INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2009; **60**:473–514
Published online 23 October 2008 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1893

An adaptive cut-cell method for environmental fluid mechanics

Michael F. Barad[1,*,†], Phillip Colella[2] and S. Geoffrey Schladow[1]

[1]*Department of Civil and Environmental Engineering, University of California, Davis, CA 95616, U.S.A.*
[2]*Lawrence Berkeley National Laboratory, Berkeley, CA 94720, U.S.A.*

SUMMARY

In this work we present a numerical method for solving the incompressible Navier–Stokes equations in an environmental fluid mechanics context. The method is designed for the study of environmental flows that are multiscale, incompressible, variable-density, and within arbitrarily complex and possibly anisotropic domains. The method is new because in this context we couple the embedded-boundary (or cut-cell) method for complex geometry with block-structured adaptive mesh refinement (AMR) while maintaining conservation and second-order accuracy. The accurate simulation of variable-density fluids necessitates special care in formulating projection methods. This variable-density formulation is well known for incompressible flows in unit-aspect ratio domains, without AMR, and without complex geometry, but here we carefully present a new method that addresses the intersection of these issues. The methodology is based on a second-order-accurate projection method with high-order-accurate Godunov finite-differencing, including slope limiting and a stable differencing of the nonlinear convection terms. The finite-volume AMR discretizations are based on two-way flux matching at refinement boundaries to obtain a conservative method that is second-order accurate in solution error. The control volumes are formed by the intersection of the irregular embedded boundary with Cartesian grid cells. Unlike typical discretization methods, these control volumes naturally fit within parallelizable, disjoint-block data structures, and permit dynamic AMR coarsening and refinement as the simulation progresses. We present two- and three-dimensional numerical examples to illustrate the accuracy of the method. Copyright © 2008 John Wiley & Sons, Ltd.

Received 25 July 2007; Revised 25 June 2008; Accepted 13 July 2008

KEY WORDS: embedded boundary; adaptive mesh refinement; variable-density incompressible flows; projection method; environmental fluid mechanics; finite-volume method

*Correspondence to: Dr Michael F. Barad, Environmental Fluid Mechanics Laboratory, Stanford University, Stanford, CA 94305, U.S.A.
†E-mail: barad@stanford.edu

Contract/grant sponsor: U.S. Department of Energy CSGF program
Contract/grant sponsor: U.S. National Science Foundation MSPRF program
Contract/grant sponsor: U.S. Department of Energy Office of Science; contract/grant number: DE-AC02-05CH11231
Contract/grant sponsor: U.S. Environmental Protection Agency; contract/grant number: R826940-01-0

1. INTRODUCTION

We have developed a new method for the study of highly nonlinear, multiscale, incompressible environmental fluid mechanics. With this method we solve the variable-density incompressible Navier–Stokes equations. Flows in oceans, lakes, and rivers are well represented by these equations. Examples of such flows are internal gravity waves, coastal plumes, density currents in lakes, buoyancy-driven exchange flows, flows in branched estuarine slough networks, and flows past highly complex topography. Issues involved include variable-density, anisotropic domains, complex and often sparse geometries, large ranges in spatial and temporal scales, moving fronts and highly complex mixing zones. We hope to provide a predictive tool for both engineering and science with an enhanced ability to interpret and extend the results of field and laboratory studies.

In this section the key features of our method for simulating environmental fluid mechanics are outlined. Specifically, the governing equations, accuracy of the method, spatial discretization, and implementation are discussed.

1.1. Background on projection methods

Historically, computational environmental fluid mechanics methods have focused on simplifying either the governing equations or the dimensionality of the problem (and often both). The equations have been reduced to as simple as the one-dimensional (1D) Bernoulli equations, yet most numerical methods for environmental fluid mechanics solve either shallow-water, hydrostatic, and/or Boussinesq approximations in 1D, 2D, or 3D. For a review of existing environmental fluid mechanics models, see [1, 2].

