BLOCKWISE ADAPTIVITY FOR TIME DEPENDENT PROBLEMS
BASED ON COARSE SCALE ADJOINT SOLUTIONS

V. CAREY*, D. ESTEP†, A. JOHANSSON‡, M. LARSON§, AND S. TAVENER¶

Abstract.
We describe and test an adaptive algorithm for evolution problems that employs a sequence of “blocks” consisting of fixed, though non-uniform, space meshes. This approach reduces computational costs in a high performance environment with respect to load balancing and with respect to solution of adjoint problems used to estimate discretization error and the effects of mesh changes. A major issue with a block-adaptive approach is determining block discretizations from coarse scale solution information that achieve the desired accuracy. We describe several strategies to achieve this goal using adjoint-based a posteriori error estimates and we demonstrate the behavior of the proposed algorithms as well as several technical issues in a set of examples.

Key words. a posteriori error analysis, adaptive error control, adaptive mesh refinement, adjoint problem, discontinuous Galerkin method, duality, generalized Green’s function, goal oriented error estimates, residual, variational analysis

AMS subject classifications. 65N15, 65N30, 65N50

1. Introduction. We describe and test an adaptive algorithm for evolution problems that we call “blockwise adaptivity”. This approach employs a sequence of “blocks” consisting of fixed, though non-uniform, space meshes, which is motivated by considerations of efficiency and accuracy. In considering efficiency, we take into account both the goals of achieving desired accuracy using discretizations with relatively few degrees of freedom and minimizing computational costs in a high performance environment such as load balancing and the cost of error estimation. A block adaptive strategy reduces the number of mesh changes that must be treated. This reduces the amount of computational time spent on load balancing and makes the problem of quantifying the effects of mesh changes on accuracy computationally feasible. A block adaptive strategy also provides a natural coarse scale discretization on which to solve the adjoint problem used to compute global a posteriori error estimates, thereby avoiding high performance issues such as storage of a fine scale forward solution to form the adjoint problem and solution of the adjoint problem on fine scale discretization. However, a major issue with a block-adaptive approach is determining block discretizations from coarse scale solution information that achieve...
the desired accuracy and efficiency. We describe several strategies to achieve this goal using adjoint-based \textit{a posteriori} error estimates.

To focus the discussion, we consider a reaction-diffusion equation for the solution $u$ on an interval $[0,T]$,

$$\begin{cases}
u - \nabla \cdot (\epsilon(x,t) \nabla u) = f(u,x,t), & (x,t) \in \Omega \times (0,T], \\ u(x,t) = 0, & (x,t) \in \partial \Omega \times (0,T], \\ u(x,0) = u_0(x), & x \in \Omega,
\end{cases} \tag{1.1}$$

where $\Omega$ is a convex polygonal domain in $\mathbb{R}^d$ with boundary $\partial \Omega$, $\dot{u}$ denotes the partial derivative of $u$ with respect to time, and there is a constant $\epsilon > 0$ such that $\epsilon(x,t) \geq \epsilon$, $x \in \Omega$, $t > 0$.

We also assume that $\epsilon$ and $f$ have smooth second derivatives. The algorithms in this paper generalize to problems with different boundary conditions, convection, nonlinear diffusion coefficients, as well as systems, see [60, 58].

In terms of adaptive mesh refinement, the interesting situation is a solution of (1.1) that exhibits “regionalized” behavior in space and time. Considerations of efficiency suggest that time steps and space meshes should be locally refined to match the regional behavior, see the plot on the left in Fig. 1.1. Adaptive mesh refinement can be described as a constrained optimization problem, e.g. determine a discretization using the fewest degrees of freedom that yields a solution satisfying a given error criterion. In general, it is impossible to determine a closed-form solution of this optimization problem. An adaptive algorithm is an iterative procedure for determining a nearly optimal solution.

We present a generic adaptive algorithm in Algorithm 1.1. An adaptive computation is generally started with an initial coarse mesh. After that, the adaptive algorithm is applied “real-time” as the integration proceeds so as to generate a new
space mesh for each new time step that is based on the mesh for the current step. In practice, the remeshing may be applied on intervals of a small number of steps.

**Algorithm 1.1** Generic Adaptive Algorithm for an Evolution Problem

Choose an initial coarse mesh and time step

while the final time has not been reached do

  Compute a numerical solution using the current time step and space mesh

  Estimate the error of the computed solution

  while the error estimate is too large do

    Space: Estimate the element contributions to the error
    Mark elements for refinement and coarsening
    Adapt the discretization

    Time: Estimate the time error contribution
    Adapt the time step

  end while

  Compute a numerical solution using the new time step and space mesh

  Estimate the error of the computed solution

end while

Increment time by the accepted time step

end while

While adaptive mesh refinement is appealing on an intuitive level, there are serious issues facing its use for evolution problems including the following.

1. **Accuracy** Each spatial mesh change requires a projection of the numerical solution onto the new mesh, and this can affect accuracy. In fact, this can destroy convergence altogether, see [45].

2. **Load Balancing** On a high performance computer, significant mesh changes generally require a redistribution of unknowns among the processors. This is often computationally intensive.

3. **Coarsening** Un-refinement or coarsening of a mesh involves loss of information about a numerical solution that cannot be recovered. Currently, there is no theory for coarsening that guarantees that there is no loss of accuracy.

4. **Global Error Estimation** Truly efficient adaptive mesh refinement requires accurate error estimates of the true, global error, but accurate error estimates are affected strongly by cancellation of error, which makes choosing adapted meshes problematic.

Using a fixed spatial mesh eliminates the first three issues. But, the scale required of the mesh is determined by the finest scale required in any region where discretization impacts global accuracy, see Fig. 1.1. This can result in increased computational time because of solver costs and memory limits may make it impossible to use the necessary uniform mesh.

In this paper, we propose a “blockwise” adaptive algorithm which employs nonuniform meshes that remain fixed for discrete periods of times, or “blocks”, see Fig. 1.2. With the proper implementation, this strategy addresses the key issues

1. **Accuracy** The projections onto new meshes occur at a relatively small set of discrete times. We use *a posteriori* error estimates to predict the effect of the projections and choose overlaps in the meshes to reduce the error induced by the mesh changes.

2. **Load Balancing** Load balancing is required only at the discrete times
demarcating blocks.

3. **Coarsening** There is no coarsening of a given mesh in the indicated strategy.

Mesh changes are handled purely as projections between different meshes.

![Diagram](image)

Fig. 1.2: The evolution of a solution with a traveling front computed using blockwise adaptivity. Two blocks are used in the computation. On each block, the space mesh is chosen to maintain the same level of control over the local residual as is achieved in the computation shown in Fig. 1.1. In addition, a sufficient degree of overlap between the two meshes (the lightly-shaded mesh region) to insure there is no loss of accuracy in projecting the solution between the two meshes. Load balancing is only required twice, once for each block.

The idea of re-meshing only after a fixed number of steps is by no means new. However, this strategy depends critically upon choosing suitable block discretizations, and thus, ultimately, on accurately predicting the behavior of the solution. The choice of block discretizations is a difficult issue that requires balancing the inefficiency of using a fixed spatial mesh inside each block against the gain in accuracy achieved by limiting projections between different meshes and the decrease in computational cost due to limiting the number of times at which load balancing is required. This is partly a computer science problem of distributing available resources, e.g. memory and compute cycles, efficiently, and partly a numerical analysis problem, e.g. determining meshes for each block and projections between blocks. The solutions of these problems require **accurate** estimates of the error in a specific quantity of interest.

We use a computable *a posteriori* error estimate to obtain the necessary information. The *a posteriori* error estimate employed in this paper yields robustly accurate estimates of the error in a specified quantity of interest in terms of a sum of space-time element contributions, see [51, 53, 49, 50, 60, 58, 18, 57, 70]. The *a posteriori* error estimates are based on duality, adjoint problems, and variational analysis. Accurate error estimates are obtained by numerically solving the linear adjoint problem related to the desired quantity of interest.

