Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 215 (2006) 59–80

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

Geophysical–astrophysical spectral-element adaptive refinement (GASpAR): Object-oriented *h*-adaptive fluid dynamics simulation

Duane Rosenberg [a,*], Aimé Fournier [a], Paul Fischer [c], Annick Pouquet [b]

[a] *Institute for Mathematics Applied to Geosciences, National Center for Atmosphere Research, P.O. Box 3000, Boulder, CO 80307-3000, USA*
[b] *Earth and Sun Systems Laboratory, National Center for Atmospheric Research, P.O. Box 3000, Boulder, CO 80307-3000, USA*
[c] *Mathematics and Computer Science Division, Argonne National Laboratory, Illinois 60439-4844, USA*

Received 22 April 2005; received in revised form 18 October 2005; accepted 20 October 2005
Available online 22 December 2005

Abstract

An object-oriented geophysical and astrophysical spectral-element adaptive refinement (GASpAR) code is introduced. 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. The formalism accommodates both conforming and non-conforming elements. Several aspects of this code derive from existing methods, but here are synthesized into a new formulation of dynamic adaptive refinement (DARe) of non-conforming *h*-type. As a demonstration of the code, several new 2D test cases are introduced that have time-dependent analytic solutions and exhibit localized flow features, including the 2D Burgers equation with straight, curved-radial and oblique-colliding fronts. These are proposed as standard test problems for comparable DARe codes. Quantitative errors are reported for 2D spatial and temporal convergence of DARe.
© 2005 Elsevier Inc. All rights reserved.

Keywords: Spectral element; Numerical simulation; Adaptive mesh; AMR

1. Introduction: a need for high-accuracy dynamic adaptivity

Accurate and efficient simulation of strongly turbulent flows is a prevalent challenge in many atmospheric, oceanic, and astrophysical applications. New simulation codes are needed to investigate such flows in the parameter regimes that interest the geophysics communities. Turbulent flows 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

[*] Corresponding author. Tel.: +1 303 497 1636.
E-mail addresses: duaner@ucar.edu (D. Rosenberg), fournier@ucar.edu (A. Fournier), fischer@mcs.anl.gov (P. Fischer), pouquet@ucar.edu (A. Pouquet).

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.10.031

turbulent motions. Nonlinearities prevail when the Reynolds number Re is large. The number of 3-dimensional degrees of freedom (d.o.f.) increases as $Re^{9/4}$ as $Re \to \infty$ in the Kolmogorov 1941 framework [16, Section 7.4]. For aeronautic flows often $Re > 10^6$, but for geophysical flows often $Re \gg 10^8$ [11,28]. Also, computations of turbulent flows must contain enough scales to encompass the energy-containing and dissipative scale ranges *distinctly*. Uniform-grid convergence studies on 3D compressible-flow simulations show that in order to achieve the desired scale content, uniform grids must contain at least 2048^3 cells [33]. 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 [19], but the Taylor Reynolds number ($\propto \sqrt{Re}$) is still no more than ≈700, very far from what is required for most geophysical flows. The *main goal of the present code development* is to ask, if the significant structures of the flow are indeed sparse, so that their dynamics can be followed accurately even if they are embedded in random noise, then does dynamic adaptivity offer a means for achieving otherwise unattainable large Re values. Thus, 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, [9,29]) make them desirable for direct numerical simulation of geophysical turbulence. Perhaps most significant is the fact that SEMs performed at high polynomial degree are inherently minimally diffusive and dispersive. This property is clearly important when trying to simulate high-Re flows with multiple spatial and temporal scales that characterize turbulence. Also, because SEMs use finite elements, they can be used in very efficient high-resolution turbulence studies in domains with complicated boundaries. It is an important feature that SEMs are naturally parallelizable (e.g., [15]). Equally important, SEMs not only provide spectral convergence when the solution is smooth (see Appendix Eq. (A.3)), but are also effective when the solution is not smooth.

Our goal in this paper is to describe GASpAR and, in particular, the procedures used in our dynamic adaptive refinement (DARe) technique. We provide SEM and DARe algorithm *details* here that are not available elsewhere, in the hope of supporting readers who wish to create their own codes. Furthermore, we propose several linear and nonlinear problems as standards to test fundamental aspects of flows that are encountered in turbulence studies, and use these to test our DARe algorithms. Because these problems have known exact time-dependent solutions, quantitative errors can be reported for DARe simulations. Our code is object-oriented, and we will describe how object-oriented programming serves our purposes. The code is parallelized, but we will discuss this aspect only when it is intrinsic to the algorithms. While we are motivated by the performance potential of SEMs generally [8,34], we do not emphasize performance metrics in the present paper, in favor of focusing on algorithmic detail and solution accuracy.

First we describe (Section 2.2) SEM discretization on a particular class of problems and introduce many of the required formulas, operators, and so forth. We explain (Section 2.4) how continuity is maintained between non-conforming elements. We provide linear-solver details in Section 2.5, and introduce innovations required to solve on non-conforming elements. In Section 2.6, we present our new adaptive-mesh algorithms: how neighboring elements are found, how conformity is established, and the procedures for refinement and coarsening. In Section 2.6.3, we describe a new implementation of element-boundary communication. DARe criteria are discussed in Section 2.6.4. Then, in Section 3 we propose and perform examples from two test-problem classes with time-dependent analytic solutions: the linear advection–diffusion equation (Section 3.2), demonstrating feature tracking of smooth and isolated features; and the 2D Burgers equation (Section 3.3), testing the ability of DARe to track well-defined increasingly sharp structures arising from nonlinear dynamics. In Section 4, we offer some conclusions, as well as comments on potential application of GASpAR to geophysical turbulence simulations.

2. Temporal and dynamically adaptive spatial discretizations

2.1. Adaptive-mesh geometry

Conforming adaptive methods (where entire element boundaries geometrically coincide, as in Fig. 1a) on quadrilaterals and hexahedra are gradually being replaced by non-conforming adaptive methods. One reason is that locally adaptive mesh generation for conforming methods is complicated [30]. Another reason is that

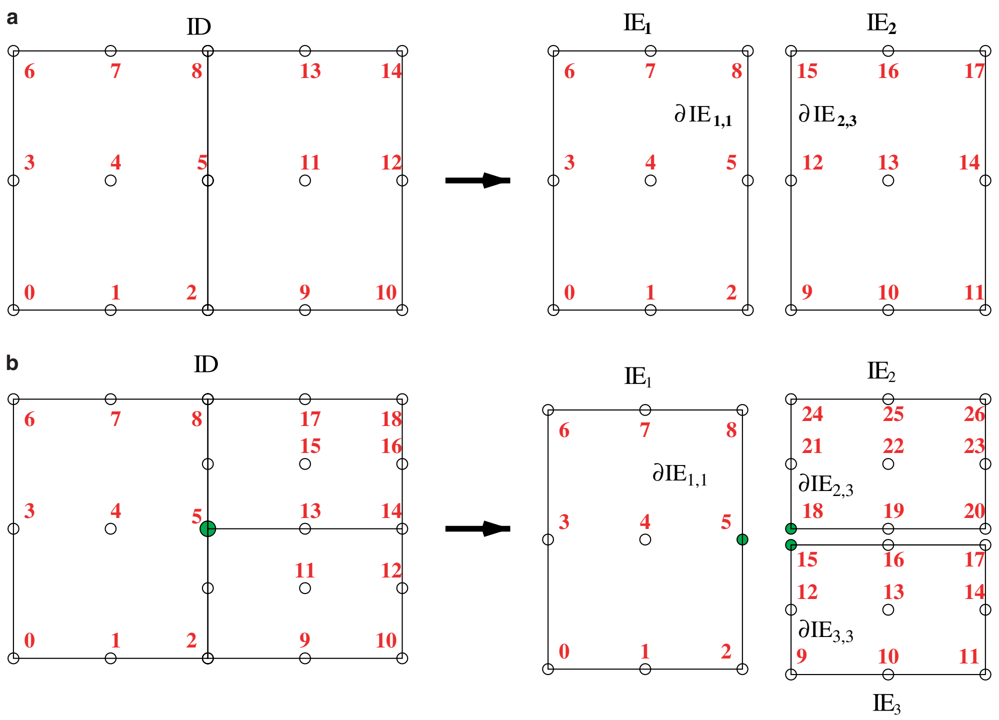

Fig. 1. (a) Conforming degree $p = 2$ mesh showing the mapping of global (i.e., unique) d.o.f. in the domain $\bar{\mathbb{D}}$ to local (i.e., redundant) d.o.f. in the elements $\mathbb{E}_k$. Edge subscripts give element key k and edge index from $s = 0$ counterclockwise to $s = 3$. Element $\mathbb{E}_1$ is bounded at the east by $\partial\mathbb{E}_{1,1}$ and $\mathbb{E}_2$ at the west by $\partial\mathbb{E}_{2,3} = \partial\mathbb{E}_{1,1}$. Interface matching occurs by assignment, so the assembly matrix $\mathbf{A}_c$ is Boolean. (b) Geometrically *non*-conforming (functionally conforming) 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 $\partial\mathbb{E}_{1,1} = \partial\mathbb{E}_{2,3} \cup \partial\mathbb{E}_{3,3}$. Interface matching occurs by *interpolation* of global d.o.f. from the function space associated with $\partial\mathbb{E}_{1,1}$ onto the union of those associated with the $\partial\mathbb{E}_{k,3}$, which contains the function space of $\partial\mathbb{E}_{1,1}$.

adaptive conforming meshes can lead to high-aspect-ratio elements that can cause difficulties for a linear solver [13]. Moreover, the fact that non-conforming elements can better localize mesh refinement implies that the computational cost over all elements can be reduced [24].

Non-conforming elements can be *geometrically* and/or *functionally* non-conforming. In the former case (Fig. 1b), neighboring-element boundaries do not entirely coincide; in the latter, the polynomial expansion degree p in neighboring elements differs. Several SEM researchers have adopted a method that simultaneously alters element size h and configuration (h-refinement) *and* the polynomial degree p across neighboring elements (p-refinement), providing for a so-called h–p-refinement strategy. The *mortar element method* (MEM) [1,4,10,26] variationally minimizes the Lebesgue $\mathbb{L}_2$ norms of the discontinuities across non-conforming-element boundaries. MEM has been shown to produce optimal convergence in solving the incompressible Stokes equation [3], and has been demonstrated experimentally to produce excellent results when used as a basis for DARe in 1D [27].

Non-conforming h–p (not always *dynamic*) adaptive MEMs have been developed for studying turbulence [17,18], ocean simulation [20,25], flame front deformation [12], electromagnetic scattering [23], wave propagation [6], seismology [7] and other topics. However, MEM for p-type refinement has been cited as sometimes causing instability [30]. Also, in most flows of interest to us, it is the nonlinear interaction of the different scales 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 we restrict ourselves to a non-conforming fixed-p, h-refinement strategy only and use an *interpolation-based* scheme to maintain continuity between non-conforming elements. This method [13,24] is akin to the formulation developed in [5]; however, the latter deals with functionally non-conforming elements, while the former relates to the geometrically non-conforming elements of interest here. We contrast this choice with other familiar DARe codes (e.g., [10]), which, while object-oriented, uses the MEM as the basis of its dynamic adaptivity, but does not accommodate h-refinement. While the interpolation-based matching scheme has been widely used for functionally non-conforming meshes, to the best of our knowledge, our implementation of it in the context of fully dynamic adaptivity is unique and new.

2.2. Discretization of a nonlinearly coupled dynamical PDE system

In order to focus on DARe methodology, we concentrate on the simplest nonlinearly coupled PDE system that encompasses many of the difficulties in simulating fluid turbulence. Thus we discretize the 2D Burgers equation, presenting in turn the spatial operators and the time discretizations. These sections are in part a review of well-established methods but also provide implementation details unavailable elsewhere, and enable us to discuss code design motivations.

The equation considered in this work is the 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 $\vec{c}$ may be $\vec{u}$ (so that (1) is the Burgers equation), or $\vec{c} = \vec{c}(t)$ (a prescribed uniform linear-advection velocity) and $\nu \propto Re^{-1}$ is the kinematic viscosity. This is to be solved in a spatio-temporal domain $(\vec{x},t) \in \mathbb{D}\times]0,t_{\mathrm{f}}]$ subject to the boundary and initial conditions

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

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

2.2.1. Variational approach to spatial discretization

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

$$\langle\vec{v},\partial_t\vec{u}\rangle + \langle\vec{v},\mathscr{C}\vec{u}\rangle = -\nu\langle\overset{\rightrightarrows}{\nabla v}^{\mathrm{T}},\overset{\rightrightarrows}{\nabla u}\rangle, \tag{4}$$

where $\mathscr{C} := \vec{c}\cdot\vec{\nabla}$ is the advection operator and the inner product is (A.8). (See the appendix for the complete mathematical details.) The treatment of (3) will not be made explicit but may be easily inferred from our general discussion.