Our approach solves the incompressible Navier–Stokes equations in either 2D or 3D, with a variable-density forcing following [3, 4], and exhibits second-order-accurate convergence in time and space. Solving the non-hydrostatic equations is essential for accurately simulating flows with pressure distributions that significantly deviate from hydrostatic (i.e. $p \neq \rho_o g z + p_o$). Avoiding the (inviscid) Boussinesq approximation permits the simulation of flows where density deviations are significant enough to alter the motions through more than just the buoyancy term.

There is a range of approaches for solving the incompressible Navier–Stokes equations. Methods include artificial compressibility methods [5], Lagrangian vortex methods [6], and different flavors of projection methods, among others. Most numerical methods for environmental fluid mechanics assume either homogeneous or Boussinesq fluids. This work is relatively new in that the projection formulation eliminates Boussinesq errors in all but the viscous term. Viscous-term Boussinesq errors (which are very small when density variations are small and even smaller when the Reynolds number is large) can be eliminated with a minor extension following [3].

Projection methods are a powerful solution technique for the incompressible Navier–Stokes equations pioneered in a series of papers by Chorin [7–9]. In Chorin's work, he developed a numerical projection method using a discrete form of the Helmholtz–Hodge decomposition. In projection methods the solution is advanced by predicting a velocity field that does not satisfy a discrete divergence constraint and then by correcting the predicted velocity so that it is divergence-free. In the predictor step, it is necessary to explicitly approximate the advective terms and (for viscous problems) to subsequently solve a parabolic equation implicitly for the predictor velocity. The correction is achieved by solving an elliptic equation implicitly via a Helmholtz–Hodge decomposition (see [10]). For the parabolic solves in the predictor step, researchers have used

DOI: 10.1002/fld

first-order-accurate backward-Euler methods, second-order-accurate Crank–Nicolson [11], and implicit Runge–Kutta methods [12–14].

Chorin's original work was based on first-order-accurate discretizations. The projection method was then extended to second-order accuracy by Bell *et al.* [11]. The method was further extended to variable-density flows in [4], and subsequently to adaptive mesh refinement (AMR) in [3, 15, 16]. A recent extension of the methodology for the study of homogeneous incompressible fluids in arbitrarily complex geometry is presented in [17]. Various other researchers have extended the projection method, including [18–24] among others. See [25] for a more detailed history of projection methods. In this paper we extend the approach of [11, 17] to environmental fluid mechanics problems, where all of the following factors are important: variable-density, anisotropic mesh spacing, AMR, and complex geometry.

The algorithms in this research yield second-order-accurate (in time and space) solution errors for the governing equations. For second-order methods, when the mesh spacing is refined by a factor of two, the error is reduced by a factor of four [11]. Higher-order methods can resolve flow features that lower-order methods can only resolve with significantly finer (and therefore less efficient) grids. In this work, and in the work of [26], we systematically show second-order-accurate results for both the decoupled operators and for a range of test problems.

1.2. Background on embedded-boundary (EB) spatial discretizations

To study fluid systems, researchers typically use one of two spatial reference frames: Eulerian or Lagrangian. The Lagrangian reference frame follows individual fluid parcels as they move through space and time, while the Eulerian reference frame focuses on fixed regions in space where fluid may pass. For this work we use the Eulerian reference frame. A discussion of Lagrangian methods is beyond the scope of this work; the interested reader will find [27–30] informative. With an Eulerian method, researchers typically discretize space using one of three methods: finite element, finite difference, or finite volume.

Typical finite-element methods have the benefit of irregular elements. These elements have the attractive characteristic that, with refinement, they can accurately fit arbitrarily complex geometries, yet they necessitate unstructured data structures for the entire flow field (unstructured data structures are discussed below), and sometimes lead to non-conservative discretizations. Galerkin finite-element methods have an established community and have been successfully used by many researchers [31–37].