Solving adjoint problems is problematic in the context of high performance computing because of the need to store the forward solution in order to form the adjoint problem and the cost of the adjoint solve. Our approach to this problem is to perform the adjoint solves using relatively coarse scale discretizations and using a coarse scale representation of the forward solution to form the adjoint problem. The benefits are both to reduce the memory overhead involved in storing the forward solution to form the adjoint equation and to reduce the cost of the adjoint solve. This approach is motivated by the following observations.
Adjoint problems are linear and often present fewer numerical difficulties than the associated forward problems.

Solutions of adjoint problems tend to vary slowly on the scale of the discretization, whereas residuals of forward solutions tend to oscillate on the scale of the discretization.

The accuracy required of the adjoint solution, which is being used only for error estimation, is orders of magnitude less than generally desired for the forward solution.

In addition, because we use a relatively coarse discretization for the adjoint problem, it is reasonable to forgo achieving high performance efficiency when solving the adjoint problem. In other words, solving the adjoint problems and computing the *a posteriori* error estimate is treated as a pre-processing step taken before the production computations on high performance platforms.

In this paper, we focus on the problem of determining blocks, e.g., the length of times for each block, the meshes for each block that maintain accuracy in the desired information, and suitable overlap meshes for transitions between blocks from the coarse-scale adjoint solutions.

**Work in adaptive discretization.** An enormous literature on adaptive methods for differential equations has grown over nearly six decades of activity. It would be impossible to give a complete list. Likewise, we found it very difficult to untangle the history of different developments. We provide a list of citations [78, 121, 84, 69, 118, 110, 32, 63, 64, 115, 122, 124, 4, 113, 117, 5, 130, 82, 116, 123, 36, 10, 45, 37, 23, 38, 19, 22, 68, 39, 11, 21, 44, 3, 112, 40, 31, 119, 25, 24, 2, 61, 107, 7, 75, 109, 86, 104, 134, 108, 27, 79, 20, 93, 105, 106, 94, 67, 46, 88, 51, 72, 132, 76, 80, 41, 135, 12, 62, 91, 98, 87, 81, 101, 47, 54, 73, 52, 89, 53, 17, 14, 49, 50, 42, 28, 128, 126, 129, 35, 60, 16, 30, 74, 66, 102, 114, 71, 65, 26, 97, 58, 99, 55, 15, 9, 1, 103, 43, 127, 18, 6, 83, 111, 125, 57, 100, 120, 70, 8, 85, 96, 95, 77, 13, 29, 56, 92, 90, 133, 48, 34, 33, 59, 131] that seem to us to have been particularly influential.

This literature contains several different trends, e.g., local refinement versus global error control, efficiency, parallel implementation, space mesh refinement, time step refinement, refinement for different kinds of problems, and so on, throughout this time. Some of this work, including the interesting issue of convergence of adaptive algorithms [42, 99, 9], is concerned with the asymptotic behavior of adaptive algorithms as the discretizations are refined. The work considered in this paper lies at the other end of the spectrum, i.e. trying to understand how to adapt discretizations in order to obtain maximally useful information from relatively coarse discretizations and solutions that are under-resolved. We note that this is an interesting direction for many large scale applications, for which such conditions are generic.

**2. Discretization, error estimation, and adaptivity on a single block.**

Before describing the block adaptive algorithm, we review the discretization, error estimation, and adaptivity problems for a single block.

**2.1. Discretization.** We describe two finite element space-time discretizations of (1.1) called the continuous and discontinuous Galerkin methods, see [51, 54, 53, 50, 60, 58]. We partition $[0, T]$ as $0 = t_0 < t_1 < t_2 < \cdots < t_n < \cdots < t_N = T$, denoting each time interval by $I_n = (t_{n-1}, t_n]$ and time step by $k_n = t_n - t_{n-1}$ and we construct a discretization $T$ of $\Omega$ such that the union of the elements in $T$ is $\Omega$ while the intersection of any two elements is either a common edge, node, or is empty. We assume that the smallest angle of any element is bounded below by a fixed constant.
To measure the size of the elements of \( T \), we use a piecewise constant function \( h \), the so-called mesh function, defined so \( h|_\triangle = \text{diam}(\triangle) \) for \( \triangle \in T \). Similarly, we use \( k \) to denote the piecewise constant function that is \( k_n \) on \( I_n \).

The approximations are polynomials in time and piecewise polynomials in space on each space-time “slab” \( S_n = \Omega \times I_n \). In space, we let \( V \subset H^1_0(\Omega) \) denote the space of piecewise linear continuous functions defined on \( T \), where each function is zero on \( \partial \Omega \). Then on each slab, we define

\[
W^q_n = \left\{ w(x, t) : w(x, t) = \sum_{j=0}^{q} u_j(x), v_j \in V, (x, t) \in S_n \right\}.
\]

Finally, we let \( W^q \) denote the space of functions defined on the space-time domain \( \Omega \times [0, T] \) such that \( v|_{S_n} \in W^q_n \) for \( n \geq 1 \). Note that functions in \( W^q \) may be discontinuous across the discrete time levels and we denote the jump across \( t_n \) by \([w]_n = w_n^+ - w_n^-\) where \( w_n^\pm = \lim_{s \to t_n^\pm} w(s)\).

We use a projection operator into \( V \), \( P v \in V \), e.g., the \( L^2 \) projection satisfying \((P v, w) = (v, w)\) for all \( w \in V \), where \((\cdot, \cdot)\) denotes the \( L^2(\Omega) \) inner product. We use the \( \| \cdot \| \) for the \( L^2 \) norm. We also use a projection operator into the piecewise polynomial functions in time, denoted by \( \pi_n : L^2(I_n) \to P^q(I_n) \), where \( P^q(I_n) \) is the space of polynomials of degree \( q \) or less defined on \( I_n \). The global projection operator \( \pi \) is defined by setting \( \pi = \pi_n \) on \( S_n \).

**Definition 2.1.** The continuous Galerkin \( cG(q) \) approximation \( U \in W^q \) satisfies \( \overline{U}_0^0 = P u_0 \) and

\[
\begin{align*}
\int_{t_{n-1}}^{t_n} \left( (\dot{U}, v) + (\epsilon \nabla U, \nabla v) \right) dt &= \int_{t_{n-1}}^{t_n} (f(U), v) dt, \\
U_{n-1}^+ &= U_{n-1}^-.
\end{align*}
\]

(2.1)

Note that \( U \) is continuous across time nodes when the space mesh is fixed.

**Definition 2.2.** The discontinuous Galerkin \( dG(q) \) approximation \( U \in W^q \) satisfies \( \overline{U}_0^0 = P u_0 \) and

\[
\int_{t_{n-1}}^{t_n} \left( (\dot{U}, v) + (\epsilon \nabla U, \nabla v) \right) dt + ([U]_{n-1}, v^+) = \int_{t_{n-1}}^{t_n} (f(U), v) dt \\
\text{for all } v \in W^q_n, \quad 1 \leq n \leq N.
\]

(2.2)

Note that the true solution satisfies both (2.1) and (2.2).

**Example 2.3.** To illustrate, we discretize the scalar problem

\[
\begin{align*}
\dot{u} - \Delta u &= f(u), \quad (x, t) \in \Omega \times \mathbb{R}^+, \\
u(x, t) &= 0, \quad (x, t) \in \partial \Omega \times \mathbb{R}^+, \\
u(x, 0) &= u_0(x), \quad x \in \Omega,
\end{align*}
\]

(2.3)

using the \( dG(0) \) method. Since \( U \) is constant in time on each time interval, we let \( \dot{U}_n^- \) denote the \( M \) vector of nodal values with respect to the nodal basis \( \{\eta_i\}_{i=1}^M \) for \( V \). We let \( B : (B)_{ij} = (\eta_i, \eta_j) \) for \( 1 \leq i, j \leq M \) denote the mass matrix and
\[ A : (A)_{ij} = (\nabla \eta_i, \nabla \eta_j) \] denote the stiffness matrix. Then \( U_n \) satisfies
\[
(B + k_n A) \vec{U}^-_n - \vec{F}(\vec{U}^-_n)k_n = B \vec{U}^-_{n-1}, \quad 1 \leq n \leq N,
\]
where \((\vec{F}(\vec{U}^-_n))_i = (f(U^-_n), \eta_i)\).

