Geophysical-astrophysical spectral-element adaptive refinement (GASpAR): Object-oriented h-adaptive code for geophysical fluid dynamics simulation

Duane Rosenberg [a] Aimé Fournier [a] Paul Fischer [c]

Annick Pouquet [b]

[a] *Institute for Mathematics Applied to Geosciences*

[b] *Earth and Sun Systems Laboratory*

National Center for Atmospheric Research

PO Box 3000, Boulder, Colorado 80307-3000 USA

[c] *Mathematics and Computer Science Division*

Argonne National Laboratory, Illinois 60439-4844 USA

Abstract

We present an object-oriented geophysical and astrophysical spectral-element adaptive refinement (GASpAR) code for application to turbulent flows. Like most spectral-element codes, GASpAR combines finite-element efficiency with spectral-method accuracy. It is also designed to be flexible enough for a range of geophysics and astrophysics applications where turbulence or other complex multiscale problems arise. For extensibility and flexibilty the code is designed in an object-oriented manner. The computational core is based on spectral-element operators, which are represented as objects. The formalism accommodates both conforming and non-

conforming elements and their associated data structures for handling interelement communications in a parallel environment. Many aspects of this code are a synthesis of existing methods; however, we focus on a new formulation of dynamic adaptive refinement (DARe) of nonconforming h-type. This paper presents the code and its algorithms; we do not consider parallel efficiency metrics or performance. As a demonstration of the code we offer several two-dimensional test cases that we propose as standard test problems for comparable DARe codes. The suitability of these test problems for turbulent flow simulation is considered.

Key words: spectral element, numerical simulation, adaptive mesh, AMR

1 Introduction

Accurate and efficient simulation of strongly turbulent flows is a prevalent challenge in many atmospheric, oceanic, and astrophysical applications. New numerical methods are needed to investigate such flows in the parameter regimes that interest the geophysics communities. Turbulence is one of the last unsolved classical physics problems, and such flows today form the focus of numerous investigations. They are linked to many issues in the geosciences, for example , in meteorology, oceanography, climatology, ecology, solar-terrestrial interactions, and solar fusion, as well as dynamo effects, specifically, magnetic-field generation in cosmic bodies by turbulent motions. Nonlinearities prevail in turbulent flows when the Reynolds number Re is large. The number of degrees of freedom (d.o.f.) increases as $\mathsf{Re}^{9/4}$ for $\mathsf{Re} \gg 1$ in the Kolmogorov

Email addresses: `duaner@ucar.edu` (Duane Rosenberg), `fournier@ucar.edu` (Aimé Fournier), `fischer@mcs.anl.gov` (Paul Fischer), `pouquet@ucar.edu` (Annick Pouquet).

(1941) framework. For aeronautic flows often $\mathsf{Re} > 10^6$, but for geophysical flows often $\mathsf{Re} \gg 10^8$; [29, 10] for this and other reasons the ability to probe large Re, and to examine in detail the large-scale behavior of turbulent flows depends critically on the numerical ability to resolve a large number of spatial and temporal scales.

Theory demands that computations of nonconvective turbulent flows reflect a clear separation between the energy-containing and dissipative scale ranges. Uniform-grid convergence studies on 3D compressible-flow simulations show that in order to achieve the desired scale separation, uniform grids must contain at least 2048^3 cells [35]. Today such computations can barely be accomplished. A pseudo-spectral Navier-Stokes code on a grid of 4096^3 uniformly spaced points has been run on the Earth Simulator [18], but the Taylor Reynolds number ($\propto \sqrt{\mathsf{Re}}$) is still no more than ≈ 700, very far from what is required for most geophysical flows. Similar scale separation is required in computations of convective-turbulent flows. However, if the flow's significant structures are indeed sparse, so that their dynamics can be followed accurately even if they are embedded in random noise, then dynamic adaptivity seems to offer a means for achieving otherwise unattainable large $\sqrt{\mathsf{Re}}$ values.

We have developed a dynamic geophysical and astrophysical spectral-element adaptive refinement (GASpAR) code for simulating and studying turbulent phenomena. Several properties of spectral-element methods (SEMs) make them desirable for direct numerical simulation (DNS) or large-eddy simulation (LES) of geophysical turbulence. Perhaps most significant is the fact that SEM are inherently minimally diffusive and dispersive. This property is clearly important when trying to simulate high Re number (low viscosity) flows that characterize turbulent behavior. Also, because SEMs use a finite el-

ement as a fundamental description of the geometry [31], they can be used in very efficient high-resolution turbulence studies in domains with complicated boundaries. Such discretizations also enable SEMs to be naturally parallelizable ([e.g., 15]), an important feature when simulating flows at high Re with many d.o.f. involving multiple spatial and temporal scales. Equally important, spectral element methods not only enjoy spectral convergence properties when the solution is smooth but are also effective when the solution is not smooth [9] (see (A.7) below).

In the case of SEMs, conforming adaptive methods (where entire element edges geometrically coincide, as in Fig. 1) are gradually being replaced by nonconforming adaptive methods. One reason is that mesh generation for conforming methods is complicated when attempting to resolve local flow features [32]. Another reason is that adaptive conforming meshes can lead to high-aspect-ratio elements that can cause difficulties for a linear solver [12]. Moreover, the fact that nonconforming elements can better localize mesh refinement implies that the computational cost among all elements can be reduced [23, 30]; with conforming adaption, however, local refinement regions may extend out to where refinement is not dictated by local features of interest within the solution.

Nonconforming element discretizations can be *geometrically* and/or *functionally* nonconforming. In the former case (Fig. 2), neighboring elements have boundaries that do not coincide; in the latter, the polynomial expansion degree p in neighboring elements can differ. Several SEM researchers have adopted the discretization method that simultaneously alters element size h and configuration (h-refinement) *and* the polynomial degree p in neighboring elements (p-refinement), providing for a so-called h-p-refinement strategy. The *mortar*

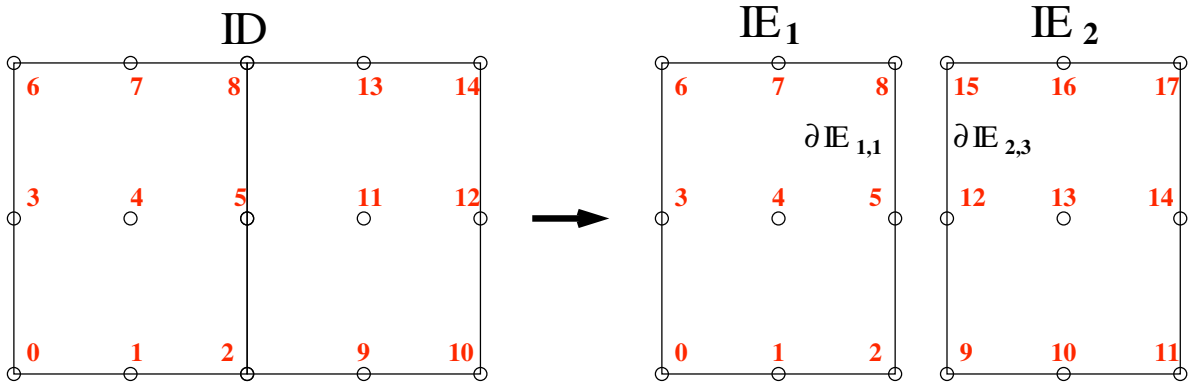

Fig. 1. Schematic of a conforming degree $p = 2$ mesh showing the mapping of global (i.e., unique) d.o.f. in the problem domain $\mathbb{D}$ (left) to local (i.e., redundant) d.o.f. in the elements $\mathbb{E}_k$ (right). Edge subscripts give element index k and edge index from $s = 0$ (south edge) counterclockwise to $s = 3$. The element $\mathbb{E}_1$ is bounded at the east by the edge $\partial\mathbb{E}_{1,1}$ and $\mathbb{E}_2$ is bounded at the west by the edge $\partial\mathbb{E}_{2,3} = \partial\mathbb{E}_{1,1}$. The interface matching condition occurs by simple assignment leading to a Boolean assembly matrix $\mathbf{A}_c$.

element method (MEM) [1, 4, 25] is a nonconforming discretization method that uses a variational formulation to minimize the Lebesgue $\mathbb{L}_2$ norm of the discontinuous jump across nonconforming spectral-element boundaries. In a number of recent applications, this method has been used in unstructured (static, nonadaptive) simulations of turbulence [16] and for ocean dynamics [19, 24]. This method has been shown to produce optimal convergence in the incompressible Stokes equations solution [3], and it has been demonstrated experimentally to produce excellent results when used as a basis for adaptive mesh refinement [28].

Nonconforming h-p adaptive methods using MEM have been developed for studying turbulent phenomena [17], ocean modeling [24], flame front deformation [11], electromagnetic scattering [22], wave propagation [6], seismology [7] and other topics. However, MEM for p-type refinement has been cited as caus-

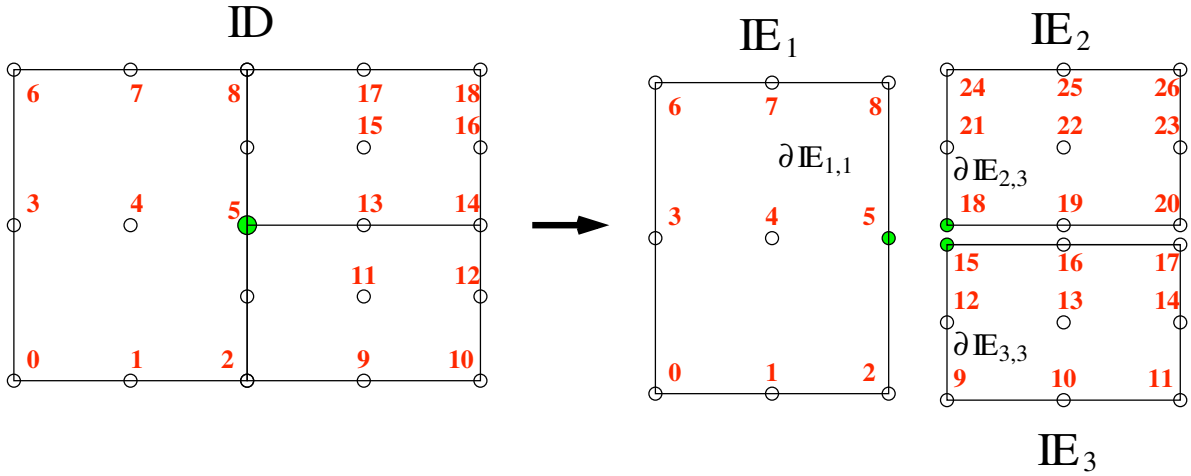

Fig. 2. As in Fig. 1 but for a geometrically *non*conforming (but functionally conforming—each element has $p = 2$) mesh. Here $\mathbb{E}_2$ and $\mathbb{E}_3$ are bounded at the west by "child" edges $\partial\mathbb{E}_{2,3}$ and $\partial\mathbb{E}_{3,3}$, and $\mathbb{E}_1$ is bounded at the east by the "parent" edge $\partial\mathbb{E}_{1,1} = \partial\mathbb{E}_{2,3} \bigcup \partial\mathbb{E}_{3,3}$. The mapping occurs by means of an interpolation of global d.o.f. from the function space associated with the parent edge onto the union of those associated with the child edges, which contains the function space of the parent.

ing instabilities in flow calculations [32]; for this reason the interpolation-based method was developed. Also, in most flows of interest to us, it is the different scales' interaction that determines not only the structures that form but also their statistics and time evolution. This suggests that reasonably high-order approximations are required in each element during much of the evolution. Thus, in the present work, using the fact that the numerical solution order is relatively high, we restrict ourselves to a nonconforming h-refinement strategy only and use an *interpolation-based* scheme to maintain continuity between nonconforming elements [12, 23]. (Throughout the remainder of this paper "nonconforming" will refer to geometrically nonconforming elements, keeping the expansion degree p fixed in all elements.) Researchers have recently used this nonconforming treatment as a basis for performing DARe [34]; however,

to the best of our knowledge, our implementation of this interpolation scheme in the fully dynamic adaptivity context is unique.

Our purpose in this paper is to describe GASpARand, in particular, the procedures used in the DARe technique. We first describe (Section 2.1) SEM discretization on a particular class of problems and introduce many of the required formulas, operators, and so forth. We discuss (Section 2.3.1) the type of nonconforming discretization allowed and the way in which continuity is maintained between elements sharing nonconforming interfaces. We explain the linear-solver details in Section 2.4. In Section 2.5 we consider how nonconforming edges are identified and how neighboring elements are found, we and present the rules for refinement and coarsening. In Section 2.5.2 we present the gather/scatter matrix that facilitates communication of edge data. A priori error estimators are discussed in Section 2.5.3 as a means to provide DARe criteria. Then, in Section 3 we consider two test-problem classes: solutions of the heat equation and linear advection-diffusion equation (Section 3.2), which demonstrate feature tracking of smooth and isolated features governed by dynamics of linear systems; and the Burgers equation, that tests the ability of DARe to capture and track well-defined and reasonably sharp structures (Section 3.3) that arise from nonlinear dynamics. In Section 4 we offer some conclusions on our current work, as well as some comments on the application of GASpAR to geophysical turbulence simulations.

2 Methodology

Because we wish to focus on the DARe methodology of GASpAR, we concentrate on the simplest multidimensional nonlinear equation that encom-

passes many of the difficulties in simulating fluid turbulence. Thus in this section we present the discrete form for the 2D Burgers equation and its variants. In turn we present the spatial discretization using a variational formulation and the spectral-element operators that arise. We then present the time discretizations.

2.1 Discretization of the dynamics

The equations considered in this work derive from the d-dimensional advection-diffusion equation for velocity $\vec{u}(\vec{x},t)$:

$$\partial_t \vec{u} + \vec{c} \cdot \vec{\nabla} \vec{u} = \nu \nabla^2 \vec{u}, \tag{1}$$

where $\nu \propto \mathsf{Re}^{-1}$ is the kinematic viscosity. This is to be solved in a spatiotemporal domain $(\vec{x},t) \in \mathbb{D} \times]0, t_{\mathrm{f}}]$ subject to the boundary and initial conditions

$$\vec{u}(\vec{x},t) = \vec{b}(\vec{x},t) \quad \text{for} \quad (\vec{x},t) \in \partial\mathbb{D} \times]0, t_{\mathrm{f}}]\,, \tag{2}$$

$$\vec{u}(\vec{x},0) = \vec{u}_{\mathrm{i}}(\vec{x}) \quad \text{for} \quad \vec{x} \in \mathbb{D}. \tag{3}$$

In this work, $\vec{c}$ may be $\vec{u}$, so that (1) is the Burgers equation, or $\vec{c} = \vec{c}(t)$, a prescribed uniform linear advection velocity.