The most basic finite-difference methods utilize 'stair-step' (i.e. in or out rectangular cells) approximations to irregular geometry and suffer from a catastrophic lack of accuracy for non-orthogonal geometries. See [26] for an analysis of geometric error due to stair-stepping. Owing to their implementation simplicity, immersed-boundary and ghost-fluid methods [38–40] are attractive finite-difference-based methods but suffer due to conservation issues stemming from a non-flux- based approach. In immersed-boundary methods the boundary is treated as a body force, which is quite different from finite-volume-based methods where fluxes are imposed at boundaries.

The finite-volume method is an extremely attractive spatial discretization method due to its conservation properties. In finite-volume methods the flow domain is decomposed into individual control volumes where discretizations are based on fluxes across faces. A flavor of finite-volume methods is the mapped-grid method, in which rectangular control volumes are mapped onto a curvilinear coordinate system through a mathematical transformation or mapping function. Mapped

DOI: 10.1002/fld

grids are a very promising method, yet they are not able to handle arbitrarily complex geometries (for lack of a mapping function).

To approximate complex geometry we use the EB (or cut-cell) method [12, 13, 17, 41, 42], an elegant finite-volume technique where control volumes are formed by the intersection of the domain with rectangular grid cells. The intersections can be computed to arbitrary accuracy and are generated with an $O(N^{D-1/D})$ algorithm from distance-type functions (where $N \equiv$ total number of control volumes, including regular, and where D is the dimensionality of the problem).

Grid generation for EB methods is robust, 'water-tight' and accurate, and naturally fits with geometric coarsening/refinement used with AMR and multigrid. A major advantage of EB finite-volume methods is that the stability and robustness properties are not dependent on geometry. Other methods often require a high degree of care in grid generation to ensure stable and robust simulations.

With this efficient and accurate EB spatial description, the geometry can have non-unit-aspect ratios and be arbitrarily complex, yielding tractable, efficient and accurate conservative finite-volume discretizations. This geometric capability permits the accurate study of a variety of complex fluid flows, such as those that occur on coastal shelves, in branched estuarine slough networks, in complex closed conduits, and even past the detailed structures of benthic invertebrates. The current generation of numerical methods for numerically simulating environmental fluid mechanics is severely limited in its ability to model such complex geometries.

1.3. Background on block-structured AMR

If control volumes are computationally represented on a rectangular array, or on a union of rectangular arrays, the mesh is termed structured; otherwise the mesh is termed unstructured. Meshes that can change as a simulation progresses are referred to as temporally adaptive. Meshes that are not uniform throughout the spatial domain are referred to as spatially adaptive.

Models that use unstructured meshes have cumbersome data structures that are computationally less efficient to access and require more complex mathematical techniques to maintain accuracy. While unstructured mesh models have the benefit of spatial adaptivity, the use of refinement in time on unstructured grids is largely unexplored. Non-adaptive, structured mesh codes utilize highly efficient data structures but require global refinement that is prohibitively costly for multiscale flows. The most successful structured, spatially and temporally adaptive codes use blocks to partition the domain into regions of equal refinement. Block-structured adaptive codes benefit from highly efficient data structures, the ability to locally refine/coarsen as a simulation progresses, and well-understood conservative finite-volume discretization methods. Very few computational fluid mechanics models use meshes that are adaptive in time and space.

Adaptive methods for the numerical solution of partial differential equations concentrate computational effort when and where it is most needed. For over two decades, block-structured AMR has proven useful for overcoming limitations in computational resources and accuracy in such fluid fields as astrophysics [43, 44], combustion [45, 46], atmospheric science [47], applied mathematics [3, 48–53], aerospace engineering [54], and ink-jet printer design [23]. In environmental fluid mechanics, most prior mesh-adapting work has focused on static, one-way nested refinement strategies, where finer grids are nested within coarser grids and these grids do not adapt in time. In one-way nesting, there is no feedback from the fine grid to the coarse grid [55], while in two-way nesting, the solution on the fine grid is coupled to the solution on the coarse grid [3, 16, 56, 57].