With appropriate use of quadrature to evaluate the integrals in the variational formulation, these Galerkin methods yield standard difference schemes. In the example above, if the lumped mass quadrature is used to evaluate the coefficients of \( B \), then the resulting set of equations for the dG(0) approximation is the same as the equations for the nodal values of the backward Euler - second order centered difference scheme for (2.3). Likewise, the cG(1) method is related to the Crank-Nicolson scheme, and the dG(1) method is related to the third order sub-diagonal Padé difference scheme. Under general assumptions, the cG(q) and dG(q) have order of accuracy \( q + 1 \) in time at any point. In addition, they enjoy a superconvergence property in time at time nodes. The dG(q) method has order of accuracy \( 2q + 1 \) in time and the cG(q) method has order \( 2q \) in time at time nodes for sufficiently smooth solutions.

### 2.2. A posteriori error analysis

We begin the a posteriori error analysis by defining a suitable adjoint problem for error analysis. The adjoint problem is a parabolic problem with coefficients obtained by linearization around an average of the true and approximate solutions.

\[
\hat{f} = \hat{f}(u, U) = \int_0^1 \frac{\partial f}{\partial u}(us + U(1-s)) \, ds. \tag{2.4}
\]

The regularity of \( u \) and \( U \) typically imply that \( \hat{f} \) is piecewise continuous with respect to \( t \) and a continuous, \( H^1 \) function in space.

Written out pointwise for convenience, the adjoint problem to (1.1) for the generalized Green’s function associated to the data \( \psi \), which determines the quantity of interest,
\[
\int_0^T (u, \psi) \, dt,
\]
is

\[
\begin{cases}
-\dot{\phi} - \nabla \cdot (\epsilon \nabla \phi) - \hat{f} \phi = \psi, \quad (x, t) \in \Omega \times (T, 0], \\
\phi(x, t) = 0, \quad (x, t) \in \partial \Omega \times (T, 0], \\
\phi(x, T) = 0, \quad x \in \Omega,
\end{cases} \tag{2.5}
\]

This choice for the adjoint yields the following error representation formulas, see [58].

**Theorem 2.4.** For the cG method, we have

\[
\int_0^T (e, \psi) \, dt = ((I - P)u_0, \phi(0))
\]

\[
+ \int_0^T ((U, \pi P \phi - \phi) + (\epsilon(U)\nabla U, \nabla (\pi P \phi - \phi)) - (f(U), \pi P \phi - \phi)) \, dt. \tag{2.6}
\]
For the dG method,
\[
\int_0^T (e, \psi) \, dt = ((I - P)u_0, \phi(0)) + \sum_{n=1}^N ([U]_{n-1}, (\pi P \phi - \phi)_{n-1}^+) \\
+ \int_0^T ((\dot{U}, \pi P \phi - \phi) + (\epsilon(U) \nabla U, \nabla (\pi P \phi - \phi)) - (f(U), \pi P \phi - \phi)) \, dt.
\]

The initial error is \( e^{-}(0) = (I - P)u_0 \).

In practice, we compute a numerical solution of a linear adjoint problem obtained from (2.5). Typically, we linearize around the computed approximate solution and solve using a higher order method in space and time. Without specifying the well-described details (see [58]), we denote the approximate adjoint solution by \( \Phi \). We focus on the dG method, while application to the cG method is obvious.

**Corollary 2.5.** The approximate a posteriori error estimate for the dG method is
\[
\left| \int_0^T (e, \psi) \, dt \right| \approx E(U) = E(U; \psi) = \left| ((I - P)u_0, \Phi(0)) + \sum_{n=1}^N ([U]_{n-1}, (\pi P \Phi - \Phi)_{n-1}^+) \right| \\
+ \int_0^T ((\dot{U}, \pi P \Phi - \Phi) + (\epsilon(U) \nabla U, \nabla (\pi P \Phi - \Phi)) - (f(U), \pi P \Phi - \Phi)) \, dt.
\]

### 2.3. Approaches to adaptive error control

The aim of goal oriented adaptive error control is to generate a mesh with a nearly minimal number of elements such that for a given tolerance \( TOL \) and data \( \psi \),
\[
\left| \int_0^T (e, \psi) \, ds \right| \lesssim TOL.
\]

We note that (2.9) cannot be verified in practice because the error is unknown, so instead we use an estimate or a bound for the error in the quantity of interest. There are various ways to generate an acceptable mesh that vary, for example, by the estimate or bound used for the quantity of interest and the strategy for mesh refinement.

For example, if we solve the adjoint problem and compute the accurate a posteriori estimate (2.8), the goal of adaptive error control could be to determine a discretization so that a mesh acceptance criterion,
\[
E(U) \lesssim TOL,
\]

is satisfied. If (2.10) is not satisfied, then we refine the mesh in order to compute a new solution for which the criterion is met. Refinement decisions require identifying the contributions to the error from discretization on each element. We can write \( E(U) \)
as a sum over space-time elements,

\[
E(U) = \left| \sum_{\triangle \in T} ((I - P)u_0, \Phi(0))_\triangle + \sum_{n=1}^{N} \sum_{\triangle \in T} ([U]_{n-1}, (\pi P \Phi - \Phi)^+_n)_\triangle \\
+ \sum_{n=1}^{N} \sum_{\triangle \in T} \int_{t_{n-1}}^{t_n} ((\dot{U}, \pi P \Phi - \Phi)_\triangle + (\epsilon(U) \nabla (\pi P \Phi - \Phi))_\triangle - (f(U), \pi P \Phi - \Phi)_\triangle) \, dt \right|,
\]

where \((\ , \ )_\triangle\) denotes the \(L^2\) inner product on element \(\triangle\). This clearly identifies the element contributions. However, a major difficulty is that the error estimate generally involves a large amount of cancellation among the element contributions, which makes determining a refinement strategy extremely difficult.

**Example 2.6.** We consider a first order accurate numerical solution that has the element contributions shown in Fig. 2.1. The first time step has the largest contribution. The next three steps each contribute \(-.033\), so cancellation means that the total contribution from the first four steps is \(.001\). Likewise, the next six steps contribute \(+.003\) in total. The last four steps contribute \(.08\) in total. The total error is therefore

\[
.1 - 3 \times .033 + .011 - .01 + .011 - .01 + .011 - .01 + 4 \times .02 = .084
\]

If we use a standard approach of refining only some fraction of the elements with the largest contributions, we are likely to refine the first four steps. For simplicity, we assume that the elements marked for refinement are divided into two time steps. The resulting integration will have accuracy

\[
\frac{1}{22} \times 2 \times .1 - \frac{1}{22} \times 6 \times .033 + .011 - .01 + .011 - .01 + .011 - .01 + 4 \times .02 \approx .0835.
\]

Note that the individual element contributions will decrease at a second order rate. The problem is that even though the element contributions in the first four steps are individually large, there is significant cancelation and refinement in this region and refinement does not decrease the error significantly. On the other hand, if we refine the last four time steps instead, we obtain

\[
.1 - 3 \times .033 + .011 - .01 + .011 - .01 + .011 - .01 + \frac{1}{22} \times 8 \times .02 \approx .044.
\]
While this is a non-standard approach, it decreases the error significantly.