2.1.1 Variational approach to spatial discretization

Define the function space

$$\mathbb{U}_{\vec{b}} \equiv \left\{ \vec{u} = \sum_{\mu=1}^{d} u^\mu \vec{\imath}^\mu \;\middle|\; u^\mu \in \mathbb{H}^1(\mathbb{D})\ \forall \mu \quad \& \quad \vec{u} = \vec{b} \text{ on } \partial\mathbb{D} \right\},$$

where $\vec{\imath}^\mu$ denotes the Cartesian unit vectors and

$$\mathbb{H}^1(\mathbb{D}) \equiv \{ f \;|\; f \in \mathbb{L}_2(\mathbb{D}) \quad \& \quad \partial_{x^\mu} f \in \mathbb{L}_2(\mathbb{D})\ \forall \mu \}\,.$$

Then the discretization of (1) is based on the following variational form: Find the trial function $\vec{u}(\cdot,t) \in \mathbb{U}_{\vec{b}}$ such that for any test function $\vec{v} \in \mathbb{U}_{\vec{0}}$

$$\langle \vec{v}, \partial_t \vec{u} \rangle + \langle \vec{v}, \mathcal{C}\vec{u} \rangle = -\nu \left\langle \vec{\nabla}\vec{v}^{\mathrm{T}}, \vec{\nabla}\vec{u} \right\rangle, \tag{4}$$

where $\mathcal{C} \equiv \vec{c} \cdot \vec{\nabla}$ is the advection operator and the inner product is (A.23). Note that the treatment of (3) will not be made explicit but may be easily inferred from the general discussion.

We assume that $\mathbb{D}$ can be partitioned as in (A.15). (See the appendix for the complete mathematical details.) To discretize (4), we adopt a Gauss-Lobatto-Legendre (GLL) basis (A.20), that is, expand u^μ and v^μ using (A.18). Inserting these expansions into (4), we arrive at the semi-discrete ODE system problem: Find the numerical solution $\vec{u}_{\mathrm{n}}(\cdot,t) = \vec{\boldsymbol{\phi}}^{\mathrm{T}}\boldsymbol{u}(t) \in \mathcal{P}_{\boldsymbol{h},\vec{p}}\mathbb{U}_{\vec{b}}$ such that for all $\vec{v} = \vec{\boldsymbol{\phi}}^{\mathrm{T}}\boldsymbol{v} \in \mathcal{P}_{\boldsymbol{h},\vec{p}}\mathbb{U}_{\vec{0}}$,

$$\boldsymbol{v}^{\mathrm{T}}\mathbf{M}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} + \boldsymbol{v}^{\mathrm{T}}\mathbf{C}\boldsymbol{u} = -\nu\boldsymbol{v}^{\mathrm{T}}\mathbf{L}\boldsymbol{u} \tag{5}$$

collocated at KN_p^d mapped Lagrange node points (A.17), where $\mathbf{M} = \mathrm{diag}_k\, \mathbf{M}_k$, $\mathbf{C} = \mathrm{diag}_k\, \mathbf{C}_k$, and $\mathbf{L} = \mathrm{diag}_k\, \mathbf{L}_k$ are the unassembled block-diagonal mass matrix, linear or nonlinear advection matrix [cf. 9, Ch. 6], and diffusion matrix, respectively. The respective $dN_p^d \times dN_p^d$ matrix blocks locally indexed to element $\mathbb{E}_k$ are

$$M^{\mu,\mu'}_{\vec{\jmath},\vec{\jmath}';k} \equiv \left\langle \vec{\phi}^{\mu}_{\vec{\jmath},k}, \vec{\phi}^{\mu'}_{\vec{\jmath}',k} \right\rangle_{\mathrm{GL}} = \delta_{\vec{\jmath},\vec{\jmath}'}\delta^{\mu,\mu'} w_{\vec{\jmath},k}, \tag{6}$$

including the quadrature weights (A.22),

$$C^{\mu,\mu'}_{\vec{\jmath},\vec{\jmath}';k} \equiv \left\langle \vec{\phi}^{\mu}_{\vec{\jmath},k}, \mathcal{C}\vec{\phi}^{\mu'}_{\vec{\jmath}',k} \right\rangle_{\mathrm{GL}} = \delta^{\mu,\mu'} w_{\vec{\jmath},k}\vec{c}_{\vec{\jmath},k} \cdot \vec{\nabla}\phi_{\vec{\jmath}',k}(\vec{x}_{\vec{\jmath},k}), \tag{7}$$

where $\vec{c}_{\vec{\jmath},k}(t) \equiv \vec{c}(\vec{x}_{\vec{\jmath},k}, t)$, and

$$L^{\mu,\mu'}_{\vec{\jmath},\vec{\jmath}';k} \equiv \left\langle \vec{\nabla}\vec{\phi}^{\mu}_{\vec{\jmath},k}, \vec{\nabla}\vec{\phi}^{\mu'}_{\vec{\jmath}',k} \right\rangle_{\mathrm{GL}} = \delta^{\mu,\mu'} \sum_{\vec{\jmath}''\in\mathbb{J}} w_{\vec{\jmath}'',k}\vec{\nabla}\phi_{\vec{\jmath},k}(\vec{x}_{\vec{\jmath}'',k}) \cdot \vec{\nabla}\phi_{\vec{\jmath}',k}(\vec{x}_{\vec{\jmath}'',k}).$$

To compute $\vec{\nabla}\phi_{\vec{\jmath},k}$, one differentiates (A.4) and uses (A.19). The weak diffusion matrix for deformed quadrilaterals or deformed cubes (nonlinear maps $\vec{\vartheta}_k$) can also be constructed (e.g., [9]). While these elements are supported in GASpAR, they are not necessary for the present discussion.

Restricting ourselves for the moment to the kth element $\mathbb{E}_k$, we see that (5) must hold for the restriction $\vec{v}|_{\bar{\mathbb{E}}_k} = \vec{\boldsymbol{\phi}}_k^{\mathrm{T}}\boldsymbol{v}_k$ of $\vec{v}$ to any $\mathbb{E}_k$, so that the unassembled ODE for $\vec{u}_{\mathrm{n}}|_{\bar{\mathbb{E}}_k} = \vec{\boldsymbol{\phi}}_k^{\mathrm{T}}\boldsymbol{u}_k$ is

$$\mathsf{M}_k \frac{\mathrm{d}\boldsymbol{u}_k}{\mathrm{d}t} + \mathsf{C}_k\boldsymbol{u}_k = -\nu\mathsf{L}_k\boldsymbol{u}_k. \tag{8}$$

Strictly speaking, (8) is true only after being assembled in the manner discussed in Section 2.3.2. Continuity of $\vec{u}_{\mathrm{n}}$ across all elements is a sufficient condition for maintaining $u_{\mathrm{n}}^{\mu} \in \mathbb{H}^1(\mathbb{D})$. We allow two element configuration types, conforming and nonconforming, as illustrated in Figs. 1 and 2, respectively. Conforming discretizations enforce continuity simply by assigning the same $\vec{u}_{\mathrm{n}}$ values to the coinciding node points $\vec{x}_{\vec{\jmath},k} = \vec{x}_{\vec{\jmath}',k'}$ along element edges $\partial\mathbb{E}_{k,s} = \partial\mathbb{E}_{k',s'}$. In the nonconforming case $\partial\mathbb{E}_{k,s} \subsetneq \partial\mathbb{E}_{k',s'}$ and functional continuity cannot be accomplished by simple assignment because most node points are not coinciding; thus, we use an interpolation-based scheme to enforce continuity along a nonconforming interface. This scheme is the subject of Section 2.3.1.

2.1.2 Time discretization

While there are many time discretization schemes (e.g.,[5]), we restrict ourselves to semi-implicit multistep methods. In all these methods the diffusion is solved fully implicitly while the time-derivative is approximated using a backward-difference formula (BDF) of order M_{bdf} (see [9, 21]) and the advec-

tion term is approximated by using an explicit extrapolation-based method (Ext) of order M_{ext} [20]. Then, to integrate (8) from time t^{n-1} to time t^n, one has

$$\mathbf{H}_k^n \boldsymbol{u}_k^n = \sum_{m=n-M_{\text{bdf}}}^{n-1} \beta_{\text{bdf}}^{m,n} \mathbf{M}_k^m \boldsymbol{u}_k^m - \sum_{m=n-M_{\text{ext}}}^{n-1} \beta_{\text{ext}}^{m,n} \mathbf{C}_k^m \boldsymbol{u}_k^m, \tag{9}$$

where

$$\mathbf{H}_k^n \equiv \beta_{\text{bdf}}^{n,n} \mathbf{M}_k^n + \nu \mathbf{L}_k^n \tag{10}$$

is a discrete spectral-element Helmholtz operator. Although the matrices $\mathbf{L}_k$ and $\mathbf{M}_k$ in (8) were t-independent, they are time-indexed in (9) because DARe will, in general, reconfigure the partition (A.15) over time. For this reason the coefficients $\beta^{m,n}$ are computed for each t^n as in the traditional schemes cited except that the timestep Δt may vary with m as the smallest spectral-element diameter $h \equiv \min_k h_k$ (A.16) changes. As discussed below, $\vec{u}_{\text{n}}^n$ continuity is maintained during the linear solve of (9) assembled over k for the solution $\boldsymbol{u}^n$. Because the matrix $\mathbf{H}^n \equiv \text{diag}_k \mathbf{H}_k^n$ is symmetric positive-definite (SPD), provided that $\vec{u}_{\text{n}}^n$ is restricted to $\mathbb{U}_{\vec{0}}$, the solution of the assembled (9) is obtained by using a preconditioned conjugate gradient (PCG, see [33, 38]) algorithm to invert $\mathbf{H}^n$ at the time step to t^n. In the presence of nonconforming elements, care must be exercised to ensure that the search directions in the PCG algorithm are in $\mathbb{U}_{\vec{0}}$. We consider appropriate modifications to PCG in Section 2.4.

2.2 Implications for code design

The fully discretized advection-diffusion equation (9) suggests several issues that we have taken into consideration when designing the code. First, all geometric (mesh) information is separated from all other code objects, since

element type information can be encoded easily into the objects that require this distinction. Second, the solution data must be available at multiple time levels, so it is reasonable to provide this information in a data structure. Thus we are led to the notion of *element* and *field* objects. The former contains all geometric information in d dimensions, including the values and tensor-product ordering of the Gauss-quadrature nodes (A.17) and weights (A.22). The element object also contains neighbor-list information and an encoding of the hierarchical element refinement level $\propto \log_{\frac{1}{2}} h_k$ of each element $\mathbb{E}_k$. The field object contains the data $\boldsymbol{u}^m$ representing the physical quantity of interest at all required time levels t^m.

We note that, while we concern ourselves with the advection-diffusion equation (1) here, the code will allow any equation that is discretized by using a Lagrangian basis on up to two meshes (important, for example, when discretizing the Navier-Stokes equations by using a formulation coupling the polynomial spaces $\mathbb{V}_p$-$\mathbb{V}_{p-2}$ [26]). The basis functions and the 1D derivative matrices and Gauss-quadrature nodes (A.10) and weights (A.14) are encapsulated within basis classes (objects), and the SEM operators such as (6,7,10) above are constructed as objects that contain pointers to the basis objects and to a local element object. Generally the d-dimensional SEM operators are not stored but are constructed from a tensor product *application* of the relevant 1D objects. High-level objects encapsulate the solution of (8) or other equations, such as the Navier-Stokes equations, and have common interfaces that allow the equations to take a single time integration step; in other words, all high-level equation solver classes are used in the same way. All the higher-level objects are constructed with "smart" arrays or linked lists of elements and fields that are independent of the objects that solve the equations. Hence, the classes

that handle DARe and enforce continuity between elements being solved. The high-level objects also contain linked lists of SEM operators that *do* depend on the equation being solved.

2.3 Local application of spectral-element operators

In general, global meshes consist of multiple-element grids. In this subsection, we make explicit the form of the SEM operators implemented in the code that ensure the solution continuity across element interfaces.

2.3.1 Continuity between nonconforming elements

In a conforming treatment, $\mathbb{C}^0$ continuity is maintained between elements by ensuring that the function values on the coinciding nodes are the same. The matching condition, then, consists of expressing the N_{g} global (unique) d.o.f. $\boldsymbol{u}_{\mathrm{g}}$ in terms of the local (redundant) d.o.f. as dN_p^d-vectors $\boldsymbol{u}_k$, $k \in \{1, \cdots K\}$. Generally $N_{\mathrm{g}} < KdN_p^d$. This mapping can be accomplished by using a $KdN_p^d \times N_{\mathrm{g}}$ Boolean assembly matrix $\mathbf{A}_{\mathrm{c}}$, such that

$$\boldsymbol{u} = \mathbf{A}_{\mathrm{c}} \boldsymbol{u}_{\mathrm{g}}. \tag{11}$$

For example, if our mesh partition is that in Fig. 1, we can easily see that (11) takes the following explicit form (suppressing zero-valued and $\mu > 1$

blocks):

$$
\boldsymbol{u} = \begin{pmatrix} u_0 \\ \vdots \\ u_{17} \end{pmatrix} = \begin{pmatrix} u_{0,1} \\ \vdots \\ u_{8,1} \\ \hline u_{0,2} \\ \vdots \\ u_{8,2} \end{pmatrix} = \left(\begin{array}{ccc|ccc|ccc|ccc|ccc|ccc}
1&0&0&&&&&&&&&&&&&&&\\
0&1&0&&&&&&&&&&&&&&&\\
0&0&1&&&&&&&&&&&&&&&\\
&&&1&0&0&&&&&&&&&&&&\\
&&&0&1&0&&&&&&&&&&&&\\
&&&0&0&1&&&&&&&&&&&&\\
&&&&&&1&0&0&&&&&&&&&\\
&&&&&&0&1&0&&&&&&&&&\\
&&&&&&0&0&1&&&&&&&&&\\
\hline
0&0&1&&&&&&&0&0&&&&&&&\\
0&0&0&&&&&&&1&0&&&&&&&\\
0&0&0&&&&&&&0&1&&&&&&&\\
&&&0&0&1&&&&&&&0&0&&&&\\
&&&0&0&0&&&&&&&1&0&&&&\\
&&&0&0&0&&&&&&&0&1&&&&\\
&&&&&&0&0&1&&&&&&&0&0&\\
&&&&&&0&0&0&&&&&&&1&0&\\
&&&&&&0&0&0&&&&&&&0&1&
\end{array}\right)
\begin{pmatrix} u_{\mathrm{g},0} \\ \vdots \\ u_{\mathrm{g},14} \end{pmatrix}.
$$

In practice, $\mathbf{A}_{\mathrm{c}}$ is never formed explicitly but is instead *applied*. The matrix $\mathbf{A}_{\mathrm{c}}$, which assigns global d.o.f. to the local d.o.f., is also called a *scatter* matrix. Its transpose $\mathbf{A}_{\mathrm{c}}^{\mathrm{T}}$ performs the associated *gather* operation.