Assume that $\bar{\mathbb{D}}$ can be partitioned as in Table A.1. Adopt a Gauss–Lobatto–Legendre (GLL) basis, that is, expand u^μ and v^μ using (A.6). Inserting these expansions into (4), we arrive at the semi-discrete ODE system problem: Find the numerical solution $\vec{u}_n(\cdot,t) = \vec{\boldsymbol{\phi}}^{\mathrm{T}}\boldsymbol{u}(t) \in \mathscr{P}_{\boldsymbol{h},\vec{p}}\mathbb{U}_{\vec{b}}$ such that for all $\vec{v} = \vec{\boldsymbol{\phi}}^{\mathrm{T}}\boldsymbol{v} \in \mathscr{P}_{\boldsymbol{h},\vec{p}}\mathbb{U}_{\vec{0}}$,

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

collocated at $K(p+1)^d$ mapped Lagrange node points (Table A.1), 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, Chapter 6]) and diffusion matrix, respectively. The respective $d(p+1)^d$-square matrix blocks for element $\mathbb{E}_k$ are formulated in Appendix A.

Note that after assembly as discussed in Section 2.4, (5) must hold for the restriction $\vec{v}|_{\mathbb{E}_k} = \vec{\boldsymbol{\phi}}_k^{\mathrm{T}}\boldsymbol{v}_k$ of $\vec{v}$ to the kth element $\mathbb{E}_k$, so that a coupled ODE system for $\vec{u}_n|_{\mathbb{E}_k} = \vec{\boldsymbol{\phi}}_k^{\mathrm{T}}\boldsymbol{u}_k$ would in an assembled state be

$$\mathbf{M}_k\frac{\mathrm{d}\boldsymbol{u}_k}{\mathrm{d}t} + \mathbf{C}_k\boldsymbol{u}_k = -\nu\mathbf{L}_k\boldsymbol{u}_k. \tag{6}$$

Assembly guarantees continuity of $\vec{u}_n$ across all elements, which in turn is sufficient to keep $u_n^\mu \in \mathbb{H}^1(\mathbb{D})$. There are conforming and non-conforming element configurations, as illustrated in Fig. 1, and an interpolation-based scheme to enforce continuity along a non-conforming interface is the subject of Section 2.4. (Throughout the remainder of this paper "non-conforming" will refer to geometrically non-conforming elements, keeping the polynomial degree p fixed in all elements.)

2.2.2. Semi-implicit multistep time discretization

GASpAR employs semi-implicit multistep time discretization schemes. The diffusion is always solved fully implicitly, the time derivative is approximated using a backward-difference formula (BDF) of order M_{bdf} [9,21] and the advection term is approximated by an explicit extrapolation-based method (Ext) of order M_{ext} [22]. Then the integral of (6) from t^{n-1} to t^n is approximated by

$$\mathbf{H}_k^n \boldsymbol{u}_k^n = \sum_{m=n-M_{\mathrm{bdf}}}^{n-1} \beta_{\mathrm{bdf}}^{m,n} \mathbf{M}_k^m \boldsymbol{u}_k^m - \sum_{m=n-M_{\mathrm{ext}}}^{n-1} \beta_{\mathrm{ext}}^{m,n} \mathbf{C}_k^m \boldsymbol{u}_k^m, \tag{7}$$

where

$$\mathbf{H}_k^n := \beta_{\mathrm{bdf}}^{n,n} \mathbf{M}_k^n + \nu \mathbf{L}_k^n \tag{8}$$

is the spectral-element Helmholtz matrix. Although the matrices $\mathbf{L}_k$ and $\mathbf{M}_k$ in (6) were t-independent, they are time-indexed in (7) and (8) because DARe will, in general, reconfigure the partition (Table A.1) over time. For this reason, the coefficients $\beta^{m,n}$ are re-computed for each t^n after a reconfiguration, as in the traditional schemes cited, except that the timestep Δt^m may vary with m as the smallest spectral-element diameter $h^m := \min_k h_k^m$ (Table A.1) changes. The accuracy of solving (7) follows from many known SEM error estimates, e.g., for the Helmholtz problem on conforming meshes [21, Section 2.3.6] or the Poisson problem on non-conforming meshes [21, Section 5.5.2.1]. In Section 2.5, the solution of (7) is explained.

2.3. Implications for code design

The fully discretized advection–diffusion equation (7) brings up several issues impinging on code design. First, all 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, solution data must be available at multiple times t^m, so this information is provided in a data structure. Thus arise both *element* and *field* objects. The former contains all d-dimensional mesh information, including the Gauss-quadrature nodes and weights (Table A.1). The *element* object also contains neighbor-list information and the hierarchical element refinement level $\propto -\log_2 h_k$ of each element $\mathbb{E}_k$. The *field* object contains the data $\boldsymbol{u}^m$ quantifying the physical system of interest at each t^m.

The 1D basis functions, the derivative matrices and Gauss-quadrature nodes and weights (Table A.1) are encapsulated in *basis* classes (objects), and the 1D matrices such as ((A.9), (A.10) and (8)) are objects that contain pointers to the basis objects and to a local element object. Generally d-dimensional SEM matrices are not constructed but are applied using 1D tensor–product matrix factors. High-level objects encapsulate the solution of (6) or other 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; they are constructed using linked lists of elements, fields and multidimensional SEM objects that depend only on the underlying mesh. Hence, the classes that handle DARe and enforce continuity between elements are independent of the system being solved.

2.4. Continuity and global assembly of non-conforming elements

Conforming discretizations enforce continuity simply by assigning the same weighted-averaged $\vec{u}_n$ values to the coinciding node points $\vec{x}_{\vec{j},k} = \vec{x}_{\vec{j}',k'}$ along element edges $\partial\mathbb{E}_{k,s} = \partial\mathbb{E}_{k',s'}$ (Fig. 1a). This matching condition 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 $d(p+1)^d$-vectors $\boldsymbol{u}_k$, $k \in \{1,\ldots,K\}$. Generally $N_{\mathrm{g}} < Kd(p+1)^d$. This expression is accomplished by using a $Kd(p+1)^d \times N_{\mathrm{g}}$ Boolean assembly matrix $\mathbf{A}_{\mathrm{c}}$ (also called a *scatter* matrix):

$$\boldsymbol{u} = \mathbf{A}_{\mathrm{c}} \boldsymbol{u}_{\mathrm{g}}. \tag{9}$$

The transpose $\mathbf{A}_{\mathrm{c}}^{\mathrm{T}}$ performs the *gather* operation associated with the $\mathbf{A}_{\mathrm{c}}$ scatter. In practice, $\mathbf{A}_{\mathrm{c}}$ is never formed explicitly but is instead *applied.*

In the non-conforming case $\partial\mathbb{E}_{k,s} \subsetneq \partial\mathbb{E}_{k',s'}$ and most boundary-node points are not coinciding (Fig. 1b). In the present work, unlike in MEM, the interface matching does not alter the underlying function space $\mathbb{U}_{\vec{b}}$ (Section

2.2). To illustrate, consider the non-conforming mesh in Fig. 1b. For the moment denote the global nodes, those nodes residing on the east *parent* edge $\partial\mathbb{E}_{1,1}$, by $\vec{x}_{g,i},\ i \in \{2,5,8\}$, and denote the nodes on the west *child* edges, $\partial\mathbb{E}_{2,3}$ and $\partial\mathbb{E}_{3,3}$ by $\vec{x}_j,\ j \in \{9,12,15,18,21,24\}$. A globally continuous function can always be found in $\mathbb{U}_{\vec{0}}$ in a proper subspace of the span of globally discontinuous functions $\phi_j(\vec{x})$ that interpolate from the local nodes $\vec{x}_j$. Therefore the weak formulation of (1) implies functions $\phi_{g,i}(\vec{x})$ exist that are globally continuous across $\mathbb{D}$, span $\mathbb{U}_{\vec{0}}$, and interpolate from the global nodes $\vec{x}_{g,i}$. Therefore the matrix $\mathbf{A}$, that generalizes the Boolean scatter matrix $\mathbf{A}_c$ used in the conforming-element formulation, can be conceived as having entries $\phi_{g,i}(\vec{x}_j)$, and accommodates both conforming *and* non-conforming elements. It is convenient to factor $\mathbf{A} = \mathbf{\Phi}\mathbf{A}_c$, where $\mathbf{\Phi}$ is the interpolation matrix from global to local d.o.f. and $\mathbf{A}_c$ is locally conforming. Another illustration appears in [31, Eqs. (14)–(16)].

To accommodate Dirichlet boundary conditions (2) into the solution, we employ a *masking projection* Π, which is 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 $\vec{u} = \vec{u}_h + \vec{u}_b$, where $\boldsymbol{u}_h := \mathbf{\Phi}\Pi\mathbf{A}_c\boldsymbol{u}_g$ constructs the projection $\vec{u}_h := \vec{\boldsymbol{\phi}}^{\mathrm{T}}\boldsymbol{u}_h \in \mathbb{U}_{\vec{0}}$ of $\vec{u}$, that is, its homogeneous part, and $\boldsymbol{u}_b := \boldsymbol{u} - \boldsymbol{u}_h$ constructs $\vec{u}_b \in \mathbb{U}_{\vec{0}}$, which vanishes at the interior nodes $\vec{x}_{\jmath,k} \in \mathbb{D} \setminus \partial\mathbb{D}$. Inserting this analysis into (5) (noting $\vec{v} \in \mathbb{U}_{\vec{0}} \ \Rightarrow\ \boldsymbol{v} = \mathbf{\Phi}\Pi\mathbf{A}_c\boldsymbol{v}_g$) and repeating the time discretization leading to (7), we arrive at the following linear equation to solve for $\boldsymbol{u}_g$ at each timestep:

$$\boldsymbol{v}^{\mathrm{T}}\mathbf{H}\boldsymbol{u} = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{f} \quad \forall \boldsymbol{v}_g \ \Rightarrow\ \mathbf{A}_c^{\mathrm{T}}\Pi\mathbf{\Phi}^{\mathrm{T}}\mathbf{H}\mathbf{\Phi}\Pi\mathbf{A}_c\boldsymbol{u}_g = \mathbf{A}_c^{\mathrm{T}}\Pi\mathbf{\Phi}^{\mathrm{T}}(\boldsymbol{f} - \mathbf{H}\boldsymbol{u}_b), \tag{10}$$

where $\mathbf{H} := \mathrm{diag}_k\mathbf{H}_k$ is symmetric positive-definite (8) and we have denoted all past-time terms from time-derivative expansion and advection in (7) by $\boldsymbol{f}$. The preconditioned conjugate-gradient (PCG, [32,36]) algorithm is used to solve (10). While (10) shows explicitly that the l.h.s. matrix is symmetric non-negative-definite, it is not in a form easily solved in parallel. Left-multiplying (10) by $\mathbf{\Phi}\Pi\mathbf{A}_c$, we get the following local problem to solve for $\boldsymbol{u}_h$:

$$\Sigma\mathbf{H}\boldsymbol{u}_h = \Sigma(\boldsymbol{f} - \mathbf{H}\boldsymbol{u}_b), \quad \text{where } \Sigma := \mathbf{\Phi}\Pi\mathbf{A}_c\mathbf{A}_c^{\mathrm{T}}\Pi\mathbf{\Phi}^{\mathrm{T}}. \tag{11}$$

The *direct stiffness summation* (DSS) matrix Σ is coded so that the gather and scatter are performed in one operation (Section 2.6.3), which reduces parallel communication overhead [34].

Two other operators must be introduced that help maintain $\mathbb{H}^1(\mathbb{D})$ continuity. The inverse *multiplicity matrix* $\mathbf{W}$ is diagonal, computed by initializing a collocated vector $g^{\mu}_{\jmath,k} = 1\ \forall \jmath, k, \mu$, setting child boundary nodes to 0, performing $\boldsymbol{g} \leftarrow \mathbf{\Phi}\mathbf{A}_c\mathbf{A}_c^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\boldsymbol{g}$, then setting

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

For example, corresponding to Fig. 1a and b the diagonals of $\mathbf{W}$ are

$$(1,1,\tfrac{1}{2},1,1,\tfrac{1}{2},1,1,\tfrac{1}{2},\tfrac{1}{2},1,1,\tfrac{1}{2},1,1,\tfrac{1}{2},1,1) \quad \text{and} \quad (1,1,1,1,1,1,1,1,1,0,1,1,0,1,1,0,\tfrac{1}{2},\tfrac{1}{2},0,\tfrac{1}{2},\tfrac{1}{2},0,1,1,0,1,1), \tag{12}$$

respectively. After a DSS operation (11) the true global d.o.f., nodes 2, 5, and 8, carry all the information held by nodes 9, 12, 15, 18, 21, and 24, so for the purpose of the PCG solve the latter give zero $\mathbf{W}$ entries in (12). Given that global inner products in the PCG solve are collected from local contributions from each element (i.e., Table 1, the lines involving $\mathbf{W}$), the $\mathbf{W}$ zeros prevent double counting when computing these products, and prevent non-global d.o.f. (e.g., child edge nodes) from contributing. Note also that in Fig. 1b, the $\mathbf{W}$ entries for nodes 17 and 20 have value 1/2, as expected for nodes such as these that lie on conforming edges. The $\mathbb{H}^1$ "smoothing" operation in the PCG algorithm also uses $\mathbf{W}$. In smoothing, we have that $\bar{\boldsymbol{g}} = \mathbf{S}\boldsymbol{g}$, where $\mathbf{S} := \mathbf{\Phi}\Pi\mathbf{A}_c\mathbf{A}_c^{\mathrm{T}}\mathbf{W}$. Smoothing acts only on quantities all of whose d.o.f. have already been distributed to global d.o.f. using DSS. The result of smoothing is a quantity that is interpolated properly to the child edges and that is expressed without multiple counting at multiple local nodes that represent the same physical location. The $\mathbf{W}$ matrix weights the operand $\boldsymbol{g}$ so that the respective sums on the parent (global) edge nodes (nodes 2, 5, and 8 in the case above) contribute to the result $\bar{\boldsymbol{g}}$ just once each, and the child edge nodes receive their $\bar{\boldsymbol{g}}$ values from the parent edge nodes by interpolation.