DOI: 10.1002/fld

Conservative flux-based two-way AMR coupling has desirable well-posedness properties; e.g. divergence operators have telescoping sums [48].

The physical processes that environmental fluid mechanics researchers study often contain spatial and temporal scales that span several orders of magnitude. Second-order-accurate AMR with conservative two-way nesting provides the capability to simultaneously capture scales ranging many orders of magnitude (e.g. from hundreds of kilometers to meters). For example, to simulate oceanic internal gravity waves, coarse grids are placed over the entire flow domain, and recursively finer, nested grids adaptively track wave generation, propagation, interaction and dissipation, expending computational resources only where and when they are needed. With AMR, one can 'zoom in' on moving regions and accurately capture the important flow physics at multiple scales. Accurately capturing a range of spatial and temporal scales is critical to accurately simulating complex environmental fluid mechanics. The finite-volume EB approach naturally fits within parallelizable, disjoint-block data structures, and permits dynamic AMR coarsening and refinement as the simulation progresses (see Figure 4 for an example of an EB AMR grid). The application of AMR to the study of environmental fluid mechanics in complex geometry is a new and powerful technique. In this research we use fully adaptive, block-structured, two-way nested meshes coupled with the EB finite-volume method.

1.4. Implementation

This work was implemented within, and extended, the Chombo [58, 59] AMR framework. Chombo provides the necessary data structures to implement the highly complex, adaptive algorithms required for solving the incompressible Navier–Stokes equations in irregular domains. Chombo builds and executes on a range of computational platforms, from laptops to parallel supercomputers. Chombo is implemented primarily in the C++ programming language. For performance reasons, Chombo also provides an interface to FORTRAN for fast, regular array operations. Chombo also provides an elegant dimension-independent programming paradigm that accelerates the development cycle due both to the ease of debugging in reduced dimensions and to a single code-base for multiple dimensions.

2. GOVERNING EQUATIONS

The governing equations that we solve with this method are the variable-density incompressible Navier–Stokes equations. These well-known partial differential equations are given in the following sub-section; this is followed by a sub-section on boundary conditions, and finally we have a sub-section that reviews the mathematical background for our variable-density projection.

2.1. Incompressible Navier–Stokes equations

The governing equations are composed of:

Momentum balance:

$$\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{\nabla p}{\rho} + \mathbf{g} + \nu\Delta\mathbf{u} \tag{1}$$

DOI: 10.1002/fld

Divergence-free constraint:

$$\nabla \cdot \mathbf{u} = 0 \tag{2}$$

Density conservation:

$$\rho_t + \mathbf{u} \cdot \nabla \rho = 0 \tag{3}$$

Passive scalar transport:

$$s_t + \mathbf{u} \cdot \nabla s = k_s \Delta s + H_s \tag{4}$$

We have intentionally omitted the Coriolis term in the momentum balance. For problems where the Coriolis term is relevant, this term can be added to our method by following [19]. Various methods can be used to incorporate free-surface effects; in this work, we have taken a rigid-lid approximation.

We assume constant dynamic viscosity, μ, and take the kinematic viscosity, ν, as constant. We note that this makes the viscous-term Boussinesq, and makes the method not a pure variable-density formulation as in [3]. In future work this approximation can be eliminated with a minor extension following [3], where $\nu \equiv \mu/\rho(\mathbf{x}, t)$. We note that these Boussinesq errors are very small when density variations are small and even smaller when the Reynolds number is large. It is important to note that the governing equations are non-Boussinesq for all but the viscous term, and as such, they eliminate inviscid Boussinesq errors. This non-Boussinesq capability enables the method to go beyond the limits of existing Boussinesq-based solvers. In future work, the constant μ approximation will be relaxed by following the work of Puckett *et al.* [22], where the full viscous stress tensor was used in their variable-density formulation.