In our nonconforming treatment, we use an interpolation-based method for establishing the interface matching conditions between neighboring (nonconforming) elements. The underlying function space $\mathbb{U}_{\vec{b}}$ (Section 2.1) remains the same, unlike in MEM. Consider the nonconforming mesh configuration depicted in Fig. 2. For the moment denote the global nodes, those nodes residing on the east *parent* edge $\partial\mathbb{E}_{1,1}$, by $\vec{x}_{\mathrm{g},i}$, $i \in \{2, 5, 8\}$, and denote the nodes on the west *child* edges, $\partial\mathbb{D}_{2,3}$ and $\partial\mathbb{D}_{3,3}$ by $\vec{x}_j$, $j \in \{9, 12, 15, 18, 21, 24\}$. We explain elsewhere how in the weak formulation of (1) or other PDEs, the test function factors $\phi_{\mathrm{g},i}(\vec{x})$ arise that are continuous across the global domain $\mathbb{D}$, and interpolate from the global node values $\vec{x}_{\mathrm{g},i}$ [14]. A continuous solution is then found in $\mathrm{span}_i\, \phi_{\mathrm{g},i}$ by projecting there from the space of trial functions $\phi_j(\vec{x})$ that interpolate from the local node values $\vec{x}_j$ but that do not need to be globally continuous. The matrix block $\phi_{\mathrm{g},i}(\vec{x}_j)$ of $\mathbf{A}$ thus generalizes the Boolean scatter matrix used in the conforming-element formulation and accommodates both conforming *and* nonconforming elements. It is convenient to factor $\mathbf{A} = \mathbf{\Phi}\mathbf{A}_{\mathrm{c}}$, where $\mathbf{\Phi}$ is the interpolation matrix from global to local

d.o.f. and $\mathbf{A}_{\mathrm{c}}$ is locally conforming.

For example, the explicit form of (11) for the nonconforming assembly matrix $\mathbf{A} = \mathbf{\Phi}\mathbf{A}_{\mathrm{c}}$ for the mesh illustrated in Fig. 2 is (suppressing zero-valued and $\mu > 1$ blocks)

$$
\boldsymbol{u} = \begin{pmatrix} u_0 \\ \vdots \\ u_{26} \end{pmatrix} = \begin{pmatrix} u_{0,1} \\ \vdots \\ u_{8,1} \\ \hline u_{0,2} \\ \vdots \\ u_{8,2} \\ \hline u_{0,3} \\ \vdots \\ u_{8,3} \end{pmatrix} = \left(\begin{array}{ccccccccccccccccccccccccc}
1 & 0 & 0 & & & & & & & & & & & & & & & & & & & & & & \\
0 & 1 & 0 & & & & & & & & & & & & & & & & & & & & & & \\
0 & 0 & 1 & & & & & & & & & & & & & & & & & & & & & & \\
 & & & 1 & 0 & 0 & & & & & & & & & & & & & & & & & & & \\
 & & & 0 & 1 & 0 & & & & & & & & & & & & & & & & & & & \\
 & & & 0 & 0 & 1 & & & & & & & & & & & & & & & & & & & \\
 & & & & & & 1 & 0 & 0 & & & & & & & & & & & & & & & & \\
 & & & & & & 0 & 1 & 0 & & & & & & & & & & & & & & & & \\
 & & & & & & 0 & 0 & 1 & & & & & & & & & & & & & & & & \\
\hline
0 & 0 & 1 & & & & & & & & & & 0 & 0 & & & & & & & & & & & \\
0 & 0 & 0 & & & & & & & & & & 1 & 0 & & & & & & & & & & & \\
0 & 0 & 0 & & & & & & & & & & 0 & 1 & & & & & & & & & & & \\
0 & 0 & 3/8 & 0 & 0 & 3/4 & 0 & 0 & -1/8 & & & & 0 & 0 & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & & 1 & 0 & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & & 0 & 1 & & & & & & & & & & & \\
 & & & 0 & 0 & 1 & & & & & & & & & 0 & 0 & & & & & & & & & \\
 & & & 0 & 0 & 0 & & & & & & & & & 1 & 0 & & & & & & & & & \\
 & & & 0 & 0 & 0 & & & & & & & & & 0 & 1 & & & & & & & & & \\
\hline
 & & & 0 & 0 & 1 & & & & & & & & & 0 & 0 & & & & & & & & & \\
 & & & 0 & 0 & 0 & & & & & & & & & 1 & 0 & & & & & & & & & \\
 & & & 0 & 0 & 0 & & & & & & & & & 0 & 1 & & & & & & & & & \\
0 & 0 & -1/8 & 0 & 0 & 3/4 & 0 & 0 & 3/8 & & & & & & & 0 & 0 & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & & & & & 1 & 0 & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & & & & & 0 & 1 & & & & & & & & \\
 & & & & & & 0 & 0 & 1 & & & & & & & & & 0 & 0 & & & & & & \\
 & & & & & & 0 & 0 & 0 & & & & & & & & & 1 & 0 & & & & & & \\
 & & & & & & 0 & 0 & 0 & & & & & & & & & 0 & 1 & & & & & &
\end{array}\right)
\begin{pmatrix} u_{\mathrm{g},0} \\ \vdots \\ u_{\mathrm{g},18} \end{pmatrix}
$$

$$
= \left(\begin{array}{ccccccccccccccccccccccccc}
1 & & & & & & & & & & & & & & & & & & & & & & & & \\
 & \ddots & & & & & & & & & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & & & & & & & & & & & \\
 & & & & & & & & & 1 & & & & & & & & & & & & & & & \\
0 & 0 & 3/8 & 0 & 0 & 3/4 & 0 & 0 & -1/8 & & & & & & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & 1 & & & & & & & & & & & & & \\
 & & & & & & & & & & & & \ddots & & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & 1 & & & & & & & & & & & \\
0 & 0 & -1/8 & 0 & 0 & 3/4 & 0 & 0 & 3/8 & & & & & & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & & & & 1 & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & & & & & \ddots & & & & & & & & & \\
 & & & & & & & & & & & & & & & & 1 & & & & & & & &
\end{array}\right)
\left(\begin{array}{ccccccccccccccccccccccccc}
1 & & & & & & & & & & & & & & & & & & & & & & & & \\
 & \ddots & & & & & & & & & & & & & & & & & & & & & & & \\
 & & & 1 & & & & & & & & & & & & & & & & & & & & & \\
\hline
0 & 0 & 1 & & & & & & & & & & 0 & 0 & & & & & & & & & & & \\
0 & 0 & 0 & & & & & & & & & & 1 & 0 & & & & & & & & & & & \\
0 & 0 & 0 & & & & & & & & & & 0 & 1 & & & & & & & & & & & \\
 & & & & & & & & & 1 & & & & & & & & & & & & & & & \\
 & & & & & & & & & & 1 & & & & & & & & & & & & & & \\
 & & & 0 & 0 & 1 & & & & & & & & & 0 & 0 & & & & & & & & & \\
 & & & 0 & 0 & 0 & & & & & & & & & 1 & 0 & & & & & & & & & \\
 & & & 0 & 0 & 0 & & & & & & & & & 0 & 1 & & & & & & & & & \\
\hline
 & & & 0 & 0 & 1 & & & & & & & & & 0 & 0 & & & & & & & & & \\
 & & & 0 & 0 & 0 & & & & & & & & & 1 & 0 & & & & & & & & & \\
 & & & 0 & 0 & 0 & & & & & & & & & 0 & 1 & & & & & & & & & \\
 & & & & & & & & & & & & & 1 & & & & & & & & & & & \\
 & & & & & & & & & & & & & & 1 & & & & & & & & & & \\
 & & & & & & 0 & 0 & 1 & & & & & & & & & 0 & 0 & & & & & & \\
 & & & & & & 0 & 0 & 0 & & & & & & & & & 1 & 0 & & & & & & \\
 & & & & & & 0 & 0 & 0 & & & & & & & & & 0 & 1 & & & & & &
\end{array}\right)
\begin{pmatrix} u_{\mathrm{g},0} \\ \vdots \\ u_{\mathrm{g},18} \end{pmatrix}.
$$

Note that the $\mathbf{A}$ entries corresponding to the child-node rows (see Fig. 2) are not Boolean but that every row sum is unity. This result is to be expected because $\mathbf{A}$ must accommodate interpolation of a constant solution (e.g., $u_{\mathrm{g},i} = 1\ \forall i$) across a nonconforming interface.

2.3.2 Global assembly

To accommodate Dirichlet boundary conditions (2) into the solution, we employ a *masking projection* $\mathbf{\Pi}$, which is locally diagonal with unit entries everywhere except corresponding to nodes on Dirichlet boundaries, where there are zero entries. Any field $\vec{\boldsymbol{\phi}}^{\mathrm{T}}\boldsymbol{u} = \vec{u} \in \mathbb{U}_{\vec{b}}$ may be analyzed as $\vec{u} = \vec{u}_{\mathrm{h}} + \vec{u}_{\mathrm{b}}$, where $\boldsymbol{u}_{\mathrm{h}} \equiv \mathbf{\Phi\Pi A}_{\mathrm{c}}\boldsymbol{u}_{\mathrm{g}}$ constructs the projection $\vec{u}_{\mathrm{h}} \equiv \vec{\boldsymbol{\phi}}^{\mathrm{T}}\boldsymbol{u}_{\mathrm{h}} \in \mathbb{U}_{\vec{0}}$ of $\vec{u}$, that is, its homogeneous part, and $\boldsymbol{u}_{\mathrm{b}} \equiv \boldsymbol{u} - \boldsymbol{u}_{\mathrm{h}}$ constructs $\vec{u}_{\mathrm{b}} \in \mathbb{U}_{\vec{0}}$, which vanishes at the interior nodes $\vec{x}_{\vec{\jmath},k} \notin \partial\mathbb{D}$ of $\mathbb{D}$. Inserting this analysis into (5) (noting $\vec{v} \in \mathbb{U}_{\vec{0}} \Rightarrow \boldsymbol{v} = \mathbf{\Phi\Pi A}_{\mathrm{c}}\boldsymbol{v}_{\mathrm{g}}$) and repeating the time discretization leading to (9), we arrive at the following linear equation to solve for $\boldsymbol{u}_{\mathrm{h}}$ at each time step:

$$\boldsymbol{v}^{\mathrm{T}}\mathbf{H}\boldsymbol{u} = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{f} \quad \forall \boldsymbol{v}_{\mathrm{g}} \Longrightarrow \mathbf{A}_{\mathrm{c}}^{\mathrm{T}}\mathbf{\Pi\Phi}^{\mathrm{T}}\mathbf{H\Phi\Pi A}_{\mathrm{c}}\boldsymbol{u}_{\mathrm{g}} = \mathbf{A}_{\mathrm{c}}^{\mathrm{T}}\mathbf{\Pi\Phi}^{\mathrm{T}}(\boldsymbol{f} - \mathbf{H}\boldsymbol{u}_{\mathrm{b}}), \tag{12}$$

where we have ascribed all explicit past-time references from the time-derivative expansion and the advection terms in (9) to $\boldsymbol{f}$. Equation (12) is solved by using the PCG algorithm. While (12) shows explicitly that the matrix on the left is symmetric nonnegative-definite, it is not in a form easily solved on a parallel system. Left-multiplying (12) by $\mathbf{\Phi\Pi A}_{\mathrm{c}}$ and letting

$$\mathbf{\Sigma} \equiv \mathbf{\Phi\Pi A}_{\mathrm{c}}\mathbf{A}_{\mathrm{c}}^{\mathrm{T}}\mathbf{\Pi\Phi}^{\mathrm{T}}, \tag{13}$$

we get the following local form:

$$\mathbf{\Sigma H}\boldsymbol{u}_{\mathrm{h}} = \mathbf{\Sigma}(\boldsymbol{f} - \mathbf{H}\boldsymbol{u}_{\mathrm{b}}). \tag{14}$$

The *direct stiffness summation* (DSS) matrix $\mathbf{\Sigma}$ is coded so that the gather and scatter are performed in one operation (see Section 2.5.2), which reduces communication overhead on parallel systems [36].

In addition to $\mathbf{\Pi}$ we must introduce the inverse *multiplicity matrix* $\mathbf{W}$ to

maintain $\mathbb{H}^1(\mathbb{D})$ continuity (see Section 2.4). This matrix is diagonal, computed by initializing a collocated vector $g^{\mu}_{\vec{\jmath},k} = 1\ \forall \vec{\jmath}, k, \mu$, setting child face nodes to 0, performing $\boldsymbol{g} \leftarrow \boldsymbol{\Phi}\mathbf{A}_{\mathrm{c}}\mathbf{A}_{\mathrm{c}}^{\mathrm{T}}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{g}$, then setting

$$W^{\mu,\mu'}_{\vec{\jmath},k,\vec{\jmath}',k'} = \begin{cases} \delta^{\mu,\mu'}/g^{\mu}_{\vec{\jmath},k}, & \text{if } \vec{x}_{\vec{\jmath},k} = \vec{x}_{\vec{\jmath}',k'} \text{ coincides with a global node,} \\ 0, & \text{otherwise.} \end{cases}$$

For example, corresponding to Figs. 1 and 2 the diagonals of $\mathbf{W}$ are

$$\begin{aligned} & (1, 1, \tfrac{1}{2}, 1, 1, \tfrac{1}{2}, 1, 1, \tfrac{1}{2}, \tfrac{1}{2}, 1, 1, \tfrac{1}{2}, 1, 1, \tfrac{1}{2}, 1, 1) \\ \text{and} \quad & (1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, \tfrac{1}{2}, \tfrac{1}{2}, 0, \tfrac{1}{2}, \tfrac{1}{2}, 0, 1, 1, 0, 1, 1), \end{aligned}$$

respectively.