Table 1
PCG algorithm modified for non-conforming element meshes

```
$\boldsymbol{u}_\mathrm{h} = \mathbf{0}$                                        // initialize homogeneous term
$\boldsymbol{r} = \Sigma(\boldsymbol{f} - \mathsf{HS}\boldsymbol{u}_\mathrm{b})$                       // initialize residual
$\boldsymbol{w} = \mathbf{0}$                                         // initialize search vector
$\rho_1 = 1$                                          // initialize parameter
while not converged:
    $\boldsymbol{e} = \mathsf{SP}^{-1}\boldsymbol{r}$                                // error estimate
    $\rho_0 = \rho_1,\ \rho_1 = \boldsymbol{r}^\mathrm{T}\mathsf{W}\boldsymbol{e}$                     // update parameters
    $\boldsymbol{w} \leftarrow \boldsymbol{e} + \boldsymbol{w}\rho_1/\rho_0$                          // increment search vector
    $\boldsymbol{r}' = \Sigma\mathsf{H}\boldsymbol{w}$                                 // image of $\boldsymbol{w}$
    $\alpha = \rho_1/\boldsymbol{w}^\mathrm{T}\mathsf{W}\boldsymbol{r}'$                           // component of $\boldsymbol{u}_\mathrm{h}$ increment
    $\boldsymbol{u}_\mathrm{h} \leftarrow \boldsymbol{u}_\mathrm{h} + \alpha\boldsymbol{w}$                            // increment $\boldsymbol{u}_\mathrm{h}$ along $\boldsymbol{w}$
    $\boldsymbol{r} \leftarrow \boldsymbol{r} - \alpha\boldsymbol{r}'$                              // increment residual
end
$\boldsymbol{u} = \mathsf{S}_\mathrm{f}(\boldsymbol{u}_\mathrm{h} + \boldsymbol{u}_\mathrm{b})$.
```

2.5. *Modified preconditioned conjugate-gradient algorithm*

It is important to modify the well-known PCG algorithm in order to solve (11) in the non-conforming case. The modifications 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 Σ matrix (11) 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 using the smoothing operator, S (cf. Section 2.4). Note that it is critical that the inhomogeneous boundary term $\vec{u}_\mathrm{b}$ belong to $\mathbb{H}^1(\mathbb{D})^d$ in (11); thus, the smoothing matrix S is applied to $\boldsymbol{u}_\mathrm{b}$ before H is. However, the non-smoothed 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 matrix $\mathsf{S}_\mathrm{f} := \boldsymbol{\Phi}\mathsf{A}_\mathrm{c}\mathsf{A}_\mathrm{c}^\mathrm{T}\mathsf{W}$.

With these considerations we present in Table 1 the PCG algorithm for the assembled local problem (11) modified from the conforming-elements case, here for non-conforming elements. Preconditioning is handled by the matrix P^{-1}. GASpAR includes block- and point-Jacobi preconditioners. For the test problems presented in Section 3, a point Jacobi preconditioner has proven to be adequate. In general, the preconditioned quantity must be smoothed, as indicated in Table 1.

2.6. *Adaptive mesh formulation*

2.6.1. *Element-mesh hierarchical configuration*

We now employ non-conforming connectivity to carry out dynamic adaptivity. Recall that the global domain $\mathbb{D}$ is initially covered (Table A.1) by a set of disjoint (non-overlapping) elements $\mathbb{E}_k$. Each of these initial elements becomes a tree *root* element, identified by a unique root *key* k_r for that tree. At each level $\ell \in \{\ell_\mathrm{min}, \ldots, \ell_\mathrm{max}\}$, 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

$$\ell = \lfloor \log_{2^d}(k/k_\mathrm{r}) \rfloor. \tag{13}$$

In order to ensure all keys are unique, the first $k_\mathrm{r} := 1$ and the next is $k'_\mathrm{r} := 2^{d\ell_\mathrm{max}}(k_\mathrm{r}+1)$, 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 $\mathbb{E}_{2^dk}, \ldots, \mathbb{E}_{2^d(k+1)-1}$ at level $\ell + 1$ to replace each $\mathbb{E}_k$, or else *coarsening* 2^d existing children $\mathbb{E}_k, \ldots, \mathbb{E}_{k+2^d-1}$ 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 ($\mathbf{A}^{\mathrm{T}}\boldsymbol{u}$ segments) or perform scatters ($\mathbf{A}\boldsymbol{u}_{\mathrm{g}}$ segments); 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 $s \in \{0,\ldots,2^d-1\}$ 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 (i.e., excluding the vertices) 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. 1b as a paradigm, in general the mortars contain node locations and the basis functions of the parent element boundary (edge in 2D, or face in 3D). The mortar structures represent the same field information for the parent and child edges; their nodes coincide with the nodes of the parent edge, 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 non-conforming. This communication uses a *voxel database* (VDB) [17]. A VDB consists of records containing geometric point locations, a component id that tells what part of the element $\mathbb{E}_k$ (in 2D, 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, the root id of that element, 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 geometrical extent of the mortar. 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 DARe operate only on the element and field lists. The SEM solvers adjust themselves automatically to accommodate the dynamic addition and removal of elements that occurs as a result of DARe.

2.6.2. Refinement and coarsening rules

The refinement and coarsening method takes as input only the local indexes of the elements to be refined and coarsened. Before refinement or coarsening is done, the tagged elements are checked for compliance with several rules. For refinement, the rules are: (R1) the refinement level must not exceed a specified limit $\ell_{\max}$ and (R2) at most one level may separate neighbor elements. Rule R2 must be followed also for interfaces at periodic boundaries. Rule R2 is enforced by tagging a coarse element for refinement too, if it has an already refined neighbor tagged for further refinement. Enforcement of R1 and R2 is most easily effected by building a global list of keys of all elements tagged for refinement, and comparing the local refinement lists with it.

We may not coarsen an element under any of the conditions: (C1) it is a root; (C2) any of its 2^d-1 siblings are not tagged for coarsening; (C3) it appears in a refinement list; or (C4) rule R2 would be violated. To enforce C4, we use a *query-list*, i.e., a global list of each element key k, its parent key $\lfloor k/2^d \rfloor$, and its level ℓ (13). The query-list contains keys gathered from all processors. The following procedure is then used.

(1) Build a global "refinement" query-list (RQL) from the keys in the local refinement list.
(2) Find level limits $\ell_{\max}$ and $\ell_{\min}$ from the coarsen list.
(3) Reorder the current local coarsen list from $\ell_{\max}$ down to $\ell_{\min}$.
(4) Looping from $\ell = \ell_{\max}$ down to $\ell_{\min}$: build a global "coarsen" query-list (CQL) from the keys in the current local coarsen list; and for all keys k in the local coarsen list at the current ℓ, if any refined neighbor is in the CQL and no refined neighbors are in the RQL, then k is retained in the current coarsen list; otherwise it is deleted.
(5) Check finally that all elements in the local coarsen list have all their siblings also tagged for coarsening. The sibling elements of k are identified by having the same parent key $\lfloor k/2^d \rfloor$.

Note that the local refinement lists are checked and possibly modified *before* checking and modifying the coarsen lists.

2.6.3. Communicating boundary data

The mortar data structures contain all the data to be communicated between elements during each application of the DSS Σ (11) or smoothing operation $\mathbf{S}$. Communication of element-boundary data requires network communication on parallel computers. This involves *initialization* and *operation* steps. Initialization establishes element-processor connectivity by bin-sorting global node indexes and having each processor examine the nodes from one bin, to determine element-neighbor lists. This method has been suggested in [9, Section 8.5.2] but to our knowledge has never before been implemented. All coinciding mortar-structure nodes $\vec{x}_{g,i} = \vec{x}_{g,i'}$ are uniquely labeled by their Morton index $M(\vec{x}_{g,i})$, computed by digitizing the d coordinates and partially interleaving the B bits along each coordinate μ. So for $a^\mu := \min_{\vec{x}\in\mathbb{D}} x^\mu$:

$$M^\mu(\vec{x}) := \left\lfloor \frac{x^\mu - a^\mu}{\Delta x} + \frac{1}{2} \right\rfloor \quad \Rightarrow \quad M(\vec{x}) := \sum_{\mu=1}^{d} 2^{(\mu-1)B} M^\mu(\vec{x}) \in \left\{0,\ldots,2^{dB}-1\right\},$$

where Δx is chosen so that $M^\mu(\vec{x}) \in \left\{0,\ldots,2^B-1\right\} \forall \vec{x}$. For P processors, a collection of P bins $\mathbb{B}_l$, $l \in \{0,\ldots,P-1\}$, is generated that partitions the dynamic range (over all processors) of the Morton indexes. Processor l partitions its list of indexes into the bins, sending the contents of $\mathbb{B}_{l'}$ to processor l', where the information is combined with those from other processors and then sent back to processor l. After this initialization step, every processor is informed of which other processors share which mortar nodes. The operation step communicates the data at any node point $\vec{x}_{g,i}$ with all other processors that share it. These data are extracted from the containing element by using the pointer indirection provided by $M(\vec{x}_{g,i})$. The field values at $\vec{x}_{g,i}$ are summed during DSS or smoothing and reassigned at $\vec{x}_{g,i}$ also by indirection. To reduce communication, shared $\vec{x}_{g,i}$ residing on the same processor are summed before being transmitted to the other processors that share the $\vec{x}_{g,i}$. At the end of the operation step, the field values at multiply-represented global nodes are identical. This gather–scatter procedure ensures that the DSS output are locally available immediately after communication. One benefit of this gather–scatter method is that it allows communication to be separated from the geometry, because Morton indexes are essentially unstructured lists of local data locations. However, a future upgrade of GASpAR will use VDBs to obviate the need for the bin-sort initialization step, which requires information already provided in the VDBs.

2.6.4. Error estimators

Elements are tagged for DARe by the use of an a posteriori criterion. The *spectral estimator* criterion, modified from [18,27], uses local Legendre spectra to estimate the quadrature and truncation errors and the spectral convergence rate in each element $\bar{\mathbb{E}}_k = \vec{\vartheta}_k([-1,1]^d)$. First, the mapping $u^\mu \circ \vec{\vartheta}_k(\vec{\xi})$ of each solution component $u^\mu(\vec{x})$ is transformed to spectral coefficients $u_j^{\mu,\mu'}$ along a 1D line in coordinate $\xi^{\mu'}$ by [9, (B.3.13)], averaging over all the $\xi^{\mu''\neq\mu'}$. The convergence rates $\lambda^{\mu,\mu'}$ are fit using $|u_j^{\mu,\mu'}| \approx C^{\mu,\mu'} \mathrm{e}^{-\lambda^{\mu,\mu'} j}$ [18, (18)] with $j \in \{p-3,\ldots,p\}$, except that instead of "equivalent" 1D coefficients [18, (17)], we combine fits using $\lambda^\mu := \min_{\mu'=1}^{d} \lambda^{\mu,\mu'}$. The solution error $\varepsilon_{\mathrm{est}}^\mu$ is estimated using [18, (19) l.h.s.], except again instead of "equivalent" 1D coefficients, we estimate the first term of [18, (19)] by $\sum_{\mu'=1}^{d} (u_p^{\mu,\mu'})^2$ and the second term by $(\prod_{\mu'=1}^{d} (C^{\mu,\mu'})^2 \int_{p+1}^{\infty} \mathrm{d}j\, \mathrm{e}^{-2\lambda^{\mu,\mu'} j})^{1/d}$. Thus, $\bar{\mathbb{E}}_k$ is *refined*, if for some μ, $\varepsilon_{\mathrm{est}}^\mu$ is above a threshold value ε_{t} or if λ^μ is below another threshold λ_{t}. For *coarsening*, for all μ, all 2^d sibling elements must have their $\varepsilon_{\mathrm{est}}^\mu$s below some value $\gamma_{\mathrm{c}}\varepsilon_{\mathrm{t}} < \varepsilon_{\mathrm{t}}$, computed by multiplying by a "coarsening multiplier" γ_{c}. This prevents "blinking", i.e., refined elements being immediately coarsened again. In conjunction with the spectral estimator, we can often obtain better overall accuracy convergence by thresholding on the $\bar{\mathbb{E}}_k$-maximum second derivative magnitude in any coordinate and taking a logical `OR` of that criterion with the spectral estimator. While the high polynomial degrees will help the spectral estimator, given the variety of our future applications, new refinement criteria may be more effective. The investigation of refinement criteria appropriate, e.g., for intermittent features is a major outstanding problem in adaptive numerical solution of PDEs that we will consider in future work.