In the adjoint-weight approach, the issue of cancelation of error is neglected in a sense by replacing the accurate error estimate \( E(U) \) by an inaccurate upper bound, \( E(U) \leq \mathcal{E}(U; \psi) = \mathcal{E}(U; \psi) \),

(2.11)

where \( \mathcal{E}(U; \psi) \) is defined as:

\[
\mathcal{E}(U; \psi) = \sum_{\Delta \in T} |((I - P)u_0, \Phi(0))_\Delta| + \sum_{n=1}^N \sum_{\Delta \in T} \left| ([U]_{n-1}, (\pi P \Phi - \Phi)_{n-1})_\Delta \right|
\]

\[
+ \sum_{n=1}^N \sum_{\Delta \in T} \left| \int_{t_{n-1}}^{t_n} (\dot{U}, \pi P \Phi - \Phi)_\Delta + (\epsilon(U) \nabla U, \nabla (\pi P \Phi - \Phi))_\Delta - (f(U), \pi P \Phi - \Phi)_\Delta dt \right|.
\]

Thus, if (2.10) is not satisfied, the mesh is refined in order to achieve

\[
\mathcal{E}(U) \leq \text{TOL}.
\]

The adaptive error control problem can now be profitably posed as a constrained minimization problem, namely to find a mesh with a minimal number of degrees of freedom on which the approximation satisfies the bound (2.12). Using the fact that the bound \( \mathcal{E} \) is a sum of positive terms and assuming the solution is asymptotically accurate, a calculus of variations argument yields the generic

**Principle of Equidistribution** The solution of the constrained optimization problem for an optimal mesh for an upper bound on the error is achieved when the elements contributions to the bound are approximately equal.

The Principle of Equidistribution has been used in various forms at least since the seventies (and probably earlier in industry). The references [49, 50, 18, 8] contain more mathematical details.

It is important to note that practical experience with a wide range of problems suggest that the bound \( \mathcal{E} \) is generically several orders of magnitude larger than the estimate \( E(U) \) and very often much larger. Thus, an “optimal” mesh produced by a strategy based on the Principle of Equidistribution that optimizes computational cost with respect to a bound and not the actual error can lead to tremendous inefficiency in the sense of over-refinement.

An adaptive mesh algorithm is a procedure for solving the constrained minimization problem associated with (2.12). If the Principle of Equidistribution is used, then the algorithm seeks to choose meshes so that the element contributions are approximately equal.

Traditionally, different approaches for adaptive mesh algorithms are used to handle spatial meshes and time discretization. Generally for spatial meshes, a global approach is used that seeks a solution using the “compute-estimate-mark-adapt” Algorithm 1.1. During a given iteration level, the estimate or bound is computed on the current mesh, some fraction of the elements on which the element contribution is largest are marked for refinement, and those elements are refined. The iteration is continued until (2.12) is satisfied, see [50, 18, 8].

On the other hand, influenced by the long history of so-called “local error control”
The traditional time step adaptive algorithm achieves equidistribution by determining a local tolerance \( LTOL \) in some fashion and then insuring that the contribution to the error estimate from the current time interval is smaller than, but approximately equal to, \( LTOL \) before proceeding to the next time step. This can be viewed as a pessimistic approach to solving the equidistribution optimization problem. The pessimism arises from the assumption that one \( LTOL \) will work for every interval. Often, \( LTOL \) is input directly without any attempt to relate it to the desired tolerance \( TOL \). Given a true global error estimate however and the asymptotic accuracy of the integration scheme, there are various heuristic arguments for determining \( LTOL \) in terms of \( TOL \).

Note that the strategy for mesh selection used in time integration tends to yield meshes on which the element contributions are nearly equal. In contrast, element contributions on a spatial mesh computed using a compute-estimate-mark-refine strategy tend to vary in size considerably.

For a partial differential equation, space and time mesh refinement strategies have to be combined somehow. In the case of a parabolic problem, it is straightforward to distinguish the time and space contributions to the bound \( A_e \). We define the time and space bounds,

**Definition 2.8.**

\[
\begin{align*}
A_t(U) &= \sum_{n=1}^{N} \sum_{\Delta \in T} \left| ([U]_{n-1}, ((\pi - I)P\phi)_{n-1}^+) \right| \\
&+ \sum_{n=1}^{N} \sum_{\Delta \in T} \int_{t_{n-1}}^{t_n} (\dot{U}, (\pi - I)P\phi) + (\epsilon(U)\nabla U, \nabla (\pi - I)P\phi) \\
&- (f(U), (\pi - I)P\phi) \Delta dt \\
A_x(U) &= \sum_{\Delta \in T} \left| ((I - P)u_0, \Phi(0)) \right| + \sum_{n=1}^{N} \sum_{\Delta \in T} \left| ([U]_{n-1}, (P\phi - \Phi)_{n-1}^+) \right| \\
&+ \sum_{n=1}^{N} \sum_{\Delta \in T} \int_{t_{n-1}}^{t_n} (\dot{U}, P\phi - \Phi) + (\epsilon(U)\nabla U, \nabla (P\phi - \Phi)) \\
&- (f(U), P\phi - \Phi) \Delta dt.
\end{align*}
\]

We see that the time bound is precisely the \textit{a posteriori} bound for the dG approximation for the “method of lines” initial value problem resulting after discretization in space. The adjoint weight depends on the projection of the adjoint solution into the time finite element space. On the other hand, the adjoint weight in the space bound depends on the projection of the adjoint solution into the spatial finite element space.

We split the error between the time and space contributions and refine the current mesh in order to achieve

\[
A_x(U) \lesssim \frac{TOL}{2} \text{ and } A_t(U) \lesssim \frac{TOL}{2}.
\]

(2.15)
On a given time interval, this requires an iteration during which both the space mesh and time steps are refined.

3. Blockwise discretization. We describe the blockwise formulation of the discontinuous Galerkin method. We partition \([0, T]\) into time blocks \(0 = T_0 < T_1 < T_2 < \cdots < T_b < \cdots < T_B = T\). We discretize each block \([T_{b-1}, T_b]\) by \(T_{b-1} = t_{b,0} < t_{b,1} < \cdots < t_{b,N_b} = T_b\), denoting each subinterval by \(I_{b,n} = (t_{b,n-1}, t_{b,n}]\) and time step by \(k_{b,n} = t_{b,n} - t_{b,n-1}\). To each block \([T_{b-1}, T_b]\), we associate a discretization \(T_b\) of \(\Omega\) arranged so the union of the elements in \(T_b\) is \(\Omega\) while the intersection of any two elements is either a common edge, node, or is empty. We assume that the smallest angle of any element is bounded below by a fixed constant. To measure the size of the elements of \(T_b\), we use the mesh function \(h_b\).

The approximations are polynomials in time and piecewise polynomials in space on each space-time “slab” \(S_{b,n} = \Omega \times I_{b,n}\). In space, we let \(V_b \subset H^1_0(\Omega)\) denote the space of piecewise linear continuous functions defined on \(T_b\), where each function is zero on \(\partial \Omega\). Then on each slab, we define

\[
W^q_{b,n} = \left\{ w(x, t) : w(x, t) = \sum_{j=0}^q t^j v_{b,j}(x), v_{b,j} \in V_b, (x, t) \in S_{b,n} \right\},
\]

Finally, we let \(W^q\) denote the space of functions defined on the space-time domain \(\Omega \times [0, T]\) such that \(v|_{S_{b,n}} \in W^q_{b,n}\) for \(b, n \geq 1\). Note that functions in \(W^q\) may be discontinuous across the discrete time levels and we denote the jump across \(t_{b,n}\) by \([w]_{b,n} = w^+_{b,n} - w^-_{b,n}\).

To compute the dG approximation on the new block, we project the final value of the approximation from the previous block onto the new mesh. We use a projection operator \(P_b v \in V_b\) and a projection operator into the piecewise polynomial functions in time, denoted by \(\pi_{b,n} : L^2(I_{b,n}) \rightarrow P^q(I_{b,n})\). We then define \(\pi_b\) as \(\pi_b = \pi_{b,n}\) on \(S_{b,n}\). Finally, we define global projections \(P\) and \(\pi\) which on each block are \(P_b\) and \(\pi_b\) respectively.