2.4 Linear solver

The modifications to the PCG algorithm required to solve (14) in the non-conforming case stem from the requirement that the iteration residuals $\boldsymbol{r}$ and the search directions $\boldsymbol{w}$ correspond to functions $\vec{r} \equiv \vec{\boldsymbol{\phi}}^{\mathrm{T}}\boldsymbol{r}$ and $\vec{w} \equiv \vec{\boldsymbol{\phi}}^{\mathrm{T}}\boldsymbol{w}$ belonging to $\mathbb{H}^1(\mathbb{D})^d$. The CG algorithm searches the global d.o.f. space for the solution to the linear equation. So that we may continue to use the local matrix forms, however, we must also mask off all Dirichlet nodes (if any exist), which are not solved for. The assembly matrix masks off these nodes in such a way that the new search direction $\vec{w} \in \mathbb{H}^1(\mathbb{D})^d$. Additionally, in all cases in the CG iteration where a quantity $\vec{g}$ must remain in $\mathbb{H}^1(\mathbb{D})^d$, we explicitly "smooth" it by $\boldsymbol{g} \leftarrow \mathbf{S}\boldsymbol{g}$, using an $\mathbb{H}^1$ smoothing matrix $\mathbf{S} \equiv \boldsymbol{\Phi}\boldsymbol{\Pi}\mathbf{A}_{\mathrm{c}}\mathbf{A}_{\mathrm{c}}^{\mathrm{T}}\mathbf{W}$. The result of the operation is a quantity that is interpolated properly to the child edges and that is expressed precisely once (unlike in the DSS case) at multiple local nodes that represent the same spatial location. A similar smoothing matrix

```
u_h = 0                          // initialize homogeneous term
r = Σ(f − HSu_b)                 // initialize residual
w = 0                            // initialize search vector
ρ_1 = 1                          // initialize parameter
Loop until convergence:
    e = SP^{-1} r                // error estimate
    ρ_0 = ρ_1, ρ_1 = r^T W e     // update parameters
    w ← e + w ρ_1/ρ_0            // increment search vector
    r' = ΣHw                     // image of w
    α = ρ_1/w^T W r'             // component of u_h increment
    u_h ← u_h + αw               // increment u_h along w
    r ← r − αr'                  // increment residual
End Loop
u = S_f(u_h + u_b).
```

Fig. 3. PCG algorithm

$\mathbf{S}_\mathrm{f} \equiv \mathbf{\Phi}\mathbf{A}_\mathrm{c}\mathbf{A}_\mathrm{c}^\mathrm{T}\mathbf{W}$, where the Dirichlet mask is not inserted, is used to smooth the final inhomogeneous solution. Note that it is critical that the inhomogeneous boundary term $\vec{u}_\mathrm{b} \in \mathbb{H}^1(\mathbb{D})^d$ in (14); thus, the smoothing matrix $\mathbf{S}$ is applied to $\boldsymbol{u}_\mathrm{b}$ before the Helmholtz operator is. However, the nonsmoothed boundary term must be added after the convergence loop in order to complete the solution. Note also that the final smoothing operation follows the addition of the boundary condition and therefore *cannot* be masked; hence the distinction of the final $\mathbf{S}_\mathrm{f}$ matrix.

2.4.1 Modified preconditioned conjugate-gradient algorithm

With these considerations we present in Table 3 the PCG algorithm for the assembled local problem (14) for conforming elements [see 9, Sec. 4.5.4] modified for nonconforming elements. Preconditioning is handled by the matrix $\mathbf{P}^{-1}$. In general, the preconditioned quantity must be smoothed.

2.4.2 Preconditioners

Much investigation remains to determine the optimal preconditioners $\mathbf{P}^{-1}$ for the equation solvers supported in GASpAR. However, the preconditioning strategy is not the present work's focus. Nevertheless, for the purpose of turbulence study we, make some general comments to guide the development of optimal preconditioners. If we consider the expression (10) for the spectral-element Helmholtz operator, we note that $\nu \propto \mathsf{Re}^{-1}$ is very small. Also, because we always use relatively high spatial degree, the timestep $\Delta t \propto 1/\beta_{\mathrm{bdf}}^{n,n}$ is very small. Hence, we see that $\mathbf{H}_k^n$ is strongly diagonally dominant. Thus, approximating $\mathbf{H}_k^n$ with its diagonal entries alone is a reasonably good choice for a preconditioner, and this is used in the present work.

2.5 Adaptive mesh formulation

As mentioned in Section 2.1.1, the global domain $\mathbb{D}$ is initially covered (A.15) by a set of disjoint (nonoverlapping) elements $\mathbb{E}_k$. Each of these initial elements becomes a tree root element, which is identified by a unique root *key* $k_{\mathrm{r}} \in \{1, 2, 3, \cdots\}$ for that tree. At each level $\ell \in \{\ell_{\min}, \cdots \ell_{\max}\}$, an element data structure provides both its own key k and its root key k_{r}. For any level ℓ, the range of $2^{d\ell}$ valid element keys will be $k \in [2^{d\ell}k_{\mathrm{r}}, 2^{d\ell}(k_{\mathrm{r}}+1)-1]$ because the refinement is *isotropic* (that is, it splits an element at the midpoints of all its edges to produce its 2^d child elements). Conversely, we obtain the level index from the element key using

$$\ell = \lfloor \log_{2^d}(k/k_{\mathrm{r}}) \rfloor. \tag{15}$$

In order to ensure all keys are unique, given a root k_{r} the next root is $k'_{\mathrm{r}} = 2^{d\ell_{\max}}(k_{\mathrm{r}}+1)$, and so on.

After elements $\mathbb{E}_k$ are identified ("tagged") for refinement or coarsening at level ℓ, three steps are involved in performing DARe: (1) performing refinement by adding a new level of 2^d child elements $\mathbb{E}_{2^d k}, \cdots \mathbb{E}_{2^d(k+1)-1}$ at level $\ell+1$ to replace each $\mathbb{E}_k$, or else coarsening 2^d existing children $\mathbb{E}_k, \cdots \mathbb{E}_{k+2^d-1}$ into a new parent $\mathbb{E}_{\lfloor k/2^d \rfloor}$; (2) building data structures for all element boundaries, which hold data representing global d.o.f. and accept gathers ($\mathbf{A}^{\mathrm{T}}\boldsymbol{u}$ values) or perform scatters ($\mathbf{A}\boldsymbol{u}_{\mathrm{g}}$ values); and (3) determining neighbor lists for data exchange. Neighbor lists consist of records (structures) that each contain the computer processor id, element key k, root key k_{r} and boundary id $s \in \left\{0, \cdots 2^d - 1\right\}$ of each neighbor element that adjoins every interface. In refining or coarsening, the field values for each child (parent) elements are interpolated from the parent (child) fields. For simplicity, the interior of each element boundary is restricted to an interface between one coarse and at most 2^{d-1} refined neighbors. Thus, at most one refinement-level difference will exist across the interior of an interface between neighboring elements.

In GASpAR, the data structures that represent global d.o.f. at the inter-element interfaces are referred to as "mortars." These structures are not to be confused with the mortars used in MEM; however, they serve as templates for that more general method. Recalling Fig. 2 as a paradigm, in general the mortars contain node locations and the basis set of the parent edge (in 2D, or face in 3D). These structures represent the same field information for the parent and child edges; their nodes coincide with the parent edge's nodes, and they interpolate global d.o.f. data to the child edges, as described above. The mortar data structures are determined by communicating with all neighbors

to determine which interfaces are nonconforming. This communication uses a *voxel database* (VDB) [16]. A voxel database consists of records containing geometric point locations, a component id that tells what part of the element $\mathbb{E}_k$ (edge $\partial\mathbb{E}_{k,s}$, vertex $\in \partial^2\mathbb{E}_{k,s}$, etc.) the point represents, an id of the element that contains the point, that element's root id, and some auxiliary data. Two VDBs are constructed: one consists of all element vertices, and one consists of all element edge midpoints. With these two VDBs, we are able to determine whether a relationship between neighbor edges is conforming and also determine the mortar's geometrical extent. The VDB approach can also be used for general deformed geometries in two and three dimensions, as long as adjacent elements share well-defined common node points.

The algorithm classes that carry out refinement operate only on the element and field lists. The SEM solvers adjust themselves automatically to accommodate the lists that are modified as a result of DARe.

2.5.1 Refinement and coarsening rules

The method that actually carries out the refinement and coarsening takes as arguments only two buffers: one containing the local indexes of the elements to be refined, and one with indexes of elements to be coarsened. While the refinement criteria that identify elements for refinement or coarsening are described in Section 2.5.3, we point out here that before mesh refinement or coarsening is undertaken, the tagged elements are checked for compliance with several rules. For refinement, we have the following rules:

r1. The refinement level must not exceed a specified value.

r2. No more than one refinement level may separate neighbor elements.

These rules must be adhered to even in the case of interfaces that lie on periodic boundaries. The refinement list is modified to remove any elements that would violate rule r1, and rule r2 is enforced by tagging a coarse element for refinement if it has a refined neighbor that is in the current refinement list. This is most easily effected by building a global refinement list, consisting of the element keys of all elements tagged for refinement. Each local element's neighbors are then checked; if a neighbor is in the global refinement list and it is a nonconforming neighbor, then the local element must also be refined.

We may not coarsen an element if under any of the following conditions:

c1. The element is the tree root.

c2. Any of its $2^d - 1$ siblings are not tagged for coarsening.

c3. The element appears in a refinement list.

c4. Refinement rule r2 would be violated.

To enforce rule c4, we introduce the notion of a *query-list*, in other words, a global list of records of each element key k, its parent key $\lfloor k/2^d \rfloor$, and its refinement level ℓ (15). The global list is built from local buffers of element indexes that are then gathered from among all processors. The following procedure is then used:

1. Build a query-list (RQL) from the element indexes in the refinement list.
2. Find the maximum and minimum refinement levels $\ell_{\max}$ and $\ell_{\min}$ represented among all keys tagged for coarsening.
3. Reorder the current local coarsen list from $\ell_{\max}$ down to $\ell_{\min}$.
4. Working from $\ell = \ell_{\max}$ down to $\ell_{\min}$,
    i. Build a query-list (CQL) from the element indexes in the current coarsen list.

ii. For all elements in the local coarsen list at the current ℓ, check that (a) any refined neighbor is in the CQL and (b) no refined neighbors are in the the RQL. If both conditions are met at this ℓ, then the local element index is retained in the current coarsen list. Otherwise it is deleted.

5. Do a final check that all elements in the local coarsen list have all their siblings also tagged for coarsening. This is done by building a query-list of all current coarsen lists and verifying that a local element's siblings are in the global list. Sibling elements are identified by having the same parent key as the local element.

We note that in order to make the algorithm consistent, the local refinement lists are checked, and possibly modified *before* checking and modifying the coarsen lists.

2.5.2 Communicating boundary data

The mortar data structures contain all the data to be communicated between elements during each application of the DSS or smoothing operation (13). GASpAR communicates element-boundary data by exchanging data between adjacent elements, which necessitates network communication on parallel computers. This data exchange involves an *initialization* step and and *operation* step. The initialization step establishes the required element/processor connectivity by performing a bin-sort of global node indexes and having each processor process the nodes from a given bin to determine neighbor lists. This method was suggested in [9, ch. 8] but to our knowledge has never been implemented. The procedure is to label the mortar-structure nodes with unique

indexes, generated by computing the Morton-ordered index for each geometric node point. This index is computed by integralizing the physical-space coordinates and then interleaving the bits of each integer component to create a unique integer. Thus, all nodes representing the same geometric position will have the same index label. For P processors, a collection of bins $\mathbb{B}_l$, $l \in \{0, \cdots P-1\}$, is generated that partitions the dynamic range (global maximum) of the node indexes. Each processor l partitions its list of node indexes into the bins, sending the contents of bin $\mathbb{B}_{l'}$ to processor l', where the data are combined with those from other processors and then sent back to the originating processor. After this step, a given processor is informed of which other processors share which nodes among all the mortar nodes.

With the information gleaned from this initialization step, the operation step involves the communication of the data at any node point with all other processors that share that node. This data is extracted from the element in question by using the pointer indirection provided by the local-to-global map represented by the unique node index. The data at common vertices is summed for the DSS or smoothing operations and returned to the local node also by indirection. The algorithm provides that common vertices residing on the same processor are summed before being transmitted to the other processors that share the node, in order to reduce the amount of data being communicated. At the end of the operation step, the data at multiply-represented global nodes are identical. This gather-scatter procedure ensures that the DSS'ed data are available for local computation immediately after the final communication of data.

One benefit of this method for performing the gather-scatter operations is that it allows the communication to separated from the geometry because the

global unique node ids are essentially unstructured lists of local data locations. However, the method is somewhat inefficient in the current formulation of DARe in GASpAR. Neighbor lists for transmitting data may also be constructed by using the VDB. Since the VDBs are synchronized and thus represent, in a sense, global data, this same VDB can also be used to determine a given element edge's neighbor lists. After the mortars are constructed, each element edge is associated with its neighbors' local ids and processor ids. Hence, the neighbor lists for handling element-boundary data exchange are determined easily from existing global data; there is no need to perform the initialization step in the gather-scatter method described above because the information is readily available from the VDB synchronization.

2.5.3 Error estimators

Elements are tagged for refinement or coarsening by using an a posteriori refinement criterion. One criterion, adapted from [27], uses the Legendre spectral information contained within the spectral-element representation to estimate the quadrature and truncation errors in the solution and to compute the rate at which the solution converges spectrally in each element $\bar{\mathbb{E}}_k$. We refer to this as the *spectral estimator*. The discretized solution along 1D lines in coordinate direction μ' (fixing the other coordinates) is represented as a Legendre expansion

$$u^{\mu}(\vec{\vartheta}_k(\cdots,\xi^{\mu'}\cdots)) = \sum_{j=0}^{p} \check{u}_j^{\mu} \mathrm{L}_j(\xi^{\mu'}), \tag{16}$$

where the $\check{u}_j^{\mu}$ can be computed easily by using the orthogonality of L_j w.r.t. (A.23). Then an approximation to the solution error along the line can be

expressed as

$$\varepsilon^{\mu}_{\text{est}} = \sqrt{\sum_{j=p}^{\infty} \left(j + \tfrac{1}{2}\right)^{-1} (\check{u}^{\mu}_{j})^2}, \tag{17}$$

where the $j = p$ term is the (over)estimate of the quadrature error and the $j > p$ terms sum to the truncation error. Since we do not have $\check{u}^{\mu}_{j>p}$, we must extrapolate using the coefficients we do have. We assume $\check{u}^{\mu}_{j>p}$ can be approximated by $\ln|\check{u}^{\mu}_{j}| \approx \ln C^{\mu} - \lambda^{\mu} j$, and we determine C^{μ} and λ^{μ} from a linear fit using the M final coefficients $\check{u}^{\mu}_{p-M<j\leq p}$ in (16). With this continuous approximation for the extrapolated coefficients, we approximate the truncation error term in (17) as an integral over j and integrate analytically to compute the total error $\varepsilon^{\mu}_{\text{est}}$. In this work, the maximum error over all N_p lines for each μ' within $\bar{\mathbb{E}}_k$ is computed and taken as the $\bar{\mathbb{E}}_k$-local error estimate. The local convergence rate is simply the minimum value of $|\lambda^{\mu}|$ over all lines and all μ'. For all our test cases, $M = 5$.

Thus, $\bar{\mathbb{E}}_k$ is refined, if for some μ, $\varepsilon^{\mu}_{\text{est}}$ is above a threshold value or if $|\lambda^{\mu}|$ is below another threshold. For coarsening, for all μ, all 2^d sibling elements must have their $\varepsilon^{\mu}_{\text{est}}$s below some value proportional to the refinement threshold. This requirement prevents "blinking," where refined elements are immediately coarsened because the error tolerances are met on the refined elements.

In conjunction with this spectral criterion, we can often obtain better overall accuracy-convergence results by checking whether the $\bar{\mathbb{E}}_k$-maximum second derivative in any coordinate is above a certain threshold and by performing a logical OR of that condition with the spectral error threshold for refinement tagging.