3. Results for adaptive (non)linear advection–diffusion simulation

Our test problems examine various aspects of (1). The primary goal is to investigate the solution temporal and spatial convergence when adaption is used. Thus we have selected problems with analytic solutions, so that errors may be determined *exactly*, instead of only by comparison, e.g., to a uniformly highly refined control solution. Tests begin with the simplest aspect of (1) and progress through more difficult problems until the behavior of the full 2D nonlinear, multi-component version of (1) is considered. We do not use filtering for any of these test problems.

For each test the BDF3 and Ext3 schemes are used for the time-derivative and the advection terms in (7), respectively, unless stated otherwise. This requires that all the required time levels t^{m-1} be initialized, $m \in \{1,\ldots,\max(M_{\text{bdf}}, M_{\text{ext}})\}$. A logical `OR` of the spectral and second-derivative error estimators or just the second-derivative estimator is used for the adaption criterion. The spectral estimator is normalized by the initial-condition norm $\|\vec{u}^0\|_\infty$, and the second derivative is normalized by $\|\vec{u}^0\|_\infty/L^2$, where L is the longest global domain length. The threshold λ_t is always set at 1 when used.

Except where we compare with published results, the viscosities are somewhat arbitrary. We reiterate that one of our motivations in considering (1) is that it exemplifies many of the characteristics of the Navier–Stokes equations of interest in simulating turbulence, including the dependence on ν via *Re*. However, we note that a recent paper [31] concludes that the MEM and the interpolation-based connectivity for non-conforming elements may manifest inconsistencies that affect convergence, which a small viscosity can prevent.

For the purposes of our tests, we perform adaption after every 10 timesteps except if stated otherwise. In practice, this is not optimal as the adaptivity overhead can overtake the computational savings achieved by reducing the required number of d.o.f. In general, it is more meaningful and efficient to adapt at a fraction of a fiducial timescale, say an eddy turnover time. The refinement criteria are applied to each component of (1) that is solved for.

In order to compare an adaptive solution, we use an *ℓ-control* grid. This is a grid that uniformly covers the domain with elements at the finest resolution $\ell_{\max} = \ell$. For all spatial convergence tests that have control solutions, we will also provide a single processor speed-up factor representative of the adaptive solutions, by giving the ratio $T_{\text{control}}/T_{\text{adaptive}}$ of the total control and adaptive cpu run times. Naturally, this factor is only to be used for reference since the speed-up will, in general, depend not only on the solution and its refinement criteria and thresholds, but also the adaption interval, and expansion degree, p.

3.1. Adaptive heat-equation solution results

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

$$u_{\mathrm{a}}^{\mu}(\vec{x},t) := \frac{\sigma(0)^d}{\sigma(t)^d} \sum_{\imath^1,\ldots,\imath^d=-\infty}^{\infty} \exp -\left(\frac{\vec{x}-\vec{x}^0+\vec{\imath}-\int_0^t \vec{c}(t')\,\mathrm{d}t'}{\sigma(t)}\right)^2 \tag{14}$$

for $t > -\sigma(0)^2/4\nu$ ($u_{\mathrm{a}}^{\mu}(\vec{x},t) := 0$ otherwise), where $\sigma(t) := \sqrt{\sigma(0)^2 + 4\nu t}$, $\sigma(0) = \sqrt{2}/20$ is the initial e-folding width, $\nu = 0.1$ and $\vec{x}^0 = \sum_{\mu=1}^{d} \vec{e}^{\mu}/2$ is the initial peak location. To compute (14), we truncate summands of value less than 10^{-18} of the partial sum. The simplest version of (1) is the heat equation, where $\vec{c} = \vec{0}$. The goal here is to determine the temporal and spatial convergence when there is no advection. The initial condition (3) is computed on $K = 4 \times 4$ elements from (14) at $t = 0$ and $d = 2$, and the mesh is refined until refinement level $\leqslant \ell_{\max}$. Both the spectral estimator with threshold $\varepsilon_t = 10^{-3}$ and second-derivative estimator with threshold of 0.25 were used. The coarsening multipliers (to prevent blinking) for each were set to $\gamma_c = 0.5$ and 0.25, respectively. A BDF2 scheme is used here for the time derivative.

3.1.1. Temporal convergence of the adaptive heat-equation solution

We examine time convergence by advancing to $t_f = 0.05$ for various constant Δt. From (14) curves of relative $\mathbb{L}_2$ error $\varepsilon = \|\vec{u}_{\mathrm{n}} - \vec{u}_{\mathrm{a}}\|_2/\|\vec{u}_{\mathrm{a}}^0\|_2$ vs. Δt are plotted for several maximum-refinement levels $\ell_{\max}$ and for degrees p, in Fig. 2a–d. The control grid here consists of 16×16 elements. The BDF2 and Ext2 are globally second-order schemes, so if the solution is well resolved spatially, we expect to find a slope of ≈2 in a log–log plot of error vs. Δt. Indeed this is seen in Fig. 2a–d; each panel shows a sequence of three curves for the refinement levels $\ell_{\max} \in \{0,\ldots,2\}$, where $\ell_{\max} = 0$ implies that no refinement is done. For the curves that are spatially resolved, the error is linear with slope 2.04. Even at low p, the solution is well resolved if DARe is used, even at $\ell_{\max} = 1$. If the refinement thresholds ε_t where increased slightly, we would see a larger reduction in the

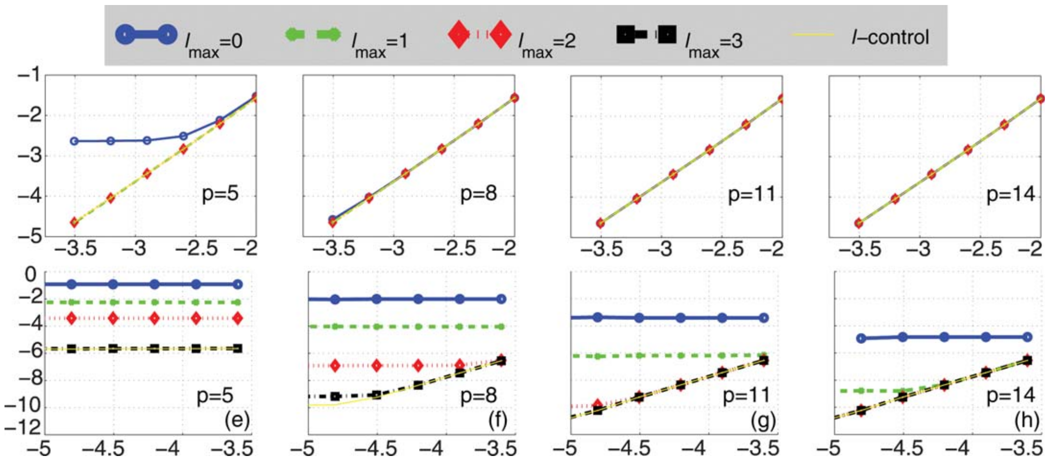

Fig. 2. Plots of normalized error $\log_{10}(\|\vec{u}_n - \vec{u}_a\|_2/\|\vec{u}_a^0\|_2)$ vs. $\log_{10}\Delta t$ for (a–d) the heat equation and (e–h) advection-dominated flow (1), for different polynomial degrees p as labeled. Each upper panel shows curves for up to three maximum refinement levels $\ell_{\max}$ indicated in the legend; each lower panel shows four maximum refinement levels. For the heat equation, The 2-control solutions (thin curves) overlie the $\ell_{\max} \geqslant 1$ adaptive curves. As p increases, the curves converge.

number of d.o.f. required, but our accuracy would decrease, requiring a higher $\ell_{\max}$ before accuracy (at small Δt) is restored. As p increases, there is less need for DARe, as is expected due to the smoothness of the solution.

3.1.2. Spatial convergence of the adaptive heat-equation solution

We now consider the effects of polynomial degree p. The maximum refinement is fixed at $\ell_{\max} = 2$. At time t^n a dynamic Courant-limited timestep

$$\Delta t^n \leqslant Co/ \max_{\vec{j}\in\{1,\ldots,p\}^d;k\in\{1,\ldots,K^n\};\mu,\mu'\in\{1,\ldots,d\}} \left(\frac{4\nu}{(\Delta^n_{\vec{j},k})^2} + \frac{|u^{\mu' n}_{\vec{j}-\vec{e}^\mu,k} + u^{\mu' n}_{\vec{j},k}|}{2\Delta^n_{\vec{j},k}} \right) \tag{15}$$

is used with a fixed Courant number $Co = 1.0$, where $\Delta^n_{\vec{j},k} := \min_{\mu\in\{1,\ldots,d\}} |\vartheta^{\mu n}_k(\vec{\xi}_{\vec{j}-\vec{e}^\mu}) - \vartheta^{\mu n}_k(\vec{\xi}_{\vec{j}})|$ (Table A.1). We can set Co to a reasonably high value because a semi-implicit scheme is used. The solution is advanced to $t_f = 0.5$, enough to observe the solution coarsening as it decays. Only the control runs use the variable timestep; the adaptive runs use as a fixed timestep the Courant-limited value of the corresponding control case at $t = t_f$. The initial mesh is the same as Section 3.1.1.

Fig. 3a shows the exponential spatial convergence characteristic of all our tests. We expect from (A.3) and Section 2.2.2 that an infinitely smooth solution will spectrally converge along a straight-line plot of $\log_{10}(\|\vec{u}_n - \vec{u}_a\|_2/\|\vec{u}_a^0\|_2)$ vs. p. For lower p, the 2-control solutions are better than the adaptive runs, but the curves merge quickly, as we would expect for such a smooth problem. The adaptive curves show some slight concavity for this problem. The low-p error source is likely the elliptic nature of (10), so that coarse elements propagate their error throughout the mesh. Fig. 4b shows that even for varying K (Fig. 4a), the error over time behaves monotonically, agreeing very closely with the control profile. We find that the adaptive cases for all but $p = 2$ case run significantly faster ($T_{\text{control}}/T_{\text{adaptive}} \approx 3$) than the controls for this problem.

3.2. Adaptive linear-advection simulation results

Next we consider the linear advection-dominated Eq. (1) with $d = 2$, $\nu = 10^{-4}$ and $\vec{c} = \vec{e}^1$. This tests the ability of the code to follow a localized translating distribution. The initial state (3) is given by (14) at $t = 0$. The spectral estimator in this problem is turned off. The second-derivative criterion is set to $\varepsilon_t = 1$ with a coarsening multiplier of $\gamma_c = 0.5$.

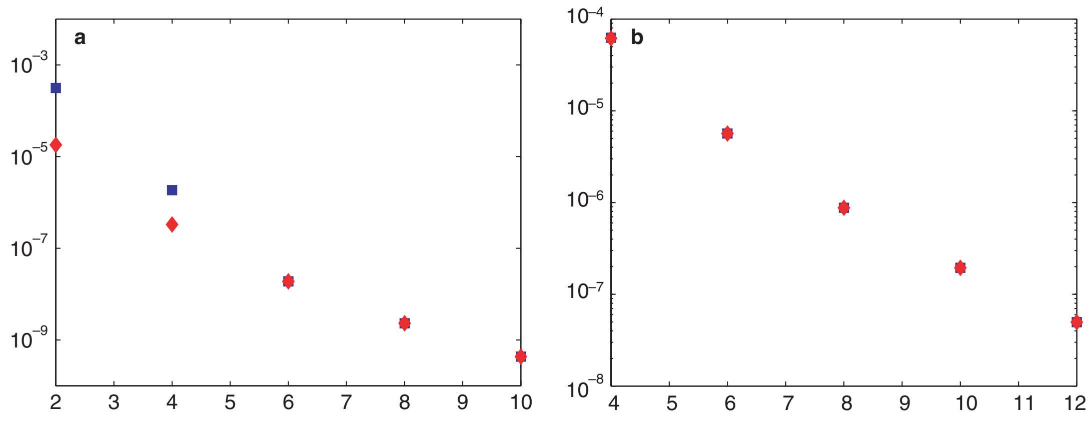

Fig. 3. Semilog plots of $\|\vec{u}_n - \vec{u}_a\|_2/\|\vec{u}_a^0\|_2$ vs. p for (a) the diffusion, (b) advection-dominated flow. Square and diamond markers indicate the adaptive and $\ell_{\max}$-control runs, respectively. For diffusion, $\ell_{\max} = 2$, and for the advection-dominated cases, $\ell_{\max} = 3$.

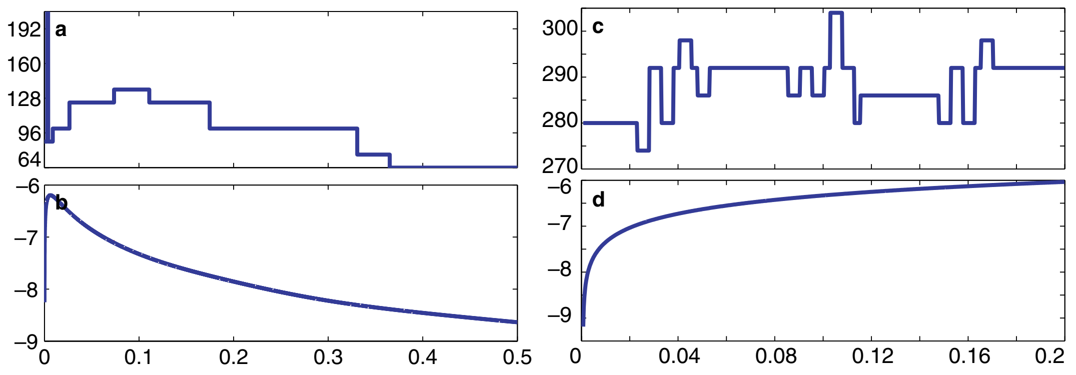

Fig. 4. For the 2D adaptive (a,b) $p = 6$ heat-equation, and (c,d) $p = 8$ linear advection tests, time series of (a,c) number K of elements, and (b,d) $\log_{10}(\|\vec{u}_n - \vec{u}_a\|_2/\|\vec{u}_a^0\|_2)$. The errors for the adaptive and control meshes lie on top of one another.

