundamentals, E. Stein, R. de Borst, and T. Hughes, eds., Wiley, Chichester, England, 2004, pp. 675–702.
[12] R. D. FALGOUT, S. FRIEDHOFF, TZ V. KOLEV, S. P. MACLACHLAN, AND J. B. SCHRODER, *Parallel time integration with multigrid*, SIAM J. Sci. Comput., 36 (2014), pp. C635–C661, https://doi.org/10.1137/130944230.
[13] B. GANAPATHYSUBRAMANIAN AND N. ZABARAS, *Control of solidification of non-conducting materials using tailored magnetic fields*, J. Cryst. Growth, 276 (2005), pp. 299–316, https://doi.org/10.1016/j.jcrysgro.2004.11.336.
[14] B. GANAPATHYSUBRAMANIAN AND N. ZABARAS, *On the control of solidification using magnetic fields and magnetic field gradients*, Int. J. Heat Mass Transf., 48 (2005), pp. 4174–4189, https://doi.org/10.1016/j.ijheatmasstransfer.2005.04.027.
[15] M. J. GANDER, 50 *years of time parallel time integration*, in Multiple Shooting and Time Domain Decomposition Methods: MuS-TDD, Heidelberg, 2013, T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, eds., Springer, Cham, Switzerland, 2015, pp. 69–113, https://doi.org/10.1007/978-3-319-23321-5_3.

[16] H. Gomez and K. G. van der Zee, *Computational Phase-Field Modeling*, 2nd ed., Encyclopedia Comput. Mech., Wiley, 2018.
[17] J. Hittinger, S. Leyffer, and J. Dongarra, *Models and algorithms for exascale computing pose challenges for applied mathematicians*, SIAM News, Dec. 2013.
[18] T. J. Hughes and J. R. Stewart, *A space-time formulation for multiscale phenomena*, J. Comput. Appl. Math., 74 (1996), pp. 217–229, https://doi.org/10.1016/0377-0427(96)00025-8.
[19] T. J. R. Hughes and G. M. Hulbert, *Space-time finite element methods for elastodynamics: Formulations and error estimates*, Comput. Methods Appl. Mech. Engrg., 66 (1988), pp. 339–363, https://doi.org/10.1016/0045-7825(88)90006-0.
[20] U. Langer, S. E. Moore, and M. Neumüller, *Space–time isogeometric analysis of parabolic evolution problems*, Comput. Methods Appl. Mech. Engrg., 306 (2016), pp. 342–363, https://doi.org/10.1016/j.cma.2016.03.042.
[21] J. Lions, Y. Maday, and G. Turinici, *A"parareal"in time discretization of PDE's*, C. R. Acad. Sci. Ser. I Math., 332 (2001), pp. 661–668.
[22] R. B. Lowrie, P. L. Roe, and B. van Leer, *Space-time methods for hyperbolic conservation laws*, in Barriers and Challenges in Computational Fluid Dynamics, V. Venkatakrishnan, M. D. Salas, and S. R. Chakravarthy, eds., Springer, Dordrecht, Netherlands, 1998, pp. 79–98, https://doi.org/10.1007/978-94-011-5169-6_5.
[23] K. Mani and D. Mavriplis, *Efficient solutions of the Euler equations in a time-adaptive space-time framework*, in 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics, Reston, VA, 2011, https://doi.org/10.2514/6.2011-774.
[24] L. C. McInnes, B. Smith, H. Zhang, and R. T. Mills, *Hierarchical Krylov and nested Krylov methods for extreme-scale computing*, Parallel Comput., 40 (2014), pp. 17–31, https://doi.org/10.1016/j.parco.2013.10.001.
[25] A. Narayan and D. Xiu, *Stochastic collocation methods on unstructured grids in high dimensions via interpolation*, SIAM J. Sci. Comput., 34 (2012), pp. A1729–A1752, https://doi.org/10.1137/110854059
[26] J. P. Pontaza and J. N. Reddy, *Space-time coupled spectral/hp least-squares finite element formulation for the incompressible Navier-Stokes equations*, J. Comput. Phys., 197 (2004), pp. 418–459, https://doi.org/10.1016/j.jcp.2003.11.030.
[27] T. C. S. Rendall, C. B. Allen, and E. D. C. Power, *Conservative unsteady aerodynamic simulation of arbitrary boundary motion using structured and unstructured meshes in time*, Internat. J. Numer. Methods Fluids, 70 (2012), pp. 1518–1542, https://doi.org/10.1002/fld.2756.
[28] C. Soize and R. Ghanem, *Physical systems with random uncertainties: Chaos representations with arbitrary probability measure*, SIAM J. Sci. Comput., 26 (2004), pp. 395–410, https://doi.org/10.1137/S1064827503424505.
[29] H. Sundar, R. S. Sampath, and G. Biros, *Bottom-up construction and* 2 : 1 *balance refinement of linear octrees in parallel*, SIAM J. Sci. Comput., 30 (2008), pp. 2675–2708, https://doi.org/10.1137/070681727.
[30] T. E. Tezduyar, S. Sathe, R. Keedy, and K. Stein, *Space-time finite element techniques for computation of fluidstructure interactions*, Comput. Methods Appl. Mech. Engrg., 195 (2006), pp. 2002–2027, https://doi.org/10.1016/j.cma.2004.09.014.
[31] R. Verfürth, *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*, Wiley-Teubner, New York, 1996.
[32] R. Verfürth, *A Posteriori Error Estimation Techniques for Finite Element Methods*, Oxford University Press, Oxford, 2008.
[33] X. Wan and G. E. Karniadakis, *Multi-element generalized polynomial chaos for arbitrary probability measures*, SIAM J. Sci. Comput., 28 (2006), pp. 901–928, https://doi.org/10.1137/050627630.
[34] L. Wang and P.-O. Persson, *A high-order discontinuous Galerkin method with unstructured spacetime meshes for two-dimensional compressible flows on domains with large deformations*, Comput Fluids, 118 (2015), pp. 53–68, https://doi.org/10.1016/j.compfluid.2015.05.026.
[35] L.-B. Zhang, *A parallel algorithm for adaptive local refinement of tetrahedral meshes using bisection*, Numer. Math. Theory Methods Appl., 2 (2009), pp. 65–89.