While the high degrees used in the expansions will help the spectral error estimator, other refinement criteria may be more effective, given the variety of

solution structures arising in our applications. The investigation of refinement criteria appropriate for such intermittent features is a major outstanding problem in numerical solution of PDEs, and one that we consider in a companion paper [14].

3 Test problems and results

We have chosen a set of test problems that examine various aspects of (1). The primary objective is to investigate the temporal and spatial convergence properties of the solutions when adaptivity is used. For this purpose, we have selected test problems that have analytic solutions, so that the errors may be determined exactly, instead of only by comparison, for example, to a highly refined control solution. The test problems begin with the simplest aspect of (1) and continue to progressively more difficult problems until the behavior of the full 2D nonlinear version of (1) is considered.

For each test problem, a BDF3/Ext3 scheme is used for the time derivative and the advection term, respectively (Section 2.1.2). This requires that, at the initial time t^0, all the required time levels t^m be initialized, $m \in \{1, \cdots \max(M_{\text{bdf}}, M_{\text{ext}}) - 1\}$. Both the spectral and second-derivative error estimators are used for adaption criteria. The spectral error is normalized by the norm of the solution, $||\boldsymbol{u}^0||_p \equiv \sqrt{\boldsymbol{u}^{0^{\text{T}}}\boldsymbol{u}^0}$, at the start of the run, and the second x^μ-derivative is normalized by $||\boldsymbol{u}^0||_p/L^2$, where L is the longest global domain length. Elements are tagged for refinement based on a logical OR of the two criteria. The $|\lambda^\mu|$ threshold in all cases is $\ln 10$.

For the linear case $\vec{c} = \vec{c}(t)$ the analytic fundamental solution of (1) is a d-periodized Gaussian in $\mathbb{D} = [0,1]^d$:

$$u_{\mathrm{a}}^{\mu}(\vec{x},t) \equiv \frac{\sigma(0)^d}{\sigma(t)^d} \sum_{\imath^1,\cdots\imath^d=-\infty}^{\infty} \exp - \left(\frac{\vec{x} - \vec{x}^0 + \vec{\imath} - \int_0^t \vec{c}(t')\,\mathrm{d}t'}{\sigma(t)} \right)^2 \tag{18}$$

for $t > -\sigma(0)^2/4\nu$ ($u_{\mathrm{a}}^{\mu}(\vec{x},t) \equiv 0$ otherwise), where $\sigma(t) \equiv \sqrt{\sigma(0)^2 + 4\nu t}$, $\sigma(0) = \sqrt{2}/20$ is the initial distribution e-folding width and $\vec{x}^0 = \sum_{\mu=1}^{d} \vec{\imath}^{\mu}/2$ is the initial peak location. To compute (18), we truncate summands of value be less than 10^{-18} of the partial sum. The simplest version of (1) is the heat equation, where $\vec{c} = \vec{0}$. The purpose here is to investigate temporal and spatial convergence of the adaptive solutions without advection. The initial condition at $t = 0$ from (18) with $d = 2$ is computed on $K = 4 \times 4$ elements, and the mesh is refined until the maximum allowed number of refinement levels may be reached. Both a spectral estimator and derivative threshold were used. The threshold and coarsening factor for each were set to be 10^{-4} (10^{-2}) and 1 (0.5), respectively.

3.1.1 Temporal convergence

First, we consider time convergence of the adaptive solutions by integrating to a fixed time $t = 0.05$ for various fixed timesteps Δt. From (18) a curve of relative $\mathbb{L}_2$ error $\varepsilon = ||\boldsymbol{u} - \boldsymbol{u}_{\mathrm{a}}||_p / ||\boldsymbol{u}_{\mathrm{a}}^0||_p$ vs Δt is generated for each of several maximum-refinement levels $\ell_{\max}$ and for four degrees p. We present the results in Fig. 4, adopting the convention ℓ-*control* for the grid that uniformly covers the domain with elements at the finest resolution in the $\ell_{\max} = \ell$ case. The BDF3/Ext3 is a globally third-order scheme, so we expect that if the solution

is well resolved spatially, we should see a slope of ≈ 3 in a log-log plot of error vs Δt. This is precisely what is seen in the figure; each plot consists of a sequence of four curves for the refinement levels $\ell_{\max} \in \{0, \cdots 3\}$, where $\ell_{\max} = 0$ implies that no refinement is done. For the spatially resolved curves in each plot, the error is linear with slope 3.14.

We see in Fig. 4 that even at low p, the solution can be well resolved as long as refinement is used. We also see that as p increases, there is less need for refinement because the unrefined mesh is able to resolve the solution adequately, at least for the period of integration.

To compare the various refinement levels, we have made the same runs using a 3-control grid. The $\ell_{\max}$-control solutions are defined to be those generated on a nonadaptive grid comprising elements at the finest scale in the $\ell_{\max}$-adaptive case. Thus, in this problem, since we initialized with a $K = 4 \times 4$ grid, the 3-control grid consists of $2^3 4 \times 2^3 4$ or $K = 32 \times 32$ elements. These plots are indicated by the dashed lines in Fig. 4, which all follow the $\ell_{\max} = 3$ curves. As the polynomial degree increases, the solution requires fewer levels of refinement to achieve the same accuracy as with the 3-control grid, at a considerable computational savings.

3.1.2 Spatial convergence

In this portion of the heat-equation test, we consider the effects of polynomial degree p on the solution. The maximum number of refinement levels is fixed to $\ell_{\max} = 3$. Here, a variable Courant-limited timestep

$$\Delta t \leq \kappa \Big/ \max_{j \in \{1,\cdots p\}, k \in \{1,\cdots K\}, \mu \in \{1,\cdots d\}} \left(\frac{4\nu}{(h^{\mu}_{j,k})^2} + \frac{|u^{\mu}_{j-1,k}| + |u^{\mu}_{j,k}|}{2h^{\mu}_{j,k}} \right)$$

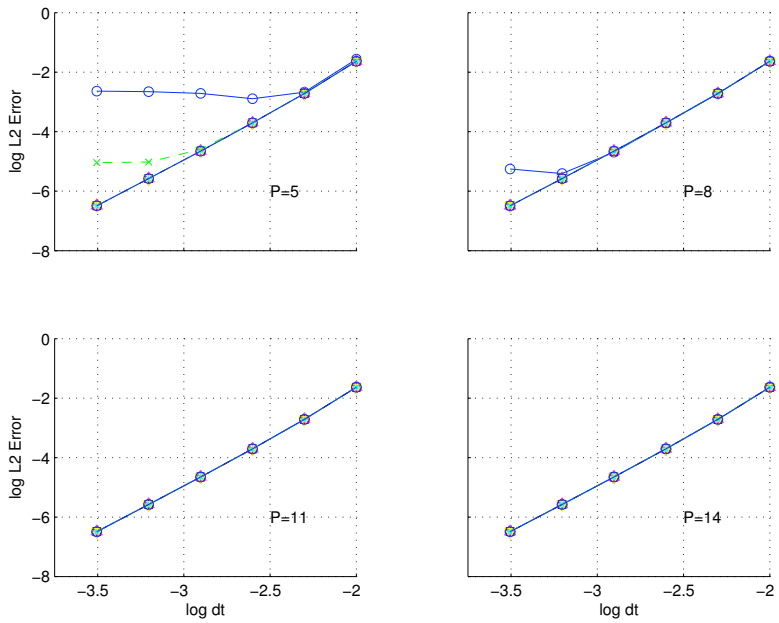

Fig. 4. Plots of $\log_{10}(||\boldsymbol{u}-\boldsymbol{u}_{\rm a}||_p/||\boldsymbol{u}_{\rm a}^0||_p)$ for the heat equation vs $\log_{10}\Delta t$, for different polynomial degrees p. Within each plot are four curves for the maximum refinement levels $\ell_{\max}=0$ (solid curve with circle markers), $\ell_{\max}=1$ (dashed curve with cross markers), $\ell_{\max}=2$ (dotted curve with diamond markers), and $\ell_{\max}=3$ (dash-dotted curve with square markers). The control solutions are indicated with dashed curves and follow closely the $\ell_{\max}=3$ curves. The axes in each plot have the same limits. Note that as p increases, the curves converge, for this range of Δt.

is used with a fixed Courant number of $\kappa=0.2$, where $h_{j,k}^{\mu}\equiv|x_{j,k}^{\mu}-x_{j-1,k}^{\mu}|$ uses (A.17). The solution is integrated to a time $t=0.5$ chosen so that we observe the solution coarsening as it decays. The initial mesh is the same as in the time convergence test. The spatial convergence result is presented in Fig. 5.

We see in the figure the linear behavior characteristic of the spatial convergence in all our test problems. We expect that a solution converging spectrally as in (A.7) will exhibit a straight line in a plot of $\log_{10}(||\boldsymbol{u}-\boldsymbol{u}_{\rm a}||_p/||\boldsymbol{u}_{\rm a}^0||_p)$ vs p, indicating that the error decays exponentially. Also plotted in this figure

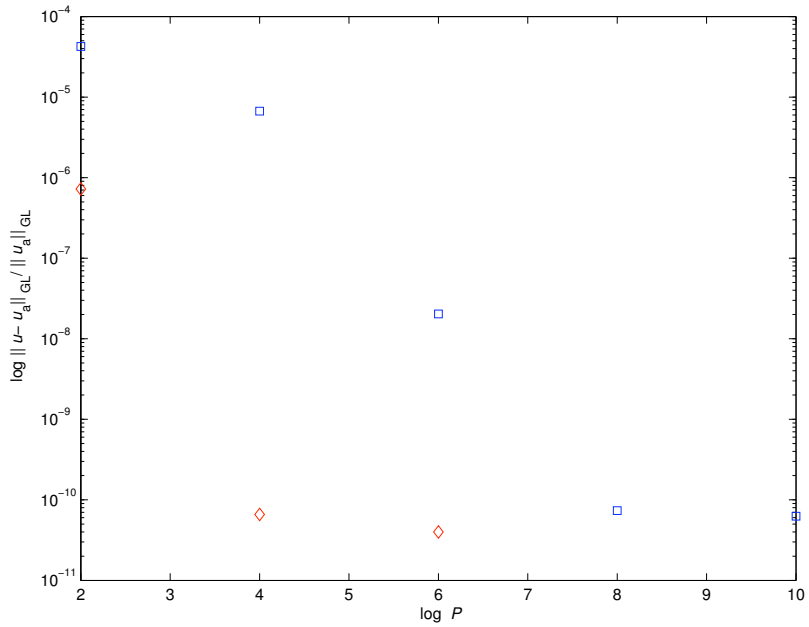

Fig. 5. Plots of $\log_{10}(||\boldsymbol{u} - \boldsymbol{u}_{\mathrm{a}}||_p/||\boldsymbol{u}_{\mathrm{a}}^0||_p)$, as a function of p for the heat equation test. The square markers indicate the adaptive runs, while the diamonds represent the 3-control grid runs at the same polynomial degree.

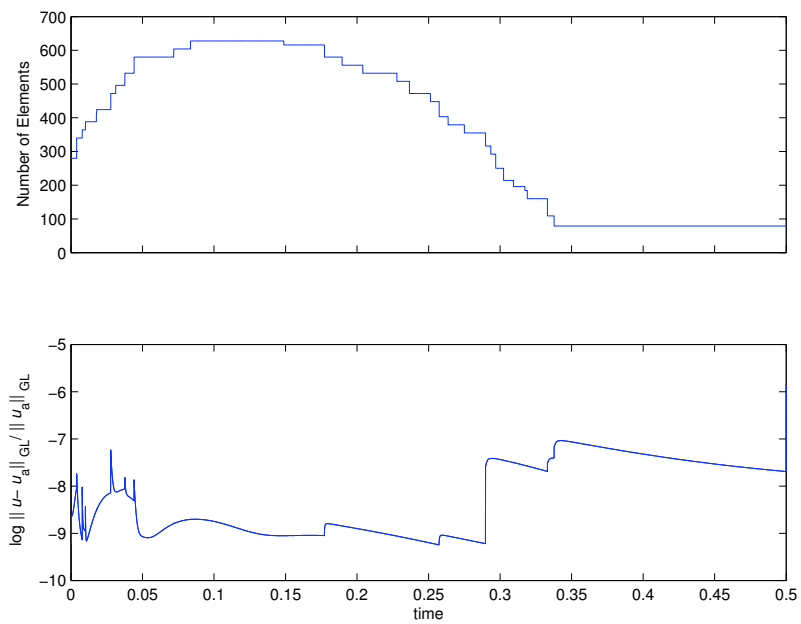

Fig. 6. Plots of diagnostic quantities vs time in the $p = 6$ heat equation test. *Top*: *Number of Elements* vs *time*. *Bottom*: $\log_{10}(||\boldsymbol{u} - \boldsymbol{u}_{\mathrm{a}}||_p/||\boldsymbol{u}_{\mathrm{a}}^0||_p)$ vs *time*.

are the 3-control runs, which saturate very quickly in this case but still provide uniformly better errors than in the adaptive case except at the highest degrees. The cause is likely the elliptic nature of the problem in which the coarsening elements transmit their error throughout the grid. In Fig. 6 we can see that the error over time does not decay monotonically except during periods where the grid is quasi-static. Keeping in mind that the decaying Gaussian contains all polynomial orders, one concludes that the solution error is globally influenced by the error from interpolations and from the inability of the truncated polynomial expansion at this degree to accurately model the decaying Gaussian.

3.2 Linear advection equation

In our next test we consider the linear advection equation (1) with $d = 2$ and constant $\vec{c} = \vec{\iota}^1$. This problem considers the ability of the code to simulate and follow a reasonably sharply localized translating distribution. The initial distribution is given by (18) at $t = 0$. For this test problem, we set $\nu = 10^{-4}$. The spectral error tolerance in this problem is turned off. The derivative criterion is set to 1.0 with a coarsening factor of 0.5. The $|\lambda^\mu|$ threshold is $\ln 10$.

3.2.1 Temporal convergence

The temporal convergence test is done in the same way as for the heat equation (Section 3.1.1). The total integration time is $t = 0.06$. We begin with a $K = 4 \times 4$ element mesh, each element of 2D degree $p \times p$. In Fig. 7 we present our results.