3.2.1. Temporal convergence for adaptive linear advection

Temporal convergence is tested as in Section 3.1.1, except that only the second-derivative criterion is used. The final $t_f = 0.06$, and we begin with a $K = 4 \times 4$ element mesh. We present the results in 2e–h. The spatially resolved curves in each plot have an average slope of 2.95. Even at high degree p, the error is Δt-independent for the unrefined mesh. For lower p, the error decays at the order of the time-stepping method only if there are several refinement levels, indicating that the solution is well resolved spatially only at higher $\ell_{\max}$. Thus, in order to achieve a temporal error $\mathcal{O}(\Delta t^3)$, refinement is necessary.

Fig. 2e–h also shows 3-control runs corresponding to the adaptive solutions, indicated by thin curves that all overlie the $\ell_{\max} = 3$ curves. As p increases, less refinement is required to achieve the same accuracy that 3-control does.

3.2.2. Spatial convergence for adaptive linear advection

We turn to the effects of polynomial degree p on the solution error. The maximum refinement level is fixed to $\ell_{\max} = 3$. Here, a Courant-limited timestep (15) is again used with $Co = 0.2$. The solution is advanced to $t_f = 0.2$, enough to see several DARe cycles occur (Fig. 4c). The initial mesh is the same as in Section 3.2.1. Spectral error decay can be seen in Fig. 3b, which also shows the 3-control solutions. The adaptive solution error decays nearly identically as does the 3-control, suggesting again that interpolation introduces no deleterious effects for this problem.

Fig. 4c–d shows typical time series of the element count K and the error. Clearly, adaptivity does not alter the monotonic error behavior. The 3-control grid ($K = 32 \times 32$ elements) error for $p = 8$ is plotted in Fig. 4d and is nearly identical to the adaptive error. Adaptivity clearly provides a significant savings in the number of d.o.f. required for a given accuracy. Indeed, the single-processor time savings is significant too; we find that $T_{\text{control}}/T_{\text{adaptive}} \approx 10$ for most p.

Note that when we set $\nu = 0$ for this problem, we obtain energy conservation to about six digits for the $\ell_{\max} = 3$ adaptive case, and to about seven digits in the $\ell_{\max} = 3$ control run, up to $t_f = L/|\vec{c}| = 1$.

3.3. 2D Burgers equation

We now examine the nonlinear ($\vec{c} = \vec{u}$) version of (1). The goal is to investigate the solution errors as the mesh resolves and tracks the stationary or propagating fronts generated and sustained by the nonlinear coupling of the system. We introduce a class of exact 2D solutions as follows. Note that any d solutions $q^\mu(y,t)$ to the 1D Burgers equation can be cast into d dimensions by substituting

$$\vec{u}(\vec{x},t) = \sum_{\mu=1}^{d} \vec{\kappa}^\mu q^\mu(\vec{\kappa}^\mu \cdot \vec{x}, \vec{\kappa}^\mu \cdot \vec{\kappa}^\mu t), \quad \text{where } \vec{\kappa}^\mu \cdot \vec{\kappa}^{\mu'} := \vec{\kappa}^\mu \cdot \vec{\kappa}^\mu \delta^{\mu,\mu'}, \tag{16}$$

into (1) [14]. If q^μ has period Y^μ w.r.t. y, then taking integer $2\kappa^{\mu,\mu'}/Y^\mu$ makes periodic boundary conditions for $\vec{x} \in [-1,1]^d$ appropriate. An initial condition (3) for a kind of straight $\vec{\kappa}^\mu$-perpendicular front is derived from

$$q^\mu(y,0) := -\sin(\pi y) + \hat{u}_2^\mu \sin(2\pi y). \tag{17}$$

The first problem is the classical Burgers stationary front, which is compared with and without adaptivity to previous results. The second problem will consider the vector nature of (1) by simulating the collision of two oppositely translating oblique fronts. The third case is a curved front, i.e., a propagating radial N-wave.

3.3.1. Stationary Burgers front

The stationary Burgers front is the classical solution to (1), exhibiting a straight front developing across the x^1-direction. We compare with analytic values the maximum derivative magnitude $|\partial_{x^1} u^1|_{\max}$ and the time $t_{\max}$ at which the maximum occurs. To compare with the literature [2], we set $\nu = 0.01/\pi$, $\hat{u}_2^\mu = 0$ and $\vec{\kappa}^\mu = \vec{e}^1 \delta^{\mu,1}$. 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 from (17) only at $t = t^0$, and integrate using a BDFM/Ext$M$ scheme to provide values at t^M ($M = 1,2$). A non-adaptive and an adaptive case with maximum refinement $\ell_{\max} = 3$ are considered. In the non-adaptive case, the element edges lie along $x^1 = 0, \pm 0.05, \pm 1$, whereas in the adaptive case, the elements are initially uniform. The second-derivative error criterion is used in this problem applied to $\vec{u}$, and the threshold and coarsening multiplier are $\varepsilon_t = 1$ and $\gamma_c = 0.5$, respectively.

Table 2a presents the non-adaptive results from GASpAR and from [27]. Besides the comparison in Tables 2a and b, we obtained analytic solutions using (16) combined with the 1D formula [37, (4.10)] computed using Gauss–Hermite quadrature, and verified $|\partial_{x^1} u^1|_{\max}$ to seven digits against the reported value [2]. Thus, we have also verified that the $\mathbb{L}_2$ accuracy of the solution is consistent with the derivative accuracy implied by Tables 2a and b. We note that the $p = 5$ case is comparatively poor (cf. [27]), possibly due to differences between the basis functions in the two methods [2], but our non-adaptive errors in $t_{\max}$ for our case are consistently better, while for $p > 5$ the $|\partial_{x^1} u^1|_{\max}$ errors are comparable (cf. [27]).

Table 2b shows the results from the adaptive case and the reference and control solutions, where *reference* refers to a solution on a non-adaptive grid with K fixed as at the adaptive solution at $t = t_{\max}$. Thus, it offers a solution computed with roughly as many d.o.f. as the adaptive solution, and hence requiring about the same computational effort, disregarding adaptivity overhead. Clearly, resolving the front is very challenging as evidenced by the reference solution for $p = 5$ actually diverging, and good solutions not being obtained until $p > 13$. The control solutions are all nearly identical to the adaptive ones, suggesting that our refinement criteria enable DARe to capture the formation of the front accurately, at a significantly reduced number of d.o.f. Indeed, on one processor, the computational times for the DARe cases are also reduced by a factor of about 7

Table 2
For the stationary Burgers front

p	Mavriplis [27]		GASpAR	
	$t_{\max}$	$\lvert\partial_x u\rvert_{\max}$	$t_{\max}$	$\lvert\partial_{x^1} u^1\rvert_{\max}$
(a) Non-adaptive results				
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
∞	0.51047	152.00516		

p	Adaptive		Reference		Control	
	$t_{\max}$	$\lvert\partial_{x^1} u^1\rvert_{\max}$	$t_{\max}$	$\lvert\partial_{x^1} u^1\rvert_{\max}$	$t_{\max}$	$\lvert\partial_{x^1} u^1\rvert_{\max}$
(b) Adaptive, reference, and control results						
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
∞	0.51047	152.00516				

compared with the control runs. Keeping in mind that on a single processor, no load balancing is required, we do not expect this level of efficiency for most turbulence problems. However, for the case where we are resolving largely isolated structures in an otherwise noisy background, we expect to see significant reductions in overall computational costs using DARe.

3.3.2. *N-wave problem*

The radial N-wave solution combines a d-dimensional Cole–Hopf transformation of (1) and a heat-equation solution (generalizing [37, (4.6) and (4.40)])

$$\vec{u} = -2\nu \vec{\nabla} \ln \chi \leftarrow \chi(\vec{x},t) = 1 + \frac{a}{t^{d/2}} \exp - \frac{(\vec{x}-\vec{x}^0)^2}{4\nu t}. \tag{18}$$

The N-wave emanates from $\vec{x}^0 = (\vec{e}^1 + \vec{e}^2)/2$. For this test, we initialize at $t^0 = 5 \times 10^{-2}$ and set $\nu = 5 \times 10^{-3}$ and $a = 10^4$. Dirichlet boundary conditions (2) on $\mathbb{D} = [0,1]^2$ are imposed at each time by evaluating (18) on $\partial\mathbb{D}$. The initial grid has $K = 4 \times 4$ elements, and we consider only the adaptive case with $\ell_{\max} = 4$. The refinement criteria are the same as in Section 3.2.

Fig. 5 presents six snapshots of the u^1 component of a typical N-wave system numerical solution, and illustrates the refinement patterns characteristic of all the runs. The solution has reflection symmetries, so for simplicity only one quadrant is shown. As the semicircular front propagates outward, the mesh refines to track it; while in the center the velocity components grow more planar, and the mesh coarsens. The front does not steepen in this problem, as it does in the planar front problem (Section 3.3.1); it simply decays as it propagates outward.

We set $p = 14$ and advance from $t = t^0$ to $t_f = 0.11$ for various constant Δt to produce the timestep error-convergence curve in Fig. 6a. This time interval was enough to provide a number of DARe events; nevertheless, the solution converges with Δt, at order (slope) 3.01.

To check spatial convergence, the solution is advanced from $t = t^0$ to $t_f = 0.11$ by using variable p and Δt (15) but fixed $Co = 0.15$. Fig. 6b shows the final $\mathbb{L}_2$ error vs. p. As with the linear advection case, the error behaves spectrally for a finite time integration.

3.3.3. *Colliding front problem*

Here we take (17) with $\hat{u}_2^\mu = (1/2)\delta^{\mu,1}$, an initial condition that develops two translating, colliding fronts, and use (16) to get a 2D bi-periodic vector solution to the system (1). We retain $\nu = 0.01/\pi$, and to set the fronts oblique to the axes put $\vec{\kappa}^\mu = (\vec{e}^1 + 2\vec{e}^2)\delta^{\mu,1}$. The mesh initially has $K = 4 \times 4$ elements of degree $p = 8$.

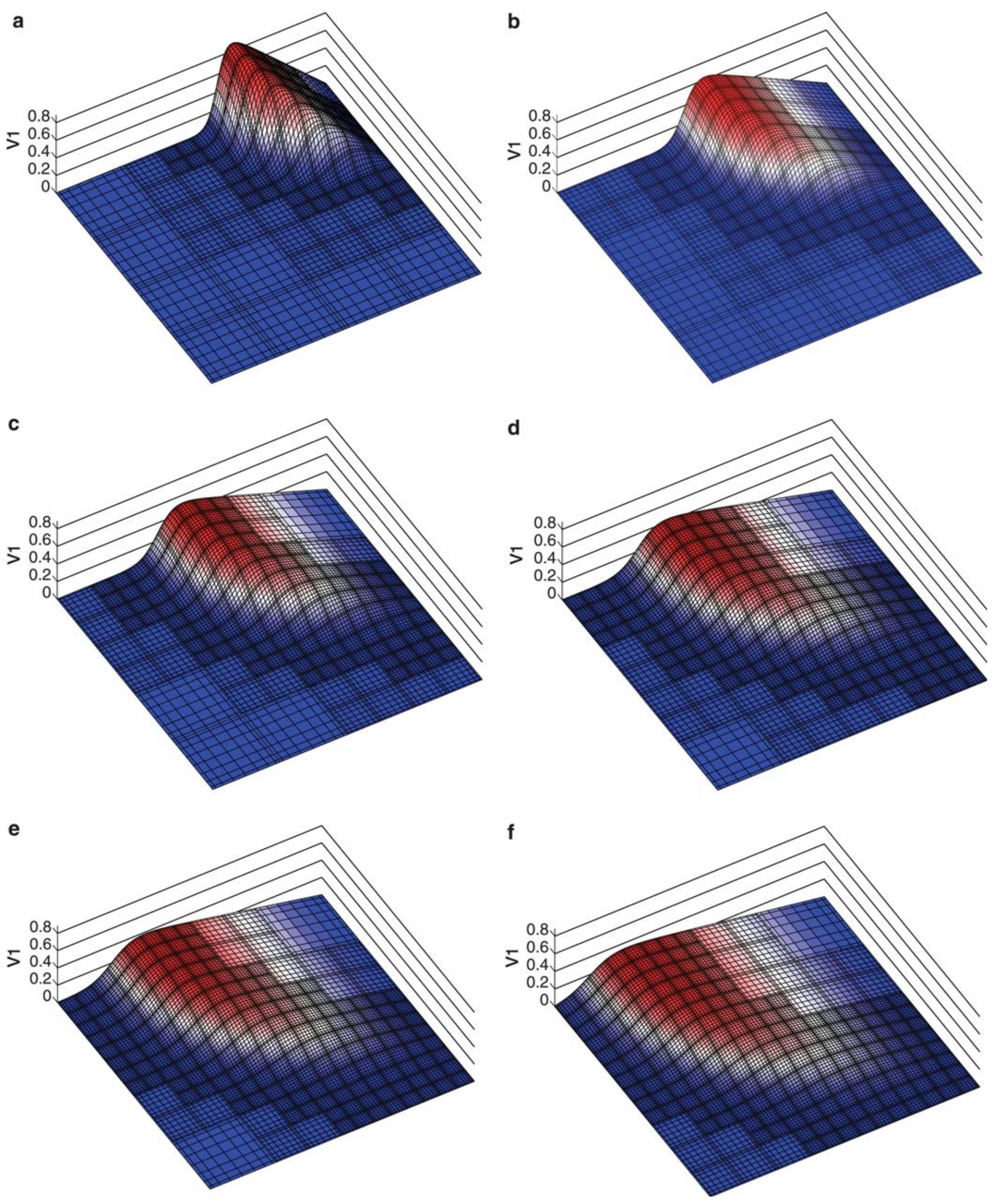

Fig. 5. For the $p = 8$ adaptive radial N-wave solution of (1) with $\vec{c} = \vec{u}$ and $\nu = 5 \times 10^{-3}$, initialized by (18), surface plots of $u^1(\vec{x}, t)$, showing $\vec{x} \in [\frac{1}{2}, 1]^2$ and $K/4 = 88$, 121, 139, 172, 181, 190 as $t = 0.18$, 0.33, 0.48, 0.65, 0.81, 1.00. Black and yellow curves show nodes and element edges, respectively.