**Definition 3.1.** The blockwise discontinuous Galerkin dG(q) approximation \(U \in W^q\) satisfies \(U|_{b,0} = P_1 u_0\) and for \(b = 1, 2, \cdots, B\),

\[
\int_{t_{b,n-1}}^{t_{b,n}} ((\dot{U}, v) + (\epsilon \nabla U, \nabla v)) dt + \left( [U]_{b,n-1}, v^+ \right) = \int_{t_{b,n-1}}^{t_{b,n}} (f(U), v) dt
\]

for all \(v \in W^q_{b,n}, 1 \leq n \leq N_b\). (3.1)

Adapting the standard argument that yields (2.7), we obtain a blockwise *a posteriori* error estimate.
Theorem 3.2.

\[ \int_0^T (e, \psi) \, dt \approx ((I - P_0)u_0, \Phi(0)) + \sum_{b=1}^B \left( ((I - P_b)U, \Phi(T_{b-1})) \right. \\
+ \int_{T_{b-1}}^{T_b} \left( \left( \dot{U}, \pi P_b \Phi - \Phi \right) + (\varepsilon(U) \nabla U, \nabla (\pi P_b \Phi - \Phi)) - (f(U), \pi P_b \Phi - \Phi) \right) \, dt \\
+ \sum_{n=1}^{N_b} \left( [U]_{b,n-1}, (\pi P_b \Phi - \Phi)^+ \right) \right). \]  

The second term on the right measures the effects of changing meshes on the accuracy of the approximation. The analog of this “jump” terms already appears in the estimate for the standard dG method. In this case, the “jump” only arises because of the mesh change between blocks and can be estimated using the indicated projection. Note that the adjoint weight does not involve the projection of \( \Phi \) into the approximation space (i.e. Galerkin orthogonality). Instead, the contributions from the projections accumulate in the same way as an initial error.

4. Blockwise adaptive computation. Our purpose is to use the a posteriori bounds \( E_x, E_t \) to choose block times \( \{T_b\} \) and corresponding meshes \( T_b \) and timesteps \( k_{b,i} \). An important issue is the effect of transferring solutions between the meshes of adjacent blocks on the accuracy of the computed information, and so we address the computation of a bound on the second term on the right in (3.2),

\[ \Xi(U) = \sum_{b=1}^B \left| ((I - P_b)U, \Phi(T_{b-1})) \right|. \]  

4.1. Modifications of standard approaches to adaptive error control. For the purpose of developing a block adaptive algorithm, we treat adaptivity with respect to space and time in the same way. The reason is that we determine the blocks by predicting the local element sizes (or number of sub-elements) that are required in the final mesh. We create a block by grouping together a set of coarse-scale space-time slabs that are adjacent in time and satisfy some criteria, e.g. similar spatial meshes are predicted for the space-time slabs in the block or a maximal number of elements are predicted to be required in the block.

We want the predictions of the element sizes needed in an acceptable fine scale mesh to be as accurate as possible. We recall that an acceptable mesh need only satisfy the estimate criterion (2.10) and not the more stringent bound criterion (2.12). We define the overestimation factor for a given mesh,

\[ \gamma = \frac{E(U)}{E(U)}, \]

and the corresponding absolute tolerance for \( E \),

\[ \text{ATOL} = \gamma \times \text{TOL}. \]
We replace (2.12) by
\[ \mathcal{E}_x(U) \lesssim \frac{\text{ATOL}}{2} \text{ and } \mathcal{E}_t(U) \lesssim \frac{\text{ATOL}}{2}. \] (4.2)

Note that \( \text{ATOL} \approx \text{TOL} \) when there is little cancellation among the element contributions and \( \text{ATOL} > \text{TOL} \) otherwise. In this way, we attempt to mitigate the inefficiency that is introduced by replacing an accurate error estimate by an inaccurate bound in decisions about mesh refinement. This approach for setting tolerances is discussed further in [59].

**4.2. Predicting an acceptable fine scale mesh.** Given a local space-time element \( \mathcal{S} = \mathcal{S}(\Delta, n) = \Delta \times [t_{n-1}, t_n] \) in the \( n \)th space-time slab that is marked for refinement, we show how to predict the number of space-time elements that are needed to meet the acceptance criterion. We assume that in the current mesh, there are \( N \) time steps and \( M \) space elements in each space-time slab, giving a total of \( NM \) space-time elements. We define a **local absolute tolerance**

\[ \text{LATOL} = \frac{\text{ATOL}}{2NM}. \]

By the Principle of Equidistribution, we adopt the goal of refining each space-time element so that the local element contribution is approximately LATOL.

Using a priori convergence analysis, see [58], it is possible to show that there is a constant \( C \) such that

\[ \mathcal{E}_x|_{\mathcal{S}(\Delta, n)} \sim C(h_\Delta)^p \] (4.3)

as \( h_\Delta \to 0 \), where \( p \) is related to the order of the finite element method in space and \( h_\Delta \) is the element size. Likewise, we can show constant \( C \) such that

\[ \mathcal{E}_t|_{\mathcal{S}(\Delta, n)} \sim Ck^q \] (4.4)

as \( k \to 0 \), where \( q \) is related to the order of the finite element method in time.

Now suppose that an element \( \mathcal{S}_{\text{new}} \) in the final mesh is obtained from \( \mathcal{S}_{\text{old}} \) in the current mesh by refinement. We have

\[ \text{LATOL} \approx \mathcal{E}_x|_{\mathcal{S}_{\text{new}}} \approx \mathcal{E}_x|_{\mathcal{S}_{\text{old}}} \times \left( \frac{h_{\Delta_{\text{new}}}}{h_{\Delta_{\text{old}}}} \right)^p. \] (4.5)

This yields a prediction for the new mesh size,

\[ h_{\Delta_{\text{new}}} \approx \left( \frac{\text{LATOL}}{\mathcal{E}_x|_{\mathcal{S}_{\text{old}}}^{1/p}} \right) \times h_{\Delta_{\text{old}}}. \] (4.6)

Recalling that \( d \) is the space dimension, this predicts that the element \( \Delta_{\text{old}} \) will have to be refined into roughly

\[ \left( \frac{h_{\Delta_{\text{old}}}}{h_{\Delta_{\text{new}}}} \right)^d = \left( \frac{\mathcal{E}_x|_{\mathcal{S}_{\text{old}}}}{\text{LATOL}} \right)^{d/p} \] (4.7)
sub-elements.

For refinement in time,
\[
\mathcal{E}_t|_{S_{\text{new}}} \approx \mathcal{E}_t|_{S_{\text{old}}} \times \left( \frac{k_{\text{new}}}{k_{\text{old}}} \right)^q \approx \text{LATOL}.
\]  

(4.8)

This yields a prediction for the new mesh size,
\[
k_{\text{new}} \approx \left( \frac{\text{LATOL}}{\mathcal{E}_t|_{S_{\text{old}}}} \right)^{1/q} \times k_{\text{old}}.
\]  

(4.9)

This predicts that the time step \( k_{\text{old}} \) will have to be refined into roughly
\[
\frac{k_{\text{old}}}{k_{\text{new}}} = \left( \frac{\mathcal{E}_t|_{S_{\text{old}}}}{\text{LATOL}} \right)^{1/q}
\]  

(4.10)

sub-intervals.

### 4.3. The block adaptive algorithm and strategies for creating blocks.

Using the development above, we now present the generic block adaptive algorithm in Algorithm 4.1. We provide a detailed algorithm in Appendix A.

**Algorithm 4.1 Block Adaptive Algorithm**

Choose the “coarse” mesh and time step

1. Compute the coarse scale numerical solution
2. Estimate the element contributions to the error for the current solution
3. Predict the number of space-time elements into which each current space-time element is to be divided using (4.7) and (4.10)
4. Build block discretizations by constructing meshes satisfying the requirements for groups of neighboring time steps
5. Compute the fine scale numerical solution using the block discretizations

We note that the Block Adaptive Algorithm 4.1 can be iterated, so that the fine scale becomes the new coarse scale, and a new fine scale is subsequently computed. In crude terms, the block adaptive Algorithm 4.1 is analogous to the core estimate-mark-refine algorithm at the heart of the generic Algorithm 1.1, but is different in the mark and refine steps. The critical step defining the block adaptive algorithm Algorithm 4.1 is the strategy used to create block discretizations. Once the blocks are identified, we can use any adaptive mesh refinement strategy for producing the actual meshes. We describe three possible strategy for determining block discretizations.