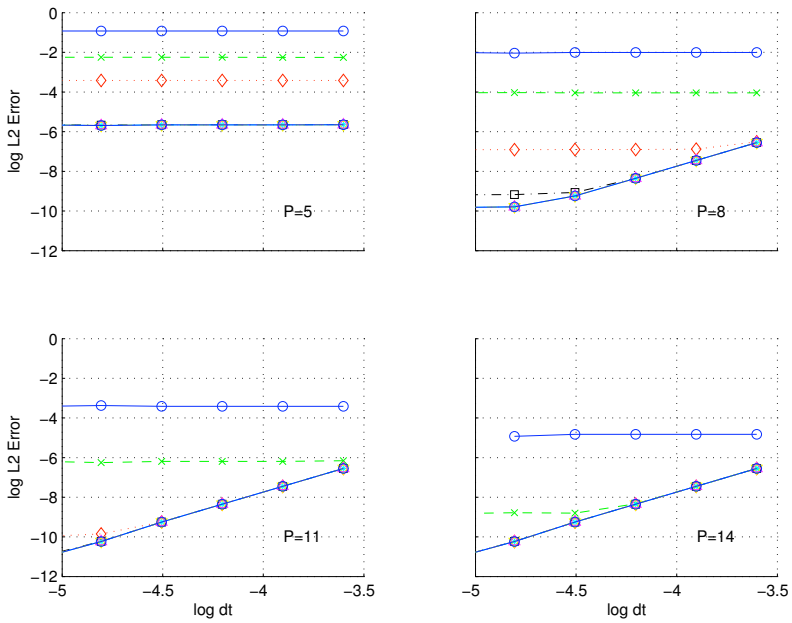

Fig. 7. Plots of $\log_{10}(||\boldsymbol{u} - \boldsymbol{u}_{\mathrm{a}}||_p / ||\boldsymbol{u}_{\mathrm{a}}^0||_p)$ for the advection-dominated problem vs $\log_{10} \Delta t$, for different p. Within each plot are subplots for each of four refinement levels, $\ell_{\max} = 0$ (solid curve with circle markers), $\ell_{\max} = 1$ (dashed curve with $\times$ markers), $\ell_{\max} = 2$ (dotted curve with diamond markers), and $\ell_{\max} = 3$ (dash-dotted curve with square markers). The control solutions are indicated with dashed curves and generally follow the $\ell_{\max} = 3$ curves. The axes in each plot have the same limits.

For the spatially resolved curves in each plot, the slope of the curve is 2.95. Even at high degree p, the error is Δt-independent for the unrefined mesh. For lower degrees, to obtain a case where the solution error decays at the order of the time-stepping method indeed requires a larger number of refinement levels, indicating that the solution is well resolved spatially only at these higher levels. Thus, in order to resolve this distribution properly so that the temporal error is $\mathcal{O}(\Delta t^3)$, refinement is necessary.

Also plotted Fig. 7 are the 3-control runs corresponding to the adaptive solutions. These runs are indicated by the dashed lines (just visible in the

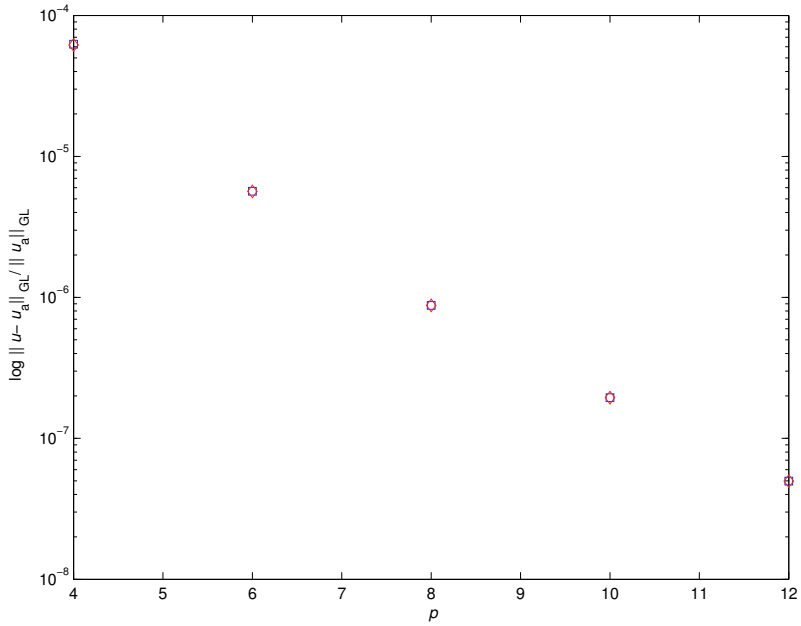

Fig. 8. Plots of $\log_{10}(||\boldsymbol{u} - \boldsymbol{u}_{\mathrm{a}}||_p / ||\boldsymbol{u}_{\mathrm{a}}^0||_p)$ as a function of p for the linear advection test. The square markers indicate the adaptive runs, while the diamonds represent the 3-control grid runs at the same polynomial degree. These two curves are nearly identical.

top-right plot), which all follow the $\ell_{\max} = 3$ curves. As the polynomial degree increases, the solution requires fewer levels of refinement to achieve the same accuracy as with the 3-control grid, again at a considerable computational savings.

3.2.2 Spatial convergence

We next consider the effects of expansion degree p on the solution error. The maximum number of refinement levels is fixed to $\ell_{\max} = 3$. Here, a Courant-limited timestep is again used with a Courant number of 0.2. The solution is integrated to a time ($t = 0.2$) chosen so that several cycles of coarsening and refinement occur (see top of Fig. 9). The initial mesh is the same as in the time convergence test. The spatial convergence result is presented in Fig. 7.

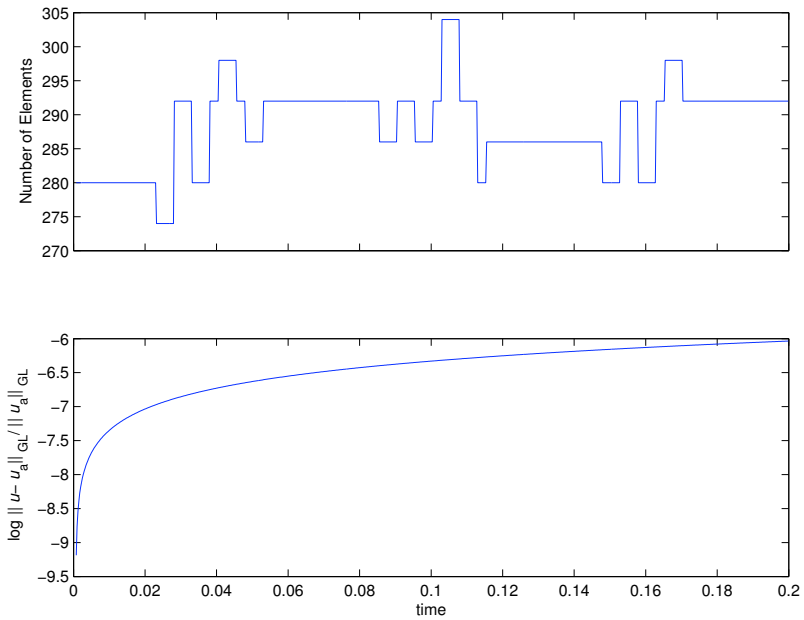

Fig. 9. Plots of diagnostic quantities vs time in the $p = 8$ linear advection test. *Top*: number of elements vs time. *Bottom*: $\log_{10}(||\boldsymbol{u} - \boldsymbol{u}_{\mathrm{a}}||_p / ||\boldsymbol{u}_{\mathrm{a}}^0||_p)$ vs *time*. The errors for the adaptive and control meshes lie on top of one another.

The anticipated spectral decay of error can be seen in Fig. 8, which also includes the uniform control solutions. We see again that the adaptive solution error decays nearly identically to the control solution, suggesting again that interpolations and polynomial truncation error introduce no deleterious effects in the linear advection case.

In Fig. 9, typical plots of two diagnostics are provided for the $p = 8$ run in Fig. 8. In the top of the figure is shown the number of elements as a function of time, and in the bottom, the error. Clearly, the adaption does not alter the monotonic behavior of the error. The error for the 3-control grid in the $p = 8$ case is also plotted in Fig. 9; the behavior of this error as a function of time is nearly identical to that for the adaptive mesh. Since the problem was initialized with a $K = 4 \times 4$ grid at $\ell_{\min} = 0$, at $\ell_{\max} = 3$, the 3-control problem has $K = 32 \times 32$ elements. The adaptive case clearly provides for a significant

savings in the number of d.o.f. required to compute the same solution.

3.3 2D Burgers equation

In this section we examine the full nonlinear $\vec{c} = \vec{u}$ version of (1). The objective is to investigate the solution errors as the mesh resolves or tracks stationary and propagating fronts that are generated and sustained by the nonlinearity of the equation. The first problem is the familiar Burgers stationary front, and the second, a radial N-wave. With the former, we investigate the effects of 2D adaptivity on a rotated 1D nonlinear problem. In the N-wave problem we consider convergence properties of a fully 2D nonlinear case.

3.3.1 Stationary Burgers front problem

The stationary Burgers front problem is the classical solution to the nonlinear advection-diffusion equation (1) with a planar front developing in the x direction. We compare the maximum derivative of the field, $|\partial_x u|_{\max}$, and the time at which the maximum occurs with the analytic solution. The classical 1D Burgers front problem for $q(y,t)$ is cast into a 2D framework by the substitution $\vec{u}(\vec{x},t) = \vec{k}q(\vec{k}\cdot\vec{x}, k^2 t)$ in (1) [13]. The initial conditions are

$$q(y,0) \equiv -\sin(\pi y) + \hat{u}_2 \sin(2\pi y). \tag{19}$$

For the first test we choose $\nu = 10^{-4}$, $\vec{k} = \vec{\imath}^1$, $\hat{u}_2 = 0$ and use biperiodic boundary conditions for $\vec{x} \in [0,1]^2$. In each case the problem is initialized with $K = 4 \times 1$ grid of a specified degree p. A BDF3/Ext3 scheme is used for the time derivative and advective terms, respectively. We initialize only at $t = t^0$ for this problem and integrate using a BDF1/Ext1 scheme to provide values at

	Mavriplis		GASpAR	
p	$T_{\max}$	$\lvert\partial_x u\rvert_{\max}$	$T_{\max}$	$\lvert\partial_x u\rvert_{\max}$
5	0.53745	167.227	0.5320	228.38977
9	0.50611	154.019	0.51074	148.04258
13	0.51103	151.496	0.51072	151.69874
17	0.51071	152.076	0.51045	152.09104
21	0.51023	152.004	0.51047	151.99624
Analytic	0.51047	152.00516	-	-

Table 1

Nonadaptive results from the stationary Burgers front problem.

t^1 and t^2. Two cases are considered: a nonadaptive case and an adaptive case with a maximum refinement level of $\ell_{\max} = 3$. In the nonadaptive case, the x^1-coordinates of the element vertices are at $x^1 = 0, \pm 0.05, \pm 1$, whereas in the adaptive case, the elements are initially uniform. The derivative error criterion is used in this problem, and the tolerance and coarsening multiplication are 1.0 and 0.5, respectively. Table 1 presents the nonadaptive results, together with the results of the nonadaptive runs from [28]. The quantity $|\partial_x u|_{\max}$ is the maximum of the derivative, and $T_{\max}$ is the time at which this occurs. We have verified that the $\mathbb{L}_2$ error of the solution is consistent with the error in the derivative. We note that the $p = 5$ case is considerably worse than the results presented in [28]. This may be due to differences between the bases used in the two methods [2]. We note that the errors in $T_{\max}$ for our nonadaptive case are uniformly better than those in [28], while the errors in $|\partial_x u|_{\max}$ are comparable for $p > 5$.

In Table 2 we present the results from the adaptive case and the so-called reference and control solutions. The *reference* solution runs on a uniform grid with the same number of elements as the adaptive solution at $T_{\max}$. Thus, it offers a solution computed with roughly as many d.o.f. as the adaptive

	Adaptive		Reference		Control	
p	$T_{\max}$	$\lvert\partial_x u\rvert_{\max}$	$T_{\max}$	$\lvert\partial_x u\rvert_{\max}$	$T_{\max}$	$\lvert\partial_x u\rvert_{\max}$
5	0.52679	224.36164	–	–	0.52674	224.37214
9	0.51095	153.39634	0.52635	227.53596	0.51095	153.39633
13	0.51030	150.03130	0.51219	181.02024	0.51030	150.03130
17	0.51048	152.25110	0.51082	149.57372	0.51048	152.25110
21	0.51047	152.00556	0.51021	147.22940	0.51047	152.00565
Analytic	0.51047	152.00516	-	-	-	-

Table 2

Adaptive, reference, and control results from the stationary Burgers front problem.

solution, and hence requires about the same computational effort. Clearly, the ability to resolve the front is critical in this case; the reference solution for $p = 5$ actually diverges, and good solutions are not produced until $p > 13$. The control solutions, as expected, are all nearly identical to the adaptive solutions. This fact suggests that our refinement criteria enable the adaptive mesh to capture the formation of the front accurately, at a significantly reduced cost.

3.3.2 N-wave problem

The 2D Cole-Hopf transformation

$$\vec{u} = -2\nu\vec{\nabla}\ln\chi. \tag{20}$$

transforms (1) into a heat equation for χ. Choosing a source solution [39]

$$\chi(\vec{x},t) = 1 + \frac{a}{t}\exp-\frac{(\vec{x}-\vec{x}^0)^2}{4\nu t},$$

we obtain the solution to (1) immediately from (20):

$$\vec{u}(\vec{x},t) = \frac{\vec{x}-\vec{x}^0}{t}\frac{a}{a+t\exp((\vec{x}-\vec{x}^0)^2/4\nu t)}. \tag{21}$$

This is a radial N-wave, where $\vec{x}^0 = (\vec{\iota}^1 + \vec{\iota}^2)/2$ is the location of the center. This solution is singular as $t \to 0$, so we initialize at a finite time $t^0 = 5\times10^{-2}$.

For this test problem, we choose $\nu = 5 \times 10^{-3}$ and $a = 10^4$. Dirichlet boundary conditions (2) on $\mathbb{D} = [0,1]^2$ are imposed at each timestep by using (21) on $\partial\mathbb{D}$. The initial grid consists of $K = 4 \times 4$ elements, and we consider only the full adaptive case with $\ell_{\max} = 4$. This $\ell_{\max}$ provides a control mesh that is expensive to compute on; hence, we examine the temporal and spatial convergence of only the adaptive solutions. The refinement criteria are the same as in Section 3.2.

Fig. 10 presents a time series of a typical N-wave solution illustrating the refinement patterns characteristic of all the runs. For simplicity only one quadrant of the axially symmetric solution is presented. As the front propagates outward, the grid refines to track it, while in the center where the velocity components are planar, the grid coarsens. Note that the front does not steepen in this problem, as it does in the planar front problem (Section 3.3); it simply decays as it moves outward.

In considering the time convergence, we set $p = 14$ and vary Δt to produce the error plot in Fig. 11. As was the case previously, each point in this plot reflects a single run integrated at the indicated fixed Δt and the error at $t = 0.11$. This integration time interval was chosen to provide a number of refinement and coarsening events. Nevertheless, the solution converges with Δt, at order (slope) 3.01.

In order to check spatial convergence, the solution is integrated to $t = 0.11$ by using a variable Δt and p and fixed Courant number of 0.15. In Fig. 12 we present the final $\mathbb{L}_2$ error vs p. As with the linear advection case, the error behaves spectrally for a finite time integration.