Initialization is as in Section 3.3.1, except that we use a BDF2/Ext2 scheme for time integration. The second-derivative error criterion is used in this test, with threshold and coarsening multiplier $\varepsilon_t = 8$ and $\gamma_c = 0.2$, respectively. The maximum refinement is $\ell_{\max} = 5$. Because the mesh only has to resolve discrete fronts as they develop, translate, merge and decay there is clear potential for *computational* savings by using adaptivity: simply reducing the number of elements on which to compute. Here, we wish to illustrate this potential and to verify that the error in the solution is consistent with the results in Section 3.3.1. We do not consider a control run for this problem.

In Fig. 7 are presented six snapshots during the evolution of the u^1 component of the colliding-fronts system, zoomed to one quadrant of the domain. The mesh refines around each of the oppositely propagating

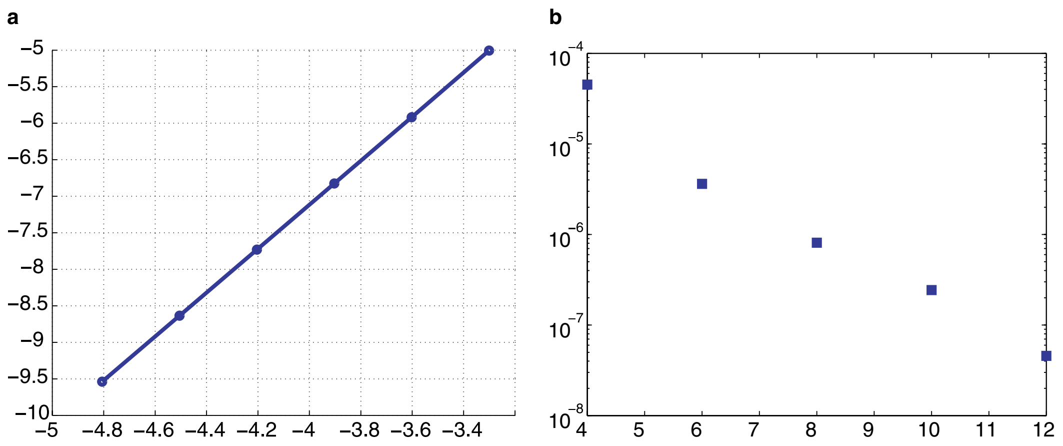

Fig. 6. For the adaptive radial N-wave solution of (1) with $\vec{c} = \vec{u}$ and $\nu = 5 \times 10^{-3}$, initialized by (18), plots of (a) $\log_{10}(\|\vec{u}_n - \vec{u}_a\|_2/\|\vec{u}_a\|_2)$ vs. $\log_{10}\Delta t$ for $p = 14$, with slope 3.01; and (b) $(\|\vec{u}_n - \vec{u}_a\|_2/\|\vec{u}_a\|_2)$ vs. p.

fronts as they steepen, merge and begin to decay. The dash-dotted curve of Fig. 8 shows the number K of elements increasing monotonically before and during the merger, and decreasing, as expected, after the merger is complete at about $t = 0.12$. Moreover, Fig. 7 shows that DARe occurs only in regions localized around the steepening or translating fronts. The maximum number of adaptive elements is $\max_t K = 3136$, while the control solution would require $K = 16{,}384$. This is a coverage fraction of about 19%, suggesting that adaptivity in this problem certainly offers a huge reduction in the required number of d.o.f.

Fig. 8 provides the time series for the maximum-magnitude and $\mathbb{L}_2$ solution errors (unnormalized) of u^1, as well as the relative error of $|\vec{\kappa}^1 \cdot \vec{\nabla} u^1|$. The solution errors are reasonably well behaved. As expected, there is much more variation of the derivative error. The analytic values for $|\partial_{x^1} u^1|_{\max}$ and the time, $t_{\max}$ at which this maximum occurs are 213 and 0.1280, respectively. From our results, we find that $|\partial_{x^1} u^1|_{\max} = 222$ and $t_{\max} = 0.1283$, which is entirely consistent with the stationary results presented in Table 2b.

Finally, Fig. 9 shows a snapshot solution and relative error field of an even more challenging problem, namely the same two colliding fronts orthogonally crossed by a stationary front. Also, to better exercise h-refinement, the degree was reduced to $p = 6$ from $p = 8$ in the previous test. The reduction in overall accuracy is consistent with the p-convergence results in Fig. 6b. The relative $\mathbb{L}_2$ error is $\|u^1_n - u^1_a\|_2/\|u^1_n\|_2 = 5.8 \times 10^{-3}$. The element distribution in Fig. 9b shows that the error estimation coincides well with the actual point-wise error field.

4. Discussion and conclusion

We have presented an overview of a geophysical and astrophysical spectral-element adaptive refinement (GASpAR) code, concentrating on the continuous Galerkin discretization of a vector (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 code constructs that establish connectivity and maintain continuity between conforming and non-conforming elements. From this basis, we have presented a new dynamic adaptive mesh refinement (DARe) algorithm for the spectral-element method, and in particular, described 2D refinement criteria and enumerated rules for self-consistent refinement and coarsening of a non-conforming element mesh. We propose several problems that have analytic time-dependent solutions, in order to test quantitatively the ability of the code to simulate accurately 2D phenomena arising as a result of linear and nonlinear advective and dissipative dynamics.

Using DARe, GASpAR can potentially generate a substantial savings both in the number of d.o.f. computed and in the computation time required to update them, despite the added overhead required by the adaptivity. These results suggest an obvious extension of this work, to apply DARe to broader classes of problems and systematically examine performance metrics and operational considerations with the goal of minimizing

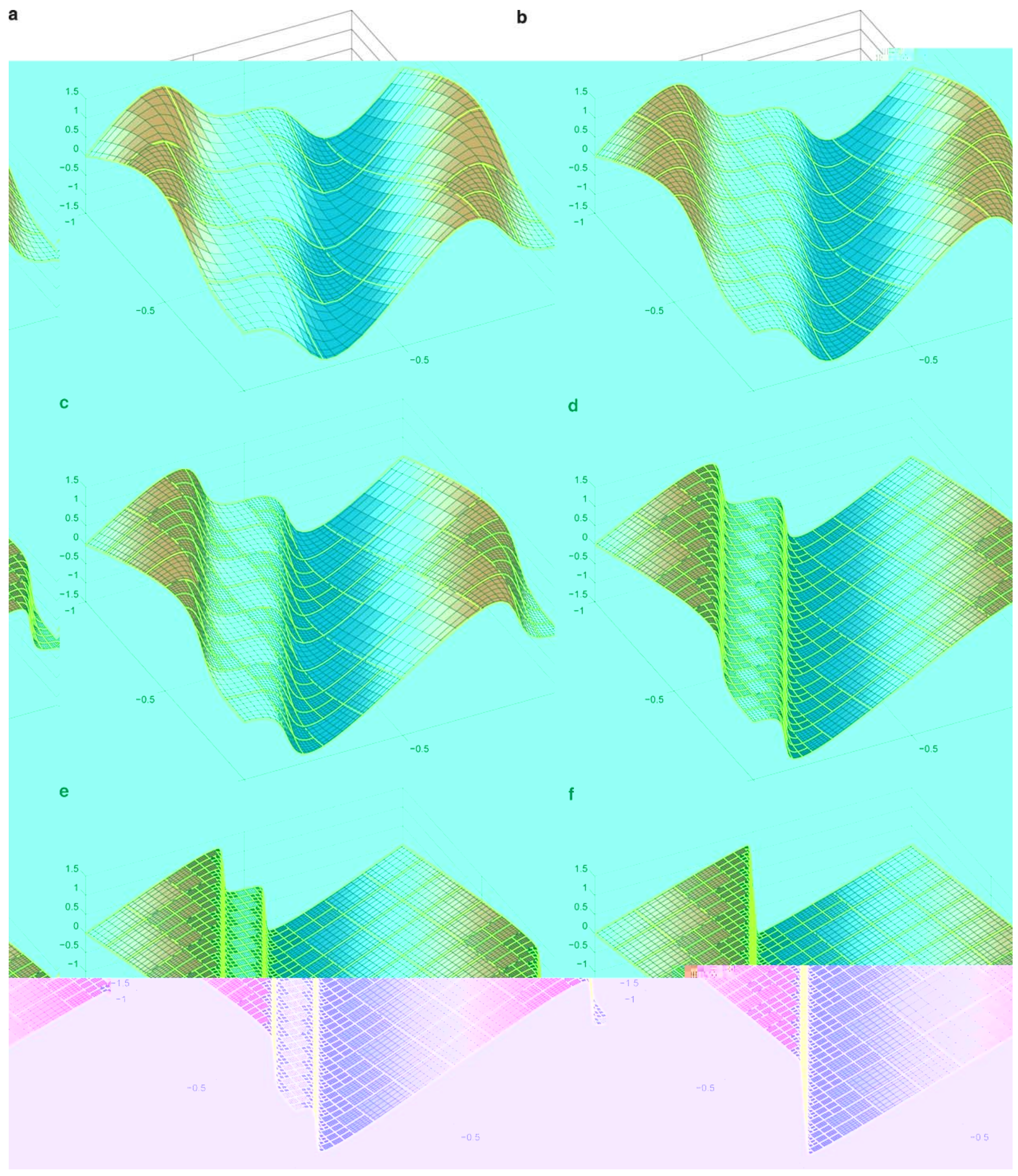

Fig. 7. For the $p = 8$ adaptive colliding front solution of (1) with $\vec{c} = \vec{u}$ and $\nu = 10^{-2}/\pi$, initialized by (16) and (17) with $\vec{\kappa}^{\mu} = (\vec{e}^{1} + 2\vec{e}^{2})\delta^{\mu,1}$ and $\hat{u}_{2}^{\mu} = \frac{1}{2}\delta^{\mu,1}$, surface plots of u^{1}, showing $\vec{x} \in [-1, 0]^{2}$ and $K/4 = 28, 52, 112, 352, 784, 481$ at the time abscissas noted in Fig. 8. Black and yellow curves show nodes and element edges, respectively.

adaptivity overhead, especially in a parallel environment. For example, for different turbulence problems, what is the optimal frequency (e.g., w.r.t. eddy turnover time) at which to perform adaptivity, in order to reduce overhead cost? This extension would also discuss load-balancing strategies that mitigate communication