2.2. *Boundary conditions*

We define our flow domain as Ω, and the boundary of this domain as $\partial\Omega$. We decompose $\partial\Omega$ into domain boundaries (faces of a box) and embedded boundaries (complex geometry inside the box), as is indicated in Figure 1. We permit different boundary conditions for each variable ($\mathbf{u}$, p, ρ, and s).

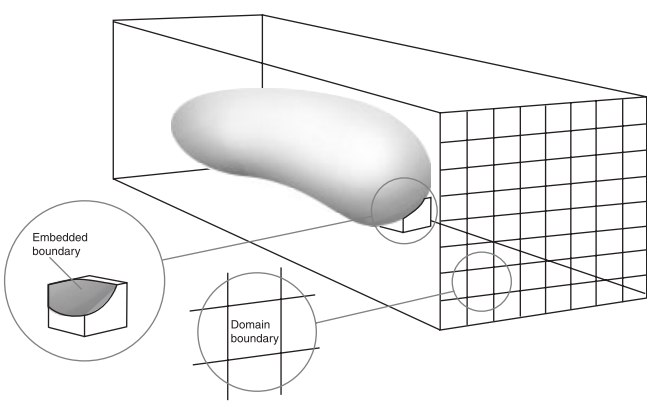

Figure 1. Embedded and domain boundary faces.

DOI: 10.1002/fld

We define $\mathbf{n}$ as the unit normal to the boundary which points into the fluid. We also define $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$ as unit vectors tangential to the boundary, where $\mathbf{n} \perp \boldsymbol{\tau}_1 \perp \boldsymbol{\tau}_2$. In the following we describe the boundary conditions in detail.

On solid surfaces the following velocity boundary conditions are permitted:

- Dirichlet for any solid surface (EB or domain boundary): $\mathbf{u} = \mathbf{b}$.
- Free-slip for domain walls: $\mathbf{u} \cdot \mathbf{n} = 0$, $\nabla(\mathbf{u} \cdot \boldsymbol{\tau}_1) \cdot \mathbf{n} = 0$ and $\nabla(\mathbf{u} \cdot \boldsymbol{\tau}_2) \cdot \mathbf{n} = 0$.

At inflow boundaries we specify $\mathbf{u}$. For outflow boundaries in the hyperbolic operators we specify the tangential velocity components by characteristic extrapolation normal to the domain face. For outflow boundaries in the viscous operators we specify a zero normal-gradient condition on velocity.

We prescribe pressure boundary conditions to be compatible with velocity boundary conditions and the governing equations. Specifically, for solid surfaces (where $\mathbf{u} \cdot \mathbf{n} = 0$) and at inflow boundary faces we prescribe,

$$\frac{\nabla p}{\rho} \cdot \mathbf{n} = \mathbf{g} \cdot \mathbf{n} \tag{5}$$

For outflow boundaries we prescribe hydrostatic pressure,

$$p = p_{\text{atm}} + \rho_{\text{o}} g z \tag{6}$$

where g is the gravitational acceleration, and where $z = 0$ is the top plane of Γ (see Γ in Section 2.2) in direction D. Recall that for this paper we take a 'rigid-lid' approach.

We prescribe scalar boundary conditions so as to be compatible with velocity boundary conditions. Note that density boundary conditions are treated the same as scalars. Specifically, for solid surfaces we prescribe no flux,

$$\nabla s \cdot \mathbf{n} = 0 \tag{7}$$

For inflow boundaries we specify scalars to be of some known value,

$$s = f(\mathbf{x}, t) \tag{8}$$

We extrapolate scalars normal to outflow boundaries in the hyperbolic operator.