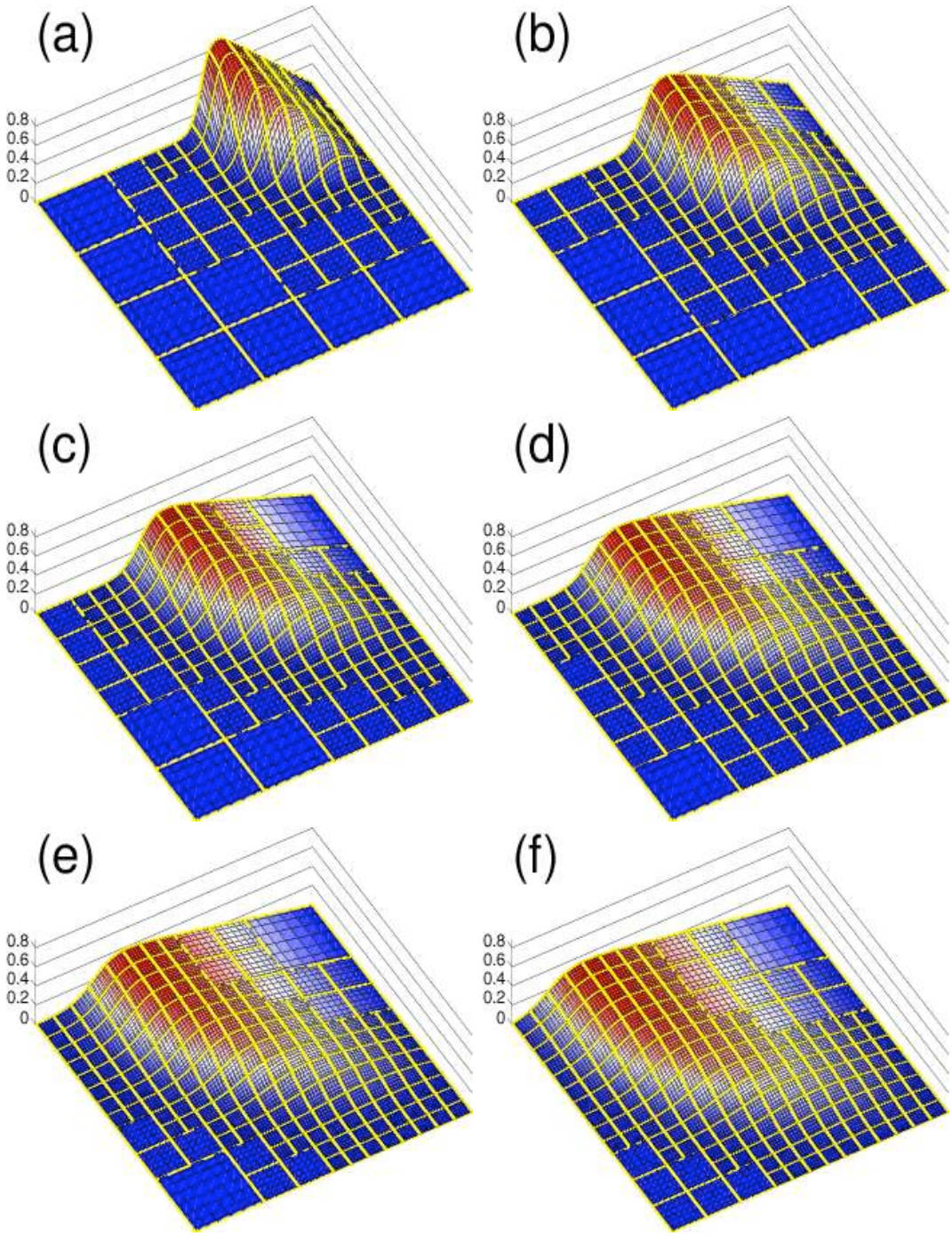

Fig. 10. Surface plots of $u^1(\vec{x},t)$ solved from (1), showing $\vec{x} \in [\frac{1}{2},1]^2$ and $K/4 = 88$, 121, 139, 172, 181, 190 as $t = 0.18$, 0.33, 0.48, 0.65, 0.81, 1.00. Here $\vec{c} = \vec{u}$, $\nu = 5 \times 10^{-3}$, $p = 8$, and (21) is the initial condition.

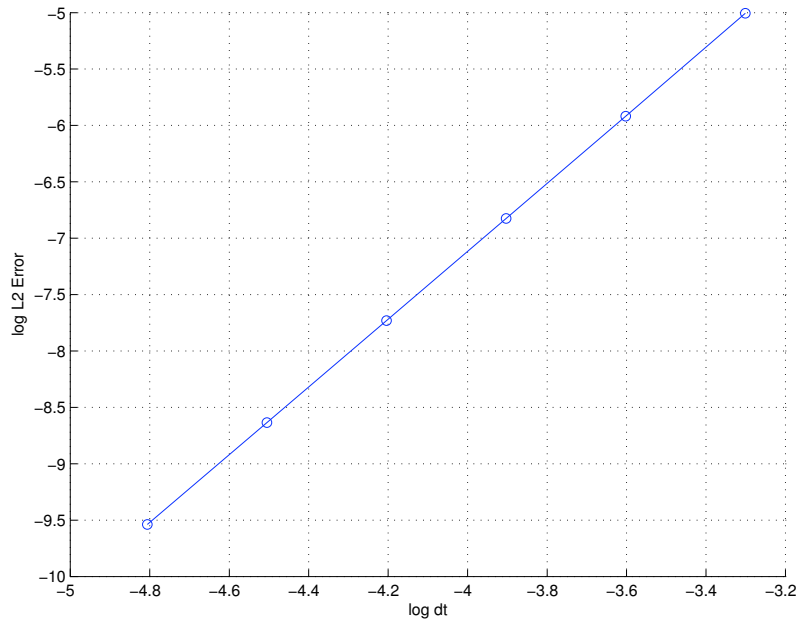

Fig. 11. Plot of $\log_{10}(||\boldsymbol{u} - \boldsymbol{u}_{\mathrm{a}}||_p/||\boldsymbol{u}_{\mathrm{a}}^0||_p)$ for the N-wave problem vs $\log_{10} \Delta t$, for different $p = 14$. The slope of the line is 3.01.

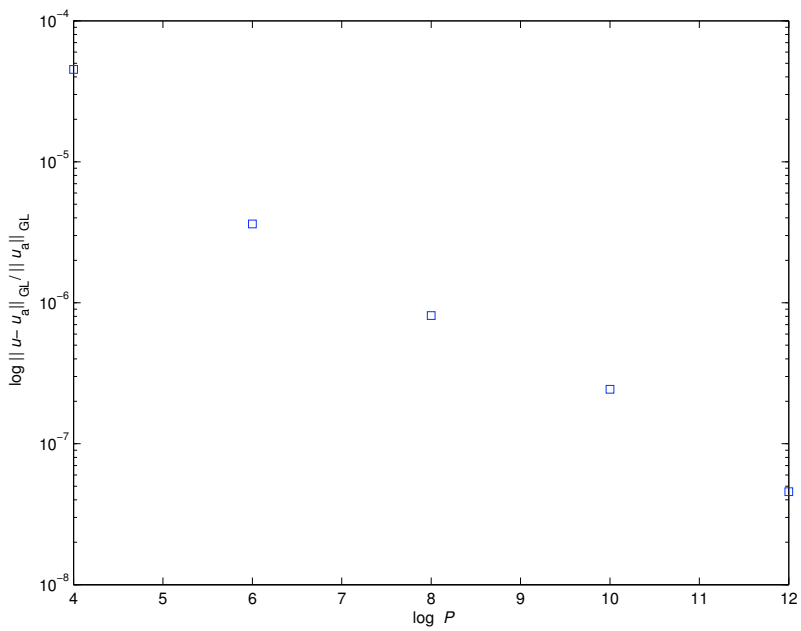

Fig. 12. Plot of $\log_{10}(||\boldsymbol{u} - \boldsymbol{u}_{\mathrm{a}}||_p/||\boldsymbol{u}_{\mathrm{a}}^0||_p)$, as a function of p for the N-wave test.

4 Discussion and conclusion

We have presented an overview of the GASpARcode, concentrating on the continuous Galerkin discretization of a single (generalized advection-diffusion)

equation to illustrate the construction of the weak and collocation operators and to highlight aspects of the code design. We have provided a detailed description of the underlying mathematics and constructs for connectivity and a new adaptive mesh refinement algorithm that are used in the code, and we have shown how they maintain continuity between conforming and nonconforming elements. In the process, we have established a consistent set of rules for refinement and coarsening. In addition, we have described the refinement criteria and using them, have demonstrated with several test problems the ability of DARe to model accurately a variety of 2D structures that evolve as a result of linear and nonlinear advective and dissipative dynamics.

The test problems suggest that for resolving isolated structures, dynamic adaptive refinement can offer a substantial computational savings over a either a pseudo-spectral or conforming spectral-element method for the same control resolution. But for turbulent flows, how likely is it that only a few isolated structures will exist? And how dependent is the time evolution of the flow on these structures, such that, if they are resolved, the statistical properties of the overall flow will be preserved using DARe? One aspect of GASpAR that can help answer these questions is that the fields solved for need not be those on which adaption criteria directly operate. The user is free to specify any functionals of the fundamental fields (velocity, pressure, etc.) for use in tagging elements for refinement or coarsening. For example, while the velocity is actually solved for in (1), the adaption criteria might operate on kinetic energy, vorticity, or enstrophy. Arguably, some fully developed turbulent flows viewed in terms of the fundamental fields may be too intricate to benefit from DARe. Nevertheless, when viewed w.r.t. some functional, some relevant structures, when resolved, may allow for accurate simulation of the significant

dynamics of the overall flow.

A Spectral-element formalism

A.1 Canonical 1D element

In this appendix we summarize our notation and results from the SEM literature. Any dependent variable $u = u(\xi)$ in the domain $\xi \in [-1, 1]$ may be approximated by its projection $\mathcal{P}_p u$ on the space $\mathbb{V}_p$ of polynomials of degree p, using u-values on any $N_p \equiv p + 1$ distinct nodal points ξ_j:

$$u = \mathcal{P}_p u + \mathcal{E}_p u \approx \mathcal{P}_p u \equiv \sum_{j=0}^{p} u(\xi_j)\phi_j, \tag{A.1}$$

where $\mathcal{E}_p u$ is the pointwise error and

$$\phi_j(\xi) \equiv \prod_{j' \neq j} \frac{\xi \ - \xi_{j'}}{\xi_j - \xi_{j'}} \xrightarrow[\xi \to \xi_{j''}]{} \delta_{j,j''}, \tag{A.2}$$

denotes the N_p unique Lagrange interpolating polynomials of degree p. Now let the ξ_j be the Gauss-Lobatto-Legendre quadrature nodes

$$\xi_j \equiv (j+1)\text{th least root of } (1 - \xi^2)\frac{\mathrm{dL}_p}{\mathrm{d}\xi}, \tag{A.3}$$

where L_j is the standard Legendre polynomial of degree j and norm $(j + \frac{1}{2})^{-\frac{1}{2}}$. In this case

$$\phi_{j'}(\xi) = w_{j'} \sum_{j=0}^{p} \mathrm{L}_j(\xi_{j'})\mathrm{L}_j(\xi) / \sum_{j''=0}^{p} w_{j''}\mathrm{L}_j(\xi_{j''})^2, \tag{A.4}$$

and the integral

$$\langle u \rangle_1 \equiv \int_{-1}^{1} u(\xi)\,\mathrm{d}\xi = \sum_{j=0}^{p} w_j u(\xi_j) + \mathcal{R}_p u(\xi'), \tag{A.5}$$

where

$$w_j \equiv 2/pN_p\mathrm{L}_p(\xi_j)^2 \tag{A.6}$$

denotes the Gauss-Lobatto-Legendre weights, $\mathcal{R}_p \equiv -2^{2p+1}\frac{p^3 N_p (p-1)!^4}{(2p+1)(2p)!^3}(\,\mathrm{d}/\,\mathrm{d}\xi)^{2p}$ is the residual operator [37] and $\xi' \in\,]-1,1[$. Then the mean-square error is bounded as

$$\langle(\mathcal{E}_p u)^2\rangle_1 \propto p^{1-2Q}\sum_{q=0}^{Q}\|u^{(q)}\|^2 \tag{A.7}$$

for any order Q of square-integrable derivative. Thus if u is infinitely smooth then $\mathcal{P}_p u$ converges to u *spectrally.*

A.2 General 1D spectral elements

Now let $[-1,1]$ be subdivided and covered by K^1 disjoint 1D elements $]x_{k-1}, x_k[\, \equiv \mathbb{E}^1_k \equiv \vartheta_k(]-1,1[)$, where

$$\vartheta_k(\xi) \equiv x_{k-1} + \tfrac{1}{2}h^1_k(1+\xi), \qquad k \in \left\{1, \cdots K^1\right\}, \tag{A.8}$$

is a coordinate map with inverse $\vartheta_k^{-1}(x) = 2(x - x_{k-1})/h^1_k - 1$. (Other invertible maps may sometimes be preferable.) For $x_0 \equiv -1$, $x_{K^1} \equiv 1$ and positive $h^1_k \equiv x_k - x_{k-1}$ one has $\mathbb{E}^1_k \bigcap \mathbb{E}^1_{k'} = \varnothing$ if $k \neq k'$ and

$$[-1,1] = \bigcup_{k=1}^{K^1} \bar{\mathbb{E}}^1_k. \tag{A.9}$$

Then u may be approximated by its projections $\mathcal{P}_{k,p}u$ on the space $\mathbb{V}_{\boldsymbol{h}^1,p}$ of piecewise polynomials of degree p on the $\mathbb{E}^1_k$, by using u-values on the $K^1 N_p$ mapped nodal points

$$x_{j,k} \equiv \vartheta_k(\xi_j) \tag{A.10}$$

generalized from (A.3). That is, (A.1) generalizes to

$$u = \sum_{k=1}^{K^1}\left(\mathcal{P}_{k,p}u + \mathcal{E}_{k,p}u\right), \qquad \mathcal{P}_{k,p}u \equiv \sum_{j=0}^{p} u(x_{j,k})\phi_{j,k}, \tag{A.11}$$

where $\mathcal{E}_{k,p}u \equiv \mathcal{E}_p(u \circ \vartheta_k) \circ \vartheta_k^{-1}$, and (A.2) generalizes to

$$\phi_{j,k}(x) \equiv 1_{\bar{\mathbb{E}}_k^1}(x)\phi_j \circ \vartheta_k^{-1}(x) \xrightarrow[x \to x_{j',k'}]{} \begin{cases} 1, & x_{j,k} = x_{j',k'}, \\ 0, & \text{otherwise.} \end{cases} \tag{A.12}$$

We have adopted the additional notations $f \circ g(x) \equiv f(g(x))$ and $1_{\mathbb{S}}(x) \equiv 1$ ($x \in \mathbb{S}$), 0 (otherwise). Then (A.5) generalizes to

$$\langle u \rangle_1 = \sum_{k=1}^{K^1} \int_{x_{k-1}}^{x_k} u(x)\,\mathrm{d}x, \quad \int_{x_{k-1}}^{x_k} u(x)\,\mathrm{d}x = \sum_{j=0}^{p} w_{j,k} u(x_{j,k}) + \mathcal{R}_{k,p} u(x'_k), \tag{A.13}$$

where (A.6) generalizes to

$$w_{j,k} \equiv |\frac{\mathrm{d}}{\mathrm{d}\xi}\vartheta_k(\xi_j)| w_j, \tag{A.14}$$

$\mathcal{R}_{k,p}u \equiv (h_k^1/2)^{2p+1}\mathcal{R}_p(u \circ \vartheta_k) \circ \vartheta_k^{-1}$ and $x'_k \in \mathbb{E}_k^1$.