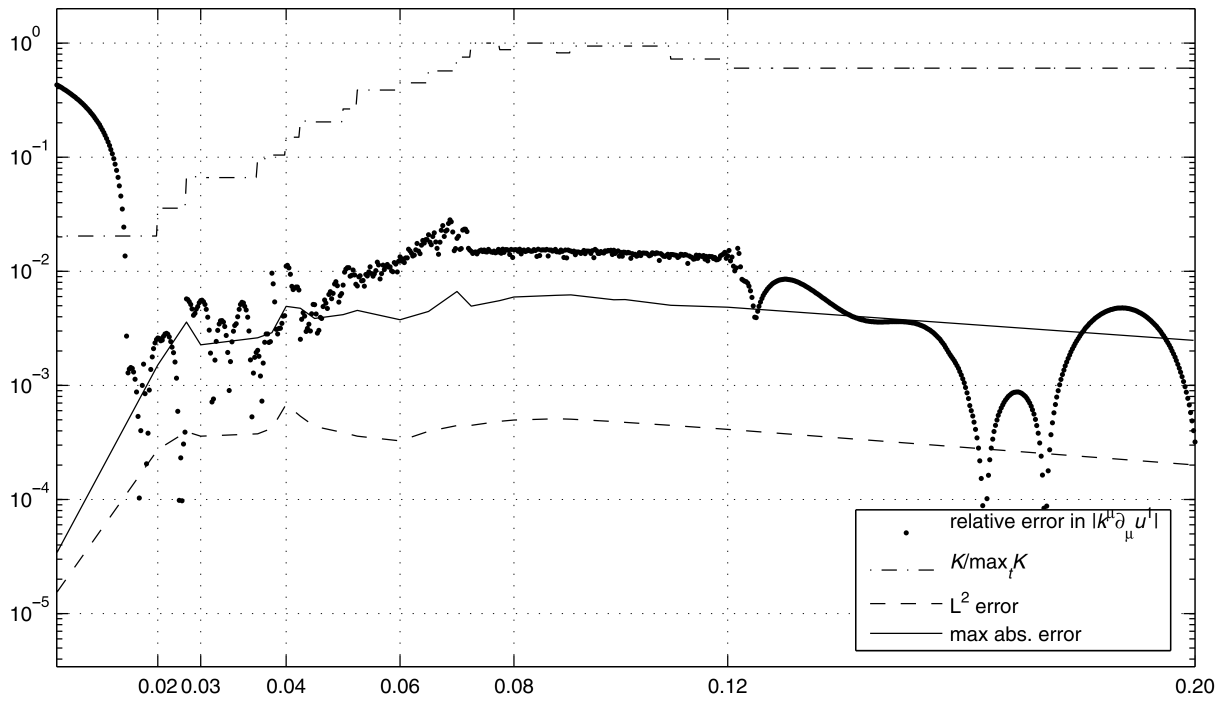

Fig. 8. For the $p = 8$ adaptive solution of (1) with $\vec{c} = \vec{u}$ and $\nu = 10^{-2}/\pi$, initialized by (16) and (17) with $\vec{\kappa}^{\mu} = (\vec{e}^1 + 2\vec{e}^2)\delta^{\mu,1}$ and $\hat{u}^{\mu}_2 = \frac{1}{2}\delta^{\mu,1}$, time series of fraction $K/\max_t K$ of elements (dash-dotted curve) and magnitude of relative maximum error in $|\vec{\kappa}^1 \cdot \vec{\nabla} u^1|$ (dot markers) vs. time. Also shown are the maximum-absolute (solid curve) and $\mathbb{L}_2$ (dashed curve) errors for u^1. The abscissa is marked at the six times of Fig. 7.

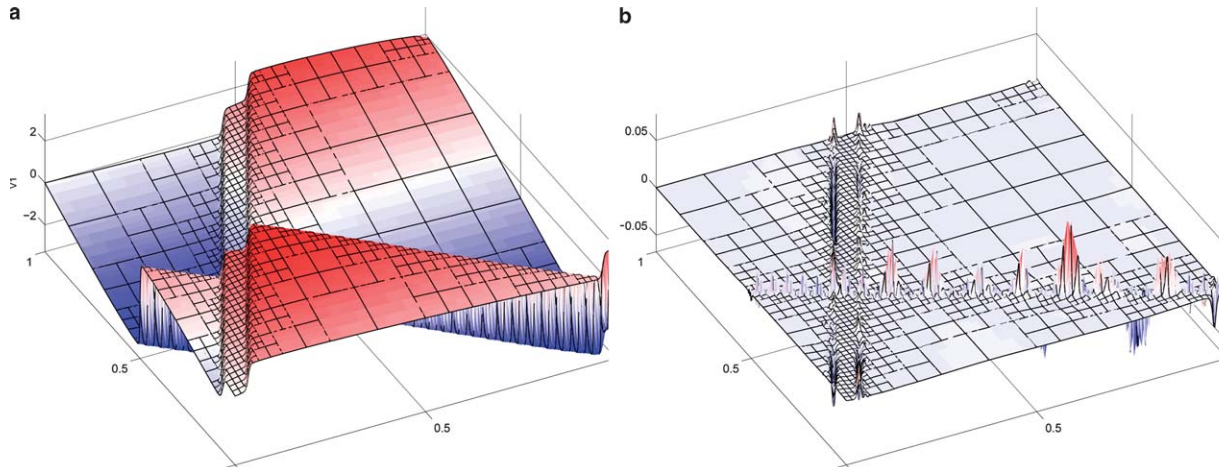

Fig. 9. For the $p = 6$ adaptive double colliding fronts solution of (1) with $\vec{c} = \vec{u}$ and $\nu = 10^{-2}/\pi$, initialized by (16) and (17) with $\vec{\kappa}^1 = \vec{e}^1 + 2\vec{e}^2$, $\vec{\kappa}^2 = \vec{e}^2 - 2\vec{e}^1$, $\hat{u}^1_2 = \frac{1}{2}$ and $\hat{u}^2_2 = 0$, surface plots of (a) $u^1_{\rm n}$ and (b) $(u^1_{\rm n} - u^1_{\rm a})/\|u^1_{\rm n}\|_{\infty}$, showing $\vec{x} \in [0, 1]^2$ and $K/4 = 1018$ at $t = 0.10$. For clarity, the node lines are not shown, but the element boundaries are now black.

bottlenecks, and optimize work-load distribution: when should repartitioning be done so as to minimize the cost of communicating to other processors?

The test problems show that DARe can be very beneficial for resolving isolated structures. But in practice, how likely is it that only a few isolated structures will exist? And how significant to the flow evolution are these structures, to the extent that their being resolved by DARe would preserve the overall flow statistics? These questions are the focus of current and future efforts and we will report on these investigations in regard to decaying turbulence in a subsequent paper. A useful approach to these questions provides that the fields solved for need not be those on which adaption criteria operate directly. 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. an appropriate functional, some relevant structures, when resolved, may allow for accurate simulation of the significant dynamics and statistics of the overall flow.

Acknowledgments

We thank Rich Loft, Peter Sullivan, Steve Thomas and Joe Tribbia for help at the outset of this work in using spectral element methods, and Huiyu Feng and Catherine Mavriplis for several useful discussions. Acknowledgment for computing time used in this research is made to NCAR, which is sponsored by the National Science Foundation and operated by the University Corporation for Atmospheric Research. The second author was also supported by the Office of Science (BER), U.S. Department of Energy, Grant No. DEFG0201ER63247 to the University of Maryland. The third author was supported by the U.S. Department of Energy under Contract W-31-109-ENG-38. Any opinions, findings and conclusions or recommendations are those of the authors and do not necessarily reflect the views of NSF or DoE.

Appendix A. Spectral-element formalism

In this appendix, we summarize results from the SEM literature, and our notation. Table A.1 shows the hierarchy of basic formulas progressing from one 1D element, through K^1 1D elements, to K d-dimensional elements. Any dependent variable $u = u(\xi)$ may be approximated by its projection $\mathscr{P}_p u$ on the space $\mathbb{V}_p$ of polynomials of degree p, using u-values on any $p+1$ distinct nodal points ξ_j:

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

where $\mathscr{E}_p u$ is the pointwise error and $\phi_j(\xi) := \prod_{j' \neq j}(\xi - \xi_{j'})/(\xi_j - \xi_{j'})$ denotes the Lagrange interpolating polynomials. Taking ξ_j and w_j from Table A.1 implies the quadrature

$$\langle u \rangle_1 := \int_{-1}^{1} u(\xi)\,\mathrm{d}\xi = \sum_{j=0}^{p} w_j u(\xi_j) + \mathscr{R}_p u(\xi'), \tag{A.2}$$

Table A.1
Hierarchy of spectral-element formulas, where L_j is the standard Legendre polynomial of degree j and norm $(j+\frac{1}{2})^{-1/2}, f \circ g(x) := f(g(x))$ and $1_{\mathbb{S}}(x) := \begin{cases} 1 & (x \in \mathbb{S}) \\ 0 & (\text{else}) \end{cases}$

Domain:	$\xi \in [-1,1]$;
	$x \in [-1,1] = \bigcup_{k=1}^{K^1} \bar{\mathbb{E}}_k^1$, where $]x_{k-1}, x_k[\equiv \mathbb{E}_k^1 := \vartheta_k(]-1,1[)$ has length $h_k^1 := x_k - x_{k-1} > 0 \Rightarrow \mathbb{E}_k^1 \bigcap \mathbb{E}_{k'}^1 = \emptyset$ if $k \neq k'$;
	$\vec{x} \in \bar{\mathbb{D}} = \bigcup_{k=1}^{K} \bar{\mathbb{E}}_k$, where $\mathbb{E}_k := \vec{\vartheta}_k(]-1,1[^d)$ has diameter $h_k := \max_\mu \max_{\vec{x},\vec{x}' \in \bar{\mathbb{E}}_k} \lvert x^\mu - x'^\mu \rvert$ and $\mathbb{E}_k \bigcap \mathbb{E}_{k'} = \emptyset$ if $k \neq k'$.
Nodes:	$\xi_j :=(j+1)$th least root of $(1-\xi^2)\frac{\mathrm{d}}{\mathrm{d}\xi}L_p$;
	$x_{j,k} :=\vartheta_k(\xi_j)$, where $\vartheta_k(\xi) := x_{k-1} + \frac{1}{2}h_k^1(1+\xi), k \in \{1,\ldots,K^1\}$;
	$\vec{x}_{\vec{j},k} := \vec{\vartheta}_k(\vec{\xi}_{\vec{j}})$, where $\xi_{\vec{j}}^\mu := \xi_{j^\mu}$ and $\vec{\vartheta}_k(\vec{\xi})$ is invertible but not necessarily linear.
Weights:	$w_j:=2/p(p+1)L_p(\xi_j)^2$;
	$w_{j,k} := \lvert \frac{\mathrm{d}}{\mathrm{d}\xi}\vartheta_k(\xi_j) \rvert w_j$;
	$w_{\vec{j},k} := \lvert \det \underset{\vec{\xi}}{\vec{\nabla}} \vec{\vartheta}_k(\vec{\xi}_{\vec{j}}) \rvert \prod_{\mu=1}^{d} w_{j^\mu}$.
Basis:	$\phi_{j'}(\xi) = w_{j'}\sum_{j=0}^{p} L_j(\xi_{j'})L_j(\xi)/\sum_{j''=0}^{p} w_{j''}L_j(\xi_{j''})^2 \underset{\xi \to \xi_j}{\longrightarrow} \delta_{j,j'}$;
	$\phi_{j,k}(x) := 1_{\mathbb{E}_k^1}(x)\phi_j \circ \vartheta_k^{-1}(x) \underset{x \to x_{j',k'}}{\longrightarrow} \begin{cases} 1, & x_{j,k} = x_{j',k'}, \\ 0, & \text{otherwise}; \end{cases}$.
	$\phi_{\vec{j},k}(\vec{x}) := 1_{\bar{\mathbb{E}}_k}(\vec{x})\phi_{\vec{j}} \circ \vec{\vartheta}_k^{-1}(\vec{x}) \underset{\vec{x} \to \vec{x}_{\vec{j}',k'}}{\longrightarrow} \begin{cases} 1, & \vec{x}_{\vec{j},k} = \vec{x}_{\vec{j}',k'}, \\ 0, & \text{otherwise}, \end{cases}$ where $\phi_{\vec{j}}(\vec{\xi}) := \prod_{\mu=1}^{d} \phi_{j^\mu}(\xi^\mu)$.

where $\mathscr{R}_p := -2^{2p+1}\frac{p^3(p+1)(p-1)!^4}{(2p+1)(2p)!^3}(\mathrm{d}/\mathrm{d}\xi)^{2p}$ is the residual operator [35] and $\xi' \in]-1,1[$. Then the mean-square error is bounded as

$$\langle(\mathscr{E}_p u)^2\rangle_1 \propto p^{1-2Q}\sum_{q=0}^{Q}\langle u^{(q)2}\rangle_1 \tag{A.3}$$

for any order Q of square-integrable derivative [9, (B.3.59)]. Thus if u is infinitely smooth then $\mathscr{P}_p u$ converges to u *spectrally*.

Now let $[-1,1]$ be covered by K^1 disjoint 1D elements $\mathbb{E}^1_k$ as in Table A.1 (noting that nonlinear invertible ϑ_k may sometimes be preferable). Then u may be approximated by its projections $\mathscr{P}_{k,p}u$ on the space $\mathbb{V}_{\boldsymbol{h}^1,p}$ of piecewise polynomials of degree p on the $\mathbb{E}^1_k$. That is, (A.1) generalizes to

$$u = \sum_{k=1}^{K^1}(\mathscr{P}_{k,p}u + \mathscr{E}_{k,p}u), \quad \mathscr{P}_{k,p}u := \sum_{j=0}^{p}u(x_{j,k})\phi_{j,k}, \tag{A.4}$$

where $\mathscr{E}_{k,p}u := \mathscr{E}_p(u \circ \vartheta_k) \circ \vartheta_k^{-1}$. Then (A.2) generalizes to

$$\langle u\rangle_1 = \sum_{k=1}^{K^1}\int_{x_{k-1}}^{x_k}u(x)\,\mathrm{d}x, \quad \int_{x_{k-1}}^{x_k}u(x)\,\mathrm{d}x = \sum_{j=0}^{p}w_{j,k}u(x_{j,k}) + \mathscr{R}_{k,p}u(x'_k), \tag{A.5}$$

where $\mathscr{R}_{k,p}u := (h^1_k/2)^{2p+1}\mathscr{R}_p(u \circ \vartheta_k) \circ \vartheta_k^{-1}$ and $x'_k \in \mathbb{E}^1_k$.