2.3. Projection formulation

Following the work of Chorin [6–10] we shall use the Helmholtz–Hodge decomposition theorem. A vector field $\mathbf{w}$ on a region of space Ω can be uniquely decomposed in the form

$$\mathbf{w} = \mathbf{u} + \nabla \phi \tag{9}$$

$$\Delta \phi = \nabla \cdot \mathbf{w} \tag{10}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{11}$$

$$\nabla \phi \cdot \mathbf{n} = \mathbf{w} \cdot \mathbf{n} \quad \text{at } \partial\Omega \tag{12}$$

$$\mathbf{u} \cdot \mathbf{n} = 0 \quad \text{at } \partial\Omega \tag{13}$$

DOI: 10.1002/fld

With this decomposition we can define an orthogonal projection operator **P**, which maps **w** onto its divergence-free part **u**. The projection operator is the following linear operator:

$$\mathbf{P} \equiv \mathbf{I} - \mathbf{Q} \tag{14}$$

$$\mathbf{Q} \equiv \nabla \Delta^{-1} \nabla \tag{15}$$

where **I** is the identity matrix. Notice that

$$\mathbf{P}\mathbf{u} = \mathbf{u} \tag{16}$$

$$\mathbf{P}\nabla\phi = 0 \tag{17}$$

and

$$\mathbf{w} = \mathbf{P}\mathbf{w} + \nabla\phi \tag{18}$$

See [10] for a more rigorous mathematical discussion of the projection operator and the Helmholtz–Hodge decomposition.

For variable-density flows we follow [4] and utilize a density-weighted decomposition

$$\mathbf{w} = \mathbf{u} + \frac{\nabla\phi}{\rho} \tag{19}$$

The density-weighted projection operator is the following linear operator:

$$\mathbf{P}_\rho \equiv \mathbf{I} - \mathbf{Q}_\rho \tag{20}$$

$$\mathbf{Q}_\rho \equiv \frac{1}{\rho} \nabla L_\rho^{-1} \nabla \tag{21}$$

where **I** is the identity matrix and where L_ρ is $\nabla \cdot (1/\rho)\nabla$. Notice that we still have

$$\mathbf{P}_\rho \mathbf{u} = \mathbf{u} \tag{22}$$

$$\mathbf{P}_\rho \frac{\nabla\phi}{\rho} = 0 \tag{23}$$

and

$$\mathbf{w} = \mathbf{P}_\rho \mathbf{w} + \frac{\nabla\phi}{\rho} \tag{24}$$

In Section 3.6 we will define discrete versions of the projection operator.

3. SPACE DISCRETIZATION

In this section we describe the discretization of space. In the first sub-section we define the EB finite-volume method for non-unit-aspect ratio complex geometry; this is followed by a sub-section that describes the block-structured AMR spatial description. Subsequently, we present detailed sub-sections that define the spatial discretizations needed for the partial differential equations (elliptic, parabolic, and hyperbolic) that compose the incompressible Navier–Stokes equations.

DOI: 10.1002/fld

3.1. EB spatial discretization

To discretize complex geometries we utilize the EB (or cut-cell) method [12, 13, 41, 42]. In the EB method, control volumes are formed by the intersection of the domain with Cartesian grid cells.

With the EB method, for the bulk of the flow, which is $O(N)$ control volumes, we compute on a regular Cartesian grid composed of rectangular parallelepiped (or cuboid) control volumes. We use an EB description for the $O(N^{(D-1)/D})$ control volumes that intersect the boundary. The advantages of an underlying rectangular grid are as follows: grid generation is tractable, with a straightforward coupling to block-structured, AMR; the discretization technology is proven (e.g. well-understood consistency theory for conservative finite differences); geometric multigrid for elliptic solvers is optimal; and away from boundaries, the method reduces to a standard conservative finite-difference method.

Example 2D and 3D EB control volumes are shown in Figure 2. In this figure the curves indicate the intersection of the exact boundary with a Cartesian cell. Note that to enable second-order-accurate methods for this work, we approximate face intersections using quadratic interpolants, but the EB method can be extended to arbitrary accuracy.