#### 4.3.1. A memory-bound strategy.

In the first strategy, we assume there is a target number of elements \( N_{\text{max}} \) in space that is maximal in some sense, e.g. the largest number of elements that can be stored in core. We form blocks by creating a union of adjacent coarse-scale space-time slabs, one slab at a time, until the projected space mesh for the block uses \( N_{\text{max}} \) elements. To create the block mesh, we use the maximum of the predicted number of elements \( N_{\text{elem}} \) on each individual element in the union forming the block. We illustrate in Fig. 4.1. The parameter \( \theta \) governs how often the mesh is replaced by a coarser mesh, where \( \theta \approx 10 \) works well in practice.
A strategy that makes use of correlations in mesh predictions. In the second strategy, we aim to choose blocks in order to use a relatively small number of elements, so $\text{Nmax}$ may be considerably smaller than for the first algorithm. This strategy forms a block by grouping together adjacent coarse-scale space-time slabs whose predicted number of elements $\text{Nelem}_{\text{children}}$ are close.

In [56], we consider the problem of detecting significant overlap of local element contributions for different computations. Following the approach there, given two vectors $\vec{v}, \vec{w}$ whose coefficients are element contributions to an error estimate, we define their correlation to be $c(\vec{v}, \vec{w}) = \vec{v} \cdot \vec{w}$. We say that $\vec{v}$ is significantly correlated with $\vec{w}$ if

$$\frac{c(\vec{v}, \vec{w})}{\| \vec{w} \|^2} > \gamma_1 \quad \text{and} \quad \frac{\| \vec{w} - \frac{c(\vec{v}, \vec{w})}{\| \vec{w} \|^2} \vec{v} \|}{\| \vec{w} \|} < \gamma_2,$$

where $0 < \gamma_1, \gamma_2$. The first condition insures that $\vec{v}$ has a suitable large projection onto $\vec{w}$ while the second condition corrects for differences in scale between $\vec{v}$ and $\vec{w}$.

We implement the new criterion for creating blocks by choosing to add the next time slab to a current block based on the correlation criterion.

Global strategies for partitioning elements. In the first two strategies for creating blocks, we sweep through time. We can also use a bisection search beginning with the original large block and subdividing to find acceptable blocks. In analog to the difference between the standard global strategy for space mesh refinement to achieve the Principle of Equidistribution and the local-error control approach, the bisection search is a global strategy that can be a more efficient way to achieve equidistribution.

Determining suitable overlaps for meshes on adjacent blocks. After the meshes for each block are determined based on the \textit{a posteriori} prediction of error, we need to estimate the effects of transferring the solution between meshes.
on adjacent blocks. See Sec. 5.1 for an example that illustrates this point. Recall that 
(4.1) provides a bound on these effects. The difficulty with using (4.1) is that we do 
not have the fine scale numerical solution \( \hat{U} \) required for that expression until after 
solving on the fine scale, whereas ideally we could predict a reasonable overlap before 
computing the expensive fine scale solution.

We list three strategies for mitigating the possibility of projection error in our 
block adaptive framework.

1. There is a very simple strategy. In forming the space mesh for the block 
\([T_{b-1}, T_b] \times \Omega\), we guide refinement by using the maximum of the element 
contributions on each individual element, taking the maximum over the time 
intervals included in the block. We may simply include the maximum over the last time interval included in the previous block, \([T_{b-2}, T_{b-1}]\), i.e., over 
the interval \([t_{b-1}, N_{b-1} - 1], t_{b-1}, N_{b-1}\). We can be even more conservative by 
including some number of the last time steps in the maximum computation.

2. We can use gradient recovery [33] to compute an approximate solution on the 
fine scale mesh in each block using the solution from the last time interval 
contained in each block. We can then directly compute \((I - P_b)\hat{U}\) for each \(b\) 
and evaluate (4.1).

3. We can evaluate (4.1) \textit{a posteriori} by evaluating \((I - P_b)\hat{U}\) using the fine scale 
forward solution and the coarse scale adjoint solution.

5. Computational Examples. We apply the block adaptive algorithms to sev-
eral prototypical examples in one and three space dimensions. The one dimensional 
examples illustrate several key points while the three dimensional examples include 
a traveling wave front, a solution that undergoes time- and space-localized perturba-
tions, and a periodic motion in a convection-dominated flow.

Details of the computations. The one dimensional examples are computed 
using the Matlab code ACES [131].

The three dimensional examples are performed on a hexahedral mesh using a tri-
linear spatial basis for the forward problem and a triquadratic basis for the adjoint. 
Local mesh refinement is accomplished by the use of hanging nodes where one hanging 
node per edge or face is allowed. Conformity of the basis is obtained by interpolation 
of the surrounding regular nodes. The use of an hierarchical octree-based data struc-
ture assists refinement but also allows for derefinement when the element indicators 
are small. For the convection driven flow problem, SUPG is employed for both the 
forward and adjoint problems. This is not an obstacle for the block-adaptive frame-
work, as we simply modify the theoretical convergence rate \(p\) in the computation of 
Nelem\_children in (4.7).

5.1. The importance of dealing with the transfer error between blocks. 
We illustrate the necessity for addressing the effect of transferring solutions between 
space-time blocks with a simple one-dimensional example involving a traveling wave.

\[
\begin{aligned}
  u_t - u_{xx} &= f(x, t), & 0 < x < 1, & 0 < t, \\
  u(0, t) &= u(1, t) = \beta(t), & 0 < t, \\
  u(x, 0) &= \tanh(\alpha(x - .2)), & 0 < x < 1,
\end{aligned}
\]

(5.1)

where \(\alpha = 50\) and \(f\) and \(\beta\) are chosen to give an exact solution 
\(u = \tanh(\alpha(x - t - .2))\).

We solve with a coarse mesh using \(h = .1\) and time step \(k = .05\) from initial time 
0 to final time \(6\). The quantity of interest is the average space-time error. We
compute a fine scale solution using two blocks derived from the coarse scale solution. The first block, \( t = [0, 0.3] \), uses a finer spatial mesh in the region \( x \in [0.1, 0.6] \), while the second block uses a fine mesh in the region \( [0.5, 1] \), so the overlap is minimal and the predictions for refinement areas are incorrect. Consequently, the approximate travelling wave travels too quickly. The first block solution at \( t = 0.3 \) and its projection onto the second block at \( t = 0.3 \) is displayed in Fig. 5.1.

In Fig. 5.1 we illustrate the \textit{a posteriori} use of (4.1) to correct the projection error. Block 1 is computed using the predicted fine scale mesh. Block 2 is tested for significant projection error using (4.1) using the fine scale solution for Block 1 and the mesh for Block 2 is refined if the elementwise projection error exceeds \( \operatorname{LATOL} \).

We note that the overlap strategy for the projection error also works well in this particular example.

### 5.2. Dependence of block predictions on the coarse scale resolution.

Since we are using the coarse scale discretization to predict the global behavior of the solution on the fine scale, it is important to insure that the coarse scale discretization is not too coarse. This is a difference between the block adaptive approach and a standard adaptive mesh refinement, which is generally started with a very coarse mesh. This issue is especially important for nonlinear problems since linearization is used to define the adjoint problem, which in turn provides the means to quantify the effects of cancellation and accumulation of errors.

Consider the one-dimensional nonlinear parabolic equation

\[
\begin{align*}
    u_t - \frac{1}{25} \Delta u &= \alpha(u - 1)(1 - u^2), & -1 < x < 1, & 0 < t < 0.6, \\
    u(0, t) &= -1, & u(1, t) &= 1, & 0 < t, \\
    u(x, 0) &= \tanh(\alpha(x - 0.2)), & -1 < x < 1, 
\end{align*}
\]  

(5.2)

We choose \( \alpha \) to obtain the same solution as the example in § 5.1, \( u = \tanh(\alpha(x - t - 0.2)) \). The quantity of interest is the average space-time error. For the coarse discretization, we use \( h = 0.05 \) and \( k = 0.05 \). These choices provide an excellent coarse scale discretization for the linear example in Sec. 5.1 but does not work well for the nonlinear version. We show two snapshots of the solution \( u \) in Fig. 5.2 at \( t = 0.3 \) and \( t = 0.6 \). The wave-speed is predicted inaccurately, which leads to a poor
block selection and this subsequently affects the fine scale accuracy. Using a coarse scale discretization with \( h = 0.1 \) and \( k = 0.1 \) yields inaccurate results.