Generalizing further, assume a d-dimensional problem domain $\mathbb{D}$ can be partitioned as in Table A.1. Now generalizing (A.4), one may approximate a field $u(\vec{x})$ by its projections $\mathscr{P}_{k,\vec{p}}\,u$ on the space $\mathbb{V}_{\boldsymbol{h},\vec{p}}$ of piecewise polynomials of degree p^μ in coordinate x^μ on the $\mathbb{E}_k$. That is, (A.4) generalizes to

$$u \approx \mathscr{P}_{\boldsymbol{h},\vec{p}}u := \sum_{k=1}^{K}\mathscr{P}_{k,\vec{p}}u, \qquad \mathscr{P}_{k,\vec{p}}\,u := \sum_{\vec{\jmath}\in\mathbb{J}}u(\vec{x}_{\vec{\jmath},k})\phi_{\vec{\jmath},k}, \tag{A.6}$$

where $\mathbb{J} := \{\vec{\jmath}\,|\,\jmath^\mu \in \{0,\ldots,p^\mu\}\}$. The appropriate approximation of a vector

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

uses $\vec{\boldsymbol{\phi}}$ with entries $\vec{\phi}^{\mu}_{\vec{\jmath},k} := \phi_{\vec{\jmath},k}\vec{e}^\mu$ and $\boldsymbol{u}$ with entries $u^\mu_{\vec{\jmath},k} := u^\mu(\vec{x}_{\vec{\jmath},k})$, where $\vec{e}^\mu$ denotes the Cartesian unit vectors. For scalars u, (A.5) generalizes to

$$\langle u\rangle := \int\underset{\mathbb{D}}{\cdots}\int u(\vec{x})\,\mathrm{d}^d\vec{x} = \sum_{k=1}^{K}\int\underset{\mathbb{E}_k}{\cdots}\int u(\vec{x})\,\mathrm{d}^d\vec{x} \approx \sum_{k=1}^{K}\sum_{\vec{\jmath}\in\mathbb{J}}w_{\vec{\jmath},k}u(\vec{x}_{\vec{\jmath},k}) =: \langle u\rangle_{\mathrm{GL}}. \tag{A.7}$$

Finally, variational formulation depends on the inner product from (A.7):

$$\langle u,v\rangle := \langle uv\rangle \approx \sum_{k=1}^{K}\sum_{\vec{\jmath}\in\mathbb{J}}w_{\vec{\jmath},k}u(\vec{x}_{\vec{\jmath},k})v(\vec{x}_{\vec{\jmath},k}) =: \langle u,v\rangle_{\mathrm{GL}} \tag{A.8}$$

for scalars, $\langle\vec{u},\vec{v}\rangle := \sum_{\mu=1}^{d}\langle u^\mu,v^\mu\rangle$ for vectors, $\langle\vec{\vec{u}},\vec{\vec{v}}\rangle := \sum_{\mu,\mu'=1}^{d}\langle u^{\mu,\mu'},v^{\mu,\mu'}\rangle$ for tensors, and so forth. This implies a norm $\|u\|_2 := \langle u,u\rangle^{1/2}_{\mathrm{GL}}$. The norm $\|u\|_\infty := \max_{k=1}^{K}\max_{\vec{\jmath}\in\mathbb{J}}|u(\vec{x}_{\vec{\jmath},k})|$ is also used.

Now define the function spaces

$$\mathbb{U}_{\vec{b}} := \left\{\vec{u} = \sum_{\mu=1}^{d}u^\mu\vec{e}^\mu\,|\,u^\mu \in \mathbb{H}^1(\mathbb{D})\forall\mu \text{ and } \vec{u} = \vec{b} \text{ on } \partial\mathbb{D}\right\}$$

$$\mathbb{H}^1(\mathbb{D}) := \{f|f \in \mathbb{L}_2(\mathbb{D}) \text{ and } \partial_{x^\mu}f \in \mathbb{L}_2(\mathbb{D})\forall\mu\}.$$

Searching the piecewise polynomial subspace $\mathscr{P}_{\boldsymbol{h},\vec{p}}\mathbb{U}_{\vec{b}}\subsetneq\mathbb{U}_{\vec{b}}$ for a solution to (4) leads to (5), where

$$M^{\mu,\mu'}_{\vec{\jmath},\vec{\jmath}';k} := \langle\vec{\phi}^{\mu}_{\vec{\jmath},k},\vec{\phi}^{\mu'}_{\vec{\jmath}',k}\rangle_{\mathrm{GL}} = \delta_{\vec{\jmath}\vec{\jmath}'}\delta^{\mu,\mu'}w_{\vec{\jmath},k}, \tag{A.9}$$

$$C^{\mu,\mu'}_{\vec{\jmath},\vec{\jmath}';k} := \langle\vec{\phi}^{\mu}_{\vec{\jmath},k},\mathscr{C}\vec{\phi}^{\mu'}_{\vec{\jmath}',k}\rangle_{\mathrm{GL}} = \delta^{\mu,\mu'}w_{\vec{\jmath},k}\vec{c}_{\vec{\jmath},k}\cdot\vec{\nabla}\,\phi_{\vec{\jmath}',k}(\vec{x}_{\vec{\jmath},k}), \tag{A.10}$$

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

and $\vec{c}_{\vec{\jmath},k}(t) := \vec{c}(\vec{x}_{\vec{\jmath},k},t)$. The matrix $\mathbf{L}$ for deformed $\mathbb{E}_k$ (nonlinear $\vec{\vartheta}_k$) can also be constructed (e.g. [9]), and is supported in GASpAR.

As an example of global assembly, for the mesh partition in Fig. 1a, (9) takes the following explicit form (suppressing zero-valued and $\mu > 1$ blocks):

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

For the mesh in Fig. 1b, the explicit form of (9) for the non-conforming assembly matrix $\mathbf{A} = \mathbf{\Phi}\mathbf{A}_{\mathrm{c}}$ is (suppressing zero-valued and $\mu > 1$ blocks)

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

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

Note that the $\mathbf{A}$ entries corresponding to the child-node rows (see Fig. 1b) 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 non-conforming interface.

References

[1] G. Anagnostou, Y. Maday, C. Mavriplis, A.T. Patera, On the mortar element method: generalizations and implementation, in: Third International Symposium on Domain Decomposition Methods, SIAM, Philadelphia, PA, 1989, pp. 157–173.
[2] C. Basdevant, M. Deville, P. Haldenwang, J.M. Lacroix, J. Ouazzani, R. Peyret, P. Orlandi, A. Patera, Spectral and finite difference solutions of the Burgers equation, Comput. Fluids 14 (1986) 23–41.

[3] F.B. Belgacem, The mixed mortar finite element method for the incompressible Stokes problem: convergence analysis, SIAM J. Numer. Anal. 37 (4) (2000) 1085–1100.
[4] C. Bernardi, Y. Maday, C. Mavriplis, A.T. Patera, The mortar element method applied to spectral discretizations, in: T.J. Chung, G.R. Karr (Eds.), Proceedings of the Seventh International Conference on Finite Element Methods in Flow Problems, University of Alabama, Huntsville, 1989.
[5] Rong-Yeu Chang, Chia-Hsiang Hsu, A variable-order spectral element method for incompressible viscous flow simulation, Int. J. Numer. Meth. Eng. 39 (1996) 2865–2887.
[6] F. Casadei, E. Gabellini, G. Fotia, F. Maggio, A. Quarteroni, A mortar spectral/finite element method for complex 2D and 3D elastodynamic problems, Comput. Meth. Appl. Mech. Eng. 191 (2002) 5119–5148.
[7] E. Chaljub, Y. Capdeville, J.P. Vilotte, Solving elastodynamics in a fluid-solid heterogeneous sphere: a parallel spectral element approximation on non-conforming grids, J. Comput. Phys. 187 (2003) 457–491.
[8] J. Dennis, A. Fournier, W. Spotz, A. St.-Cyr, M. Taylor, S. Thomas, H. Tufo, High resolution mesh convergence properties and parallel efficiency of a spectral element atmospheric dynamical core, Int. J. High Perf. Comput. Appl. 19 (2005) 225–235.
[9] M.O. Deville, P.F. Fischer, E.H. Mund, High-Order Methods for Incompressible Fluid Flow, Cambridge University Press, Cambridge, 2002.
[10] Y. Dubois-Pelerin, V. Van Kemenate, M.O. Deville, An object-oriented toolbox for spectral element analysis, J. Sci. Comput. 14 (1999) 1–29.
[11] B.G. Elmegreen, J. Scalo, Interstellar turbulence, I: observations and processes, Ann. Rev. Astron. Astrophys. 42 (2004) 211–273.
[12] H. Feng, C. Mavriplis, Adaptive spectral element simulations of thin flame sheet deformations, J. Sci. Comput. 17 (2002) 1–3.
[13] P.F. Fischer, G.W. Kruse, F. Loth, Spectral element methods for transitional flows in complex geometries, J. Sci. Comput. 17 (1) (2002) 81–98.
[14] A. Fournier, G. Beylkin, V. Cheruvu, Multiresolution adaptive space refinement in geophysical fluid dynamics simulation, Lecture Notes Comput. Sci. Eng. 41 (2005) 161–170.
[15] A. Fournier, M.A. Taylor, J.J. Tribbia, The spectral element atmosphere model (SEAM): high-resolution parallel computation and localized resolution of regional dynamics, Mon. Wea. Rev. 132 (2004) 726–748.
[16] Uriel Frisch, Turbulence: The Legacy of A.N. Kolmogorov, Cambridge University Press, Cambridge, 1995.
[17] R.D. Henderson, Unstructured spectral element methods for simulation of turbulent flows, J. Comput. Phys. 122 (1995) 191–217.
[18] R.D. Henderson, Dynamic refinement algorithms for spectral element methods, Comput. Meth. Appl. Mech. Eng. 175 (1999) 395–411.
[19] T. Isihara, Y. Kaneda, M. Yokokawa, K. Itakura, A. Uno, Spectra of energy dissipation, enstrophy and pressure by high-resolution direct numerical simulations of turbulence in a periodic box, J. Phys. Soc. Jpn. 72 (2003) 983–986.
[20] M. Iskandarani, D.B. Haidvogel, J.C. Levin, A three-dimensional spectral element model for the solution of the hydrostatic primitive equations, J. Comput. Phys. 186 (2003) 397–426.
[21] G.E. Karniadakis, S.J. Sherwin, Spectral/hp Element Methods for CFD, Oxford University Press, New York, 1999.
[22] G.E. Karniadakis, M. Israeli, S.A. Orszag, High-order splitting methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 97 (1991) 414–443.
[23] D.A. Kopriva, S.L. Woodruff, M.Y. Hussaini, Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method, Int. J. Numer. Meth. Eng. 53 (2002) 105–122.
[24] G.W. Kruse, Parallel nonconforming spectral element solution of the incompressible Navier–Stokes equations in three dimensions, Ph.D. Dissertation, Division of Applied Mathematics, Brown University, 1997.
[25] J.G. Levin, M. Iskandarani, D.B. Haidvogel, A nonconforming spectral element ocean model, Intl. J. Numer. Meth. Fluids 34 (2000) 495–525.
[26] Y. Maday, C. Mavriplis, A.T. Patera, Nonconforming mortar element methods: Application to spectral discretizations, in: Domain Decomposition Methods, SIAM, Philadelphia, PA, 1989, pp. 392–418. Also ICASE Report 88-59.
[27] C. Mavriplis, Adaptive mesh strategies for the spectral element method, Comput. Meth. Appl. Mech. Eng. 116 (1994) 77–86.
[28] C. Meneveau, J. Katz, Scale-invariance and turbulence models for large-eddy simulation, Annu. Rev. Fluid Mech. 32 (2000) 1–32.
[29] A. Patera, A spectral element method for fluid dynamics: laminar flow in a channel expansion, J. Comput. Phys. 54 (1984) 468–488.
[30] E. Rønquist, Convection treatment using spectral elements of different order, Int. J. Numer. Meth. Fluids 22 (1996) 241–264.
[31] C. Sert, A. Beskok, Spectral element formulation on non-conforming grids: A comparative study of pointwise matching and integral projection methods, J. Comput. Phys. 211 (2006) 300–325.
[32] R.J. Shewchuck, An introduction to the conjugate gradient method without the agonizing pain Available from: http://www-2.cs.cmu.edu/jrs/jrspapers.html 1994.
[33] I. Sytine, D. Porter, P. Woodward, S. Hodson, K.-H. Winkler, Convergence tests for the piecewise parabolic method and Navier–Stokes solutions for homogeneous compressible turbulence, J. Comput. Phys. 158 (2000) 225–238.
[34] H.M. Tufo, P.F. Fischer, Terascale spectral element algorithms and implementations", in: Proceedings of the ACM/IEEE SC99