A.3 General d-dimensional spectral elements

Generalizing (A.9), assume the d-dimensional problem domain $\mathbb{D}$ can be partitioned into K disjoint images of $[-1,1]^d \equiv \left\{\vec{\xi} \mid \xi^\mu \in [-1,1]\right\}$ as

$$\bar{\mathbb{D}} = \bigcup_{k=1}^{K} \bar{\mathbb{E}}_k, \tag{A.15}$$

where $\mathbb{E}_k \equiv \vec{\vartheta}_k(]-1,1[^d)$ has diameter

$$h_k \equiv \max_{\mu} \max_{\vec{x},\vec{x}' \in \bar{\mathbb{E}}_k} |x^\mu - x'^\mu| \tag{A.16}$$

and $\vec{\vartheta}_k(\vec{\xi})$ has inverse $\vec{\vartheta}_k^{-1}(\vec{x})$ but is not necessarily linear. Now generalizing (A.11), one may approximate a field $u(\vec{x})$ by its projections $\mathcal{P}_{k,\vec{p}}u$ on the space $\mathbb{V}_{\boldsymbol{h},\vec{p}}$ of piecewise polynomials of degree p^μ in coordinate x^μ on the $\mathbb{E}_k$, using u-values on the $N_{K,\vec{p}} \equiv K \prod_{\mu=1}^{d} N_{p^\mu}$ mapped nodes

$$\vec{x}_{\vec{\jmath},k} \equiv \vec{\vartheta}_k(\vec{\xi}_{\vec{\jmath}}), \tag{A.17}$$

generalized from (A.10), where $\xi^{\mu}_{\vec{\jmath}} \equiv \xi_{\jmath^{\mu}}$ are d-dimensional GLL nodes. That is, (A.11) generalizes to

$$u \approx \mathcal{P}_{\boldsymbol{h},\vec{p}}u \equiv \sum_{k=1}^{K} \mathcal{P}_{k,\vec{p}}u, \qquad \mathcal{P}_{k,\vec{p}}u \equiv \sum_{\vec{\jmath}\in\mathbb{J}} u(\vec{x}_{\vec{\jmath},k})\phi_{\vec{\jmath},k}, \tag{A.18}$$

where $\mathbb{J} \equiv \{\vec{\jmath} \mid \jmath^{\mu} \in \{0, \cdots p^{\mu}\}\}$, (A.12) generalizes to

$$\phi_{\vec{\jmath},k}(\vec{x}) \equiv 1_{\mathbb{E}_k}(\vec{x})\phi_{\vec{\jmath}} \circ \vec{\vartheta}_k^{-1}(\vec{x}) \xrightarrow[\vec{x}\to\vec{x}_{\vec{\jmath}',k'}]{} \begin{cases} 1, & \vec{x}_{\vec{\jmath},k} = \vec{x}_{\vec{\jmath}',k'} \\ 0, & \text{otherwise,} \end{cases} \tag{A.19}$$

and $\phi_{\vec{\jmath}}(\vec{\xi}) \equiv \prod_{\mu=1}^{d} \phi_{\jmath^{\mu}}(\xi^{\mu})$. The appropriate approximation of a vector

$$\vec{u} = \sum_{\mu=1}^{d} u^{\mu}\vec{\imath}^{\mu} \approx \mathcal{P}_{\boldsymbol{h},\vec{p}}\vec{u} = \vec{\boldsymbol{\phi}}^{\mathrm{T}}\boldsymbol{u}$$

uses $\vec{\boldsymbol{\phi}}$ with entries

$$\vec{\phi}^{\mu}_{\vec{\jmath},k} \equiv \phi_{\vec{\jmath},k}\vec{\imath}^{\mu} \tag{A.20}$$

and $\boldsymbol{u}$ with entries $u^{\mu}_{\vec{\jmath},k} \equiv u^{\mu}(\vec{x}_{\vec{\jmath},k})$. For scalars u (A.13) generalizes to

$$\begin{aligned} \langle u \rangle &\equiv \int_{\mathbb{D}} \cdots \int u(\vec{x})\,\mathrm{d}^d\vec{x} = \sum_{k=1}^{K} \int_{\mathbb{E}_k} \cdots \int u(\vec{x})\,\mathrm{d}^d\vec{x} \\ &\approx \sum_{k=1}^{K} \sum_{\vec{\jmath}\in\mathbb{J}} w_{\vec{\jmath},k} u(\vec{x}_{\vec{\jmath},k}) \equiv \langle u \rangle_{\mathrm{GL}}, \end{aligned} \tag{A.21}$$

where (A.14) generalizes to

$$w_{\vec{\jmath},k} \equiv |\det \vec{\nabla}_{\vec{\xi}}\vec{\vartheta}_k(\vec{\xi}_{\vec{\jmath}})| \prod_{\mu=1}^{d} w_{\jmath^{\mu}}. \tag{A.22}$$

Finally, variational formulation depends on the inner product from (A.21):

$$\langle u, v \rangle \equiv \langle uv \rangle \approx \sum_{k=1}^{K} \sum_{\vec{\jmath}\in\mathbb{J}} w_{\vec{\jmath},k} u(\vec{x}_{\vec{\jmath},k}) v(\vec{x}_{\vec{\jmath},k}) \equiv \langle u, v \rangle_{\mathrm{GL}} \tag{A.23}$$

for scalars, $\langle \vec{u}, \vec{v} \rangle \equiv \sum_{\mu=1}^{d} \langle u^{\mu}, v^{\mu} \rangle$ for vectors, $\left\langle \vec{\vec{u}}, \vec{\vec{v}} \right\rangle \equiv \sum_{\mu,\mu'=1}^{d} \left\langle u^{\mu,\mu'}, v^{\mu,\mu'} \right\rangle$ for tensors, and so forth.

Acknowledgments

We thank Rich Loft, Peter Sullivan, Steve Thomas and Joe Tribbia for help at the onset of this work in using spectral element methods, and Huiyu Feng and Catherine Mavriplis for several useful discussions. Computer time was provided by NSF under sponsorship of the National Center for Atmospheric Research. The third author was supported by the U.S. Department of Energy under Contract W-31-109-ENG-38.

References

[1] Anagnostou, G., Y. Maday, C. Mavriplis, and A. T. Patera, "On the mortar element method: Generalizations and implementation," in *Third International Symposium on Domain Decomposition Methods,* pp. 157–173, SIAM, (1989).

[2] Basdevant, C, Deville, M., Haldenwang, P., Lacroix, J. M., Quazzani, J., Peyret, R., Orlandi, P, and Patera, A., "Spectral and finite difference solutions of the Burgers equation," *Comp. Fluids*, **14**, pp., 23–41 (1986).

[3] Belgacem, F. B., "The mixed mortar finite element method for the incompressible Stokes problem: Convergence analysis," *SIAM J. Numer. Anal.,* **37,**(4) pp. 1085-1100 (2000).

[4] Bernardi, C., Y. Maday, C. Mavriplis, and A. T. Patera, "The mortar element method applied to spectral discretizations", *Proceedings of the Seventh International Conference on Finite Element Methods in Flow Problems,* Huntsville (1989).

[5] C. Canuto, M. Yousuff Hussaini, A. Quateroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics.* New York, Springer–Verlag (1988).

[6] Casadei F., E. Gabellini, G. Fotia, F. Maggio, and A. Quarteroni, "A mortar spectral/finite element method for complex 2D and 3D elastodynamic problems," *Comp. Meth. Appl. Mech. Eng.*, **191**, 5119–5148 (2002).

[7] Chaljub, E., Y. Capdeville, and J. P. Vilotte, "Solving elastodynamics in a fluid-solid heterogeneous sphere: A parallel spectral element approximation on non-conforming grids," *J. Comp. Phys.*, **187**, 457–491 (2003).

[8] Chen, S. Y., D. D. Holm, L. G. Margolin, E. J. Olson, E. S. Titi, and S. Wynne, "The Camassa-Holm equations as a closure model for turbulent channel and pipe flows," *Phys. Rev. Lett.* **81**, 5338–5341 (1998).

[9] Deville, M. O., P. F. Fischer and E. H. Mund, *High-Order Methods for Incompressible Fluid Flow.* Cambridge, Cambridge University Press (2002).

[10] Elmegreen, B. G., & J. Scalo "Interstellar turbulence, I: Observations and Processes" *Ann. Rev. Astron. Astrophys.* **42**, 211–273 (2004).

[11] Feng, H. and C. Mavriplis, "Adaptive spectral element simulations of thin flame sheet deformations," *J. Sci. Comp.*, **17**, pp. 1–3 (2002).

[12] Fischer, P. F., G. W. Kruse, and F. Loth, "Spectral element methods for transitional flows in complex geometries," *J. Sci. Comput.,* **17,** 1, pp. 81–98 (2002).

[13] Fournier, A., G. Beylkin and V. Cheruvu, "Multiresolution adaptive space refinement in geophysical fluid dynamics simulation," *Lecture Notes Comp. Sci. Eng.,* **41,** pp. 161–170 (2005).

[14] Fournier, A. and D. Rosenberg, "Multiresolution adaptive spectral-element modeling of geophysical fluid dynamics using GASpAR," in preparation (2005).

[15] Fournier, A., M. A. Taylor, and J. J. Tribbia, "The spectral element atmosphere model (SEAM): High-resolution parallel computation and localized resolution of regional dynamics," *Mon. Wea. Rev.*, **132**, pp. 726–748

(2004).

[16] Henderson, R. D., "Unstructured spectral element methods for simulation of turbulent flows," *J. Comp. Phys.* **122**, pp. 191–217 (1995).

[17] Henderson, R. D., "Dynamic refinement algorithms for spectral element methods," *Comput. Methods Appl. Mech. Engrg.* **175**, pp. 395–411 (1999).

[18] Isihara T., Y. Kaneda, M. Yokokawa, K. Itakura, and A. Uno, "Spectra of energy dissipation, enstrophy and pressure by high-resolution direct numerical simulations of turbulence in a periodic box," *J. Phys. Soc. Japan* **72**, pp. 983–986 (2003).

[19] Iskandarani, M., D. B. Haidvogel, and J. C. Levin, "A three-dimensional spectral element model for the solution of the hydrostatic primitive equations" *J. Comp. Phys.* **186**, pp. 397–426 (2003).

[20] Karniadakis, G. E., M. Israeli, and S. A. Orszag, "High–Order splitting methods for the incompressible Navier-Stokes equations " *J. Comp. Phys.* **97**, pp. 414–443 (1991).

[21] Karniadakis, G. E. and S. J. Sherwin, *Spectral/hp Element Methods for CFD*. New York, Oxford Iniversity Press (1999).

[22] Kopriva D. A., S. L. Woodruff, and M. Y. Hussaini, "Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method," *Int. J. Num. Meth. Eng.*, **53**, pp. 105–122 (2002).

[23] Kruse, G. W., "Parallel nonconforming spectral element solution of the incompressible Navier–Stokes equations in three dimensions," Ph.D. Dissertation, Division of Applied Mathematics, Brown University (1997).

[24] Levin, J. G., M. Iskandarani, and D. B. Haidvogel, "A nonfoncorming spectral element ocean model," *Intl. J. Numer. Meth. Fluids* **34**, pp. 495–525 (2000).

[25] Maday, Y., C. Mavriplis, and A. T. Patera, "Nonconforming mortar el-

ement methods: Application to spectral discretizations," in *Domain Decomposition Methods,* pp. 392–418, SIAM (1989). Also ICASE Report 88-59.

[26] Maday, Y., A. T. Patera, and E. M. Rønquist, "The $\mathbb{P}_N$-$\mathbb{P}_{N-2}$ method for the approximation of the Stokes problem," Publications du Laboratoire d'Analyse Numérique, Université Pierre et Marie Curie, Centre National de la Recherche Scientifique (1992).

[27] Mavriplis, C., "Nonconforming discretizations and a posteriori error estimations for adaptive spectral element techniques," Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA (1993).

[28] Mavriplis, C., "Adaptive mesh strategies for the spectral element method," *Comput. Methods Appl. Mech, Engrg.* **116**, pp. 77–86 (1994).

[29] C. Meneveau and J. Katz, "Scale-invariance and turbulence models for large-eddy simulation," *Annu. Rev. Fluid Mech.* **32**, pp. 1–32 (2000).

[30] Orszag, S. A., "Spectral methods for problems in complex geometries," *J. Comp. Phys.* **37**, pp. 70–92 (1980).

[31] Patera, A., "A spectral element method for fluid dynamics: laminar flow in a channel expansion," *J. Comp. Phys.* **54**, pp. 468–488 (1984).

[32] E. Rønquist "Convection treatment using spectral elements of different order," *Intl. J. Num. Meth. Fluids* **22**, pp. 241–264 (1996).

[33] Shewchuck, Richard J. "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain," http://www-2.cs.cmu.edu/jrs/jrspapers.html (1994).

[34] St-Cyr, A., J. M. Dennis, S. J. Thomas, and H. Tufo, "An adaptive nonconforming spectral element atmospheric model", *SIAM J. Sci. Comp.*, submitted (2004).

[35] I. Sytine, D. Porter, P. Woodward, S. Hodson and K-H Winkler, "Con-

vergence tests for the Piecewise Parabolic Method and Navier-Stokes solutions for homogeneous compressible turbulence", *J. Comp. Phys.* **158**, pp. 225–238 (2000).

[36] Tufo, H.M., Fischer, P.F., "Terascale Spectral element Algorithms and Implementations", Proceedings of the ACM/IEEE SC99 Conference on High Performance Networking and Computing, IEEE Computer Soc. (1999).

[37] Eric W. Weisstein. "Lobatto Quadrature." From MathWorld—A Wolfram Web Resource. http://mathworld.wolfram.com/LobattoQuadrature.html.

[38] Eric W. Weisstein. "Conjugate Gradient Method." From MathWorld—A Wolfram Web Resource. http://mathworld.wolfram.com/ConjugateGradientMethod.html.

[39] Whitham, G.B., *Linear and Nonlinear Waves* New York, Wiley (1974).

The submitted manuscript has been created in part by the University of Chicago as Operator of Argonne National Laboratory ("Argonne") under Contract No. W-31-109-ENG-38 with the U.S. Department of Energy. The U.S. Government retains for itself, and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.