The poor predictions by the coarse-scale discretization can be avoided by slightly enriching the discretization with a finer time step. We use a coarse discretization with \( h = 0.05 \) and \( k = 0.01 \) and the correlation strategy to produce blocks. The approximate solution on the adapted mesh at \( t = 0.45 \) is shown in Fig. 5.3.

5.3. A traveling wave solution. The next example is a wave propagating along the main diagonal of the unit cube (\( \Omega = [0, 1] \times [0, 1] \times [0, 1] \)). The governing equation is

\[
\begin{aligned}
\begin{cases}
\frac{\partial u}{\partial t} - \Delta u = f(x, t), & x \in \Omega, 0 < t, \\
u(x, 0) = 0, & x \in \partial \Omega, 0 < t, \\
u(x, 0) = (x_3 - x_1^2)(x_1 - x_2^2)(x_2 - x_3^2) \times \arctan(\frac{c\sqrt{3}/3}{x_1^2 + x_2^2 + x_3^2}), & x \in \Omega,
\end{cases}
\end{aligned}
\tag{5.3}
\]

where \( c = 75 \) and \( f \) is constructed to yield the exact solution

\[
u = \frac{\sqrt{3}}{3} \arctan(\frac{c\sqrt{3}/3}{x_1^2 + x_2^2 + x_3^2} - t).
\]

The coarse block solution \( u_C \) is constructed on an \( 8 \times 8 \times 8 \) uniform mesh using
hexahedral meshes with an initial time step of .1. We solve the coarse scale solution using continuous piecewise linear elements in space and the dG(0) scheme in time. The quantity of interest is the time average of the solution value. The memory bound strategy is used to construct the discretization blocks with $ATOL = 0.000178$ and $\text{Nmax} = 50000$. Block information is given in Table 5.1. As might be expected, all of the blocks use approximately the same number of elements.

<table>
<thead>
<tr>
<th>Block</th>
<th>$T_{b-1}$</th>
<th>$T_b$</th>
<th># vertices</th>
<th># hexahedra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.4</td>
<td>58711</td>
<td>50394</td>
</tr>
<tr>
<td>2</td>
<td>0.4</td>
<td>0.6</td>
<td>63219</td>
<td>54503</td>
</tr>
<tr>
<td>3</td>
<td>0.6</td>
<td>0.7</td>
<td>72267</td>
<td>61265</td>
</tr>
<tr>
<td>4</td>
<td>0.7</td>
<td>0.8</td>
<td>62626</td>
<td>52368</td>
</tr>
<tr>
<td>5</td>
<td>0.8</td>
<td>1</td>
<td>64764</td>
<td>54860</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1.1</td>
<td>62790</td>
<td>54377</td>
</tr>
</tbody>
</table>

Table 5.1: Problem (5.3): Blocks resulting from the memory bound strategy.

We show “slices” of some of the block meshes along the plane $x = 0.5$ in Fig. 5.4.

Fig. 5.4: Problem (5.3). Memory bound strategy. Slices through the mesh perpendicular to the $x$-axis at $x = 0.5$. Upper left: $t = 0$ (block 1). Upper right: $t = 0.44$ (block 2). Lower left: $t = 0.6$ (block 3). Lower right: $t = 1.1$ (block 6).

5.4. An example with sources that are localized in space and time. The second example contrasts the difference in block meshes produced by the memory
bound and correlation strategies when solving an equation with source terms that are localized in space and time. The governing equation on the unit cube $\Omega$ is

$$\begin{align*}
\frac{\partial u}{\partial t} - \Delta u &= 50e^{-(\alpha_1(x-x_1)^2 + (t-t_1)^2)} + 20e^{-(\alpha_2(x-x_2)^2 + (t-t_2)^2)}, \quad x \in \Omega, 0 < t, \\
u(x, 0) &= 0,
\end{align*}$$

(5.4)

along with homogeneous Neumann boundary conditions on all the sides except the bottom where a homogeneous Dirichlet condition is imposed. We choose $\alpha_1 = 50$, $\alpha_2 = 10$, $t_1 = 1$, $t_2 = 10$, $x_1 = (0.125, 0.125, 0.125)$, and $x_2 = (0.75, 0.5, 0.75)$. The quantity of interest is the time average of the solution value.

We use a coarse discretization consisting of an $8 \times 8 \times 8$ uniform hexahedral mesh and time step of $0.1$. We show the block information for the memory bound and correlation strategies with $ATOL = 0.0010044$ and $N_{\text{max}} = 50000$ respectively in Table 5.2 and Table 5.3. The algorithms lead to significantly different block meshes. The correlation strategy chooses many more blocks, but many of the blocks have very low numbers of elements.

We show planar slices near $x_1$ and $x_2$ of the meshes for Blocks 1 and 3 in Fig. 5.5. For comparison, we show planar slices perpendicular to the $x$-axis near $x_1$ and $x_2$ of the meshes for blocks constructed using the two strategies in Fig. 5.6. Both strategies result in similar meshes near $x_2$ at time $t = 10$. However, at $t = 8.8$, the correlation strategy leads to coarse meshes that are not produced by the memory bound strategy. The mesh from resulting from the the memory bound strategy retains the refinement

Table 5.2: Problem (5.4). Blocks resulting from the memory bound strategy.

<table>
<thead>
<tr>
<th>Block</th>
<th>$T_{b-1}$</th>
<th>$T_b$</th>
<th># vertices</th>
<th># hexahedra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1.1</td>
<td>59465</td>
<td>54125</td>
</tr>
<tr>
<td>2</td>
<td>1.1</td>
<td>1.2</td>
<td>63112</td>
<td>57772</td>
</tr>
<tr>
<td>3</td>
<td>1.2</td>
<td>2.4</td>
<td>45359</td>
<td>40958</td>
</tr>
<tr>
<td>4</td>
<td>2.4</td>
<td>11.9</td>
<td>12383</td>
<td>10165</td>
</tr>
<tr>
<td>5</td>
<td>11.9</td>
<td>14.9</td>
<td>2029</td>
<td>1478</td>
</tr>
</tbody>
</table>

Table 5.3: Problem (5.4). Blocks resulting from the correlation strategy.

<table>
<thead>
<tr>
<th>Block</th>
<th>$T_{b-1}$</th>
<th>$T_b$</th>
<th># vertices</th>
<th># hexahedra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1.1</td>
<td>63112</td>
<td>57772</td>
</tr>
<tr>
<td>2</td>
<td>1.1</td>
<td>1.2</td>
<td>63112</td>
<td>57772</td>
</tr>
<tr>
<td>3</td>
<td>1.2</td>
<td>1.6</td>
<td>45359</td>
<td>40958</td>
</tr>
<tr>
<td>4</td>
<td>2.5</td>
<td>2.9</td>
<td>966</td>
<td>652</td>
</tr>
<tr>
<td>5</td>
<td>2.9</td>
<td>8.5</td>
<td>966</td>
<td>652</td>
</tr>
<tr>
<td>7</td>
<td>8.5</td>
<td>9</td>
<td>2617</td>
<td>1926</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10.8</td>
<td>12651</td>
<td>10382</td>
</tr>
<tr>
<td>9</td>
<td>10.8</td>
<td>11.3</td>
<td>7363</td>
<td>5860</td>
</tr>
<tr>
<td>10</td>
<td>11.3</td>
<td>12.6</td>
<td>3139</td>
<td>2360</td>
</tr>
<tr>
<td>11</td>
<td>12.6</td>
<td>14.9</td>
<td>729</td>
<td>512</td>
</tr>
</tbody>
</table>
resulting from the earlier perturbation near $x_1$ at $t = 1$.

\begin{align*}
\begin{cases}
  u_t + \beta \cdot \nabla u - \Delta u = f, & \quad x \in \Omega, 0 < t < 1, \\
  u(x, t) = 0, & \quad x \in \partial \Omega, 0 < t < 1, \\
  u(x, 0) = 0, & \quad x \in \Omega,
\end{cases}
\end{align*}

\tag{5.5}

with

$$\beta = b \times \begin{pmatrix}
\cos(\pi ct) \times \sin(\pi dt) \\
\sin(\pi ct) \times \sin(\pi dt) \\
\cos(\pi dt)
\end{pmatrix},$$

$b = 20$, $c = 1$, $d = 2$, and $f(x) = e^{-50(x_1^2 + x_2^2 + x_3^2)}$. The quantity of interest is the time average value. The coarse discretization used 4913 vertices and at time step of 0.01.

The information for the blocks constructed by the memory-bound strategy using $ATOL = 0.00044$ and $N_{max}=50000$ is given in Table 5.4.

We provide “slices” perpendicular to the $x$-axis at $x = 0.5$ through some of the
Fig. 5.6: Problem (5.4). Slices through the mesh perpendicular to the $x$-axis. Left: Correlation strategy. Slice near $x_2$ at $t = 10$ (block 8). Middle: Correlation strategy. Slice near $x_1$ at $t = 8.8$ (block 7). Right: Memory bound strategy. Slice near $x_1$ at $t = 8.8$ (block 4).

<table>
<thead>
<tr>
<th>Block</th>
<th>$T_b$</th>
<th>$T_{b+1}$</th>
<th># vertices</th>
<th># hexahedra</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.09</td>
<td>58799</td>
<td>51066</td>
</tr>
<tr>
<td>2</td>
<td>0.09</td>
<td>0.15</td>
<td>58424</td>
<td>50289</td>
</tr>
<tr>
<td>3</td>
<td>0.15</td>
<td>0.27</td>
<td>58393</td>
<td>50359</td>
</tr>
<tr>
<td>4</td>
<td>0.27</td>
<td>0.61</td>
<td>59102</td>
<td>50744</td>
</tr>
<tr>
<td>5</td>
<td>0.61</td>
<td>0.99</td>
<td>28395</td>
<td>23388</td>
</tr>
</tbody>
</table>

Table 5.4: Problem (5.5). Blocks resulting from the memory bound strategy.

6. Conclusion. We describe and test an adaptive algorithm for evolution problems that use a sequence of “blocks” which employ fixed, non-uniform space meshes. A block adaptive strategy reduces the number of mesh changes that must be treated and provides a way to quantify the effects of mesh changes on accuracy. A major issue with such an approach is determining good block discretizations from coarse scale solution information. We describe a way to do this using adjoint-base a posteriori error estimates, where the adjoint solution is computed using a coarse scale solution. We demonstrate the behavior of the proposed algorithms as well as several technical issues in a set of examples.

REFERENCES

Fig. 5.7: Problem (5.5). Memory bound strategy. Slices through the mesh perpendicular to the $x$-axis at $x = 0.5$. Upper left: $t = 0.04$ (block 1). Upper right: $t = 0.16$ (block 3). Lower left: $t = 0.42$ (block 4). Lower right: $t = 0.62$ (block 5).
 BLOCKWISE ADAPTIVITY

25

of influence linear model problem ods

equation with variable coefficients

differential equations

differential equations

differential equations

differential equations

equation

A posteriori error bounds and global error control for approximations of ordinary

A posteriori error estimates based on residuals and variational analysis

The relative efficiency of alternative defect control schemes for high-order continuous


A posteriori error bounds and global error control for approximations of ordinary

Generalized Green’s functions and the effective domain


Cancellation of error and adaptive error control for ordinary


A-posteriori error analysis and adaptive processes in the finite- element method 2. Adaptive mesh refine-
Detailed adaptive algorithm. We use the following notation and variables to give a detailed block construction algorithm.

- $N_{\text{timestep}}$ = current number of time steps
- $N_{\text{elem}}(j)$ = number of space elements in timestep $j$, i.e., for $t \in [t_{j-1}, t_j]
- N_{\text{timestep,children}}(j)$ = number of subintervals in to which timestep $j$ is to be divided
- $N_{\text{elem,children}}(i, j)$ = number of subelements in to which finite element $i$ is to be divided in timestep $j$
- The $b$th “block” is time interval $t \in [T_{b-1}, T_b] = [t_{b,0}, t_{b,N_b}]
- The $b$th “block” comprises timesteps $j_{b-1}, \ldots, j_b$, i.e., $N_b = j_b - j_{b-1}$, $t_{b,0} = t_{b-1}$ and $t_{b,N_b} = t_b$
- $\text{Block}(i, b)$ = number of intervals the parent element $i$ will be divided into on block $b$
- $N_{\text{elem,block}}(b)$ = number of elements in block $b$

We use the MATLAB colon operator : to denote the full row or column in a two-dimensional array. The parameter $\theta$ governs how often the mesh is replaced by a coarser mesh, where $\theta \approx 10$ works well in practice.

We present the algorithm in Algorithm 6.1.
Algorithm 6.1 A memory-bound strategy

Input error tolerance $\text{TOL}$, maximum number of elements in any block $N_{\text{max}}$, the initial coarse-scale discretization for the forward problem, and the coarse-scale discretization for the adjoint problems

Solve forward problem for $U$ on $[0, T]$
Project forward solution onto coarse-scale adjoint problem mesh
Solve adjoint problem on coarse scale mesh and compute $E(U)$

Compute LATOL $\mathcal{E}_x$, $\mathcal{E}_t$

for $j = 1, \ldots, \text{Ntimesteps}$ do
    Compute $\text{Ntimestep}_\text{children}(j)$
    for $i = 1, \ldots, \text{Nelem}(j)$ do
        Compute $\text{Nelem}_\text{children}(i,j)$
    end for
end for

Ntimesteps $\leftarrow \sum_{j=1}^{\text{Ntimesteps}} \text{Ntimestep}_\text{children}(j)$

Each subinterval of $[t_{j-1}, t_j]$ inherits $\text{Nelem}_\text{children}(i,j)$

$b = 1$, $T_0 = 0$, $T_1 = k_1$, $j_0 = 1$, $j = 2$
$\text{block}(;b) \leftarrow \text{Nelem}_\text{children}(;1)$
$\text{Nelem}_\text{block}(b) \leftarrow \sum_i \text{block}(i,b)$

while $T_b < T$ do
    while $\text{Nelem}_\text{block}(b) < N_{\text{max}}$ and $\text{Nelem}_\text{block}(b) < \theta \times \sum_{i=1}^{\text{Nelem}(j)} \text{Nelem}_\text{children}(i,j)$ do
        $j_b \leftarrow j$
        $T_b \leftarrow T_b + k_j$
        $\text{block}(;b) \leftarrow \max[\text{block}(;b), \text{Nelem}_\text{children}(;j)]$
        $\text{Nelem}_\text{block}(b) = \sum_i \text{block}(i,b)$
        $j \leftarrow j + 1$
    end while
    $b \leftarrow b + 1$
end while

for $i = 1, \ldots, b$ do
    Compute new mesh for block $b$.
    Optional: Estimate projection error and correct predicted meshes if necessary
end for

for $i = 1, \ldots, b$ do
    Solve forward problem on block $b$ for $U$.
    Project $U$ onto mesh for block $b + 1$.
    Optional: Compute projection error between blocks and correct meshes
end for

To implement the correlation-based strategy, we simply alter the block selection criteria ($\sum \text{block}(b) \leq N_{\text{max}}$) with a step which accepts a block if $\text{block}(;b)$ is correlated to $\text{Nelem}_\text{children}(;j)$ and $\text{Nelem}_\text{block}(b)$ is less than $N_{\text{max}}$.

The algorithm assumes that the blocks are always generated (even on repeat solve cycles) using the coarse mesh as a base. The algorithm may be easily modified to
work recursively on the blocks. It may also be modified, with a little more care, to allow merging and splitting of blocks during repeated solves.