To discretize our conservation laws using embedded boundaries, all we need are the following quantities: volume fractions, area fractions, centroids, boundary areas, and boundary normals. Given these geometric quantities on a fine grid, we can compute the same quantities on coarser grids without reference to the original geometry (by geometric coarsening). This permits highly efficient, dynamic coarsening and refinement of arbitrarily complex geometry as the simulation progresses (e.g. AMR and multigrid).

Embedded boundaries are 'water-tight' by construction; i.e. if two control volumes share a face, they both have the same area fraction for that face. Unlike typical complex-geometry discretization methods, the EB control volumes naturally fit within parallelizable, disjoint-block data structures.

3.1.1. Geometry generation. The underlying discretization of space is given by rectangular control volumes on a Cartesian grid: $\Upsilon_{\mathbf{i}_d} = [\mathbf{i}_d \Delta x_d, (\mathbf{i}_d + \mathbf{u}_d)\Delta x_d]$, $\mathbf{i} \in \mathbb{Z}^D$, where D is the dimensionality of the problem, Δx_d is the mesh spacing in direction d, and $\mathbf{u}$ is the vector whose entries are all ones. In the case of a fixed, irregular domain Ω, the geometry is represented by the intersection of Ω with the Cartesian grid. We obtain control volumes $V_\mathbf{i} = \Upsilon_\mathbf{i} \cap \Omega$ and faces $A_{\mathbf{i} \pm 1/2\mathbf{e}_d}$, which are the intersection of $\partial V_\mathbf{i}$ with the coordinate planes $\{\mathbf{x} : x_d = (i_d \pm \frac{1}{2})\Delta x_d\}$. Here $\mathbf{e}_d$ is the unit vector in the d direction. We also define $A^\mathrm{B}_\mathbf{i}$ to be the intersection of the boundary of the irregular domain with the Cartesian control volume: $A^\mathrm{B}_\mathbf{i} = \partial\Omega \cap \Upsilon_\mathbf{i}$. We will assume here that there is a one-to-one correspondence between the control volumes and faces and the corresponding geometric entities on the underlying Cartesian grid. The description can be generalized to allow for boundaries whose width is less than the mesh spacing or boundaries with sharp trailing edges.

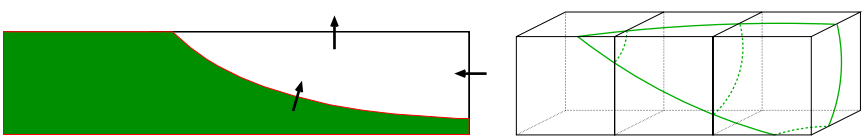

Figure 2. On the left is an example 2D embedded-boundary control volume; arrows indicate fluxes. On the right are example 3D embedded-boundary control volumes.

DOI: 10.1002/fld

In order to construct conservative finite-difference methods, we will need only a small number of real-valued quantities that are derived from geometric objects.

- Areas and volumes are expressed in these dimensionless terms:

$$\text{Volume fractions: } \kappa_{\mathbf{i}} = |V_{\mathbf{i}}| \prod_{d=1}^{D} \frac{1}{\Delta x_d} \tag{25}$$

$$\text{Face apertures: } \alpha_{\mathbf{i}\pm(1/2)\mathbf{e}_d} = |A_{\mathbf{i}\pm(1/2)\mathbf{e}_d}| \prod_{d' \neq d} \frac{1}{\Delta x_{d'}} \tag{26}$$

$$\text{Boundary apertures: } \alpha_{\mathbf{i}}^{\mathrm{B}} = |A_{\mathbf{i}}^{\mathrm{B}}| \Delta x_1^{-(D-1)} \tag{27}$$

  We assume that we can compute estimates of the dimensionless quantities that are sufficiently accurate (i.e. $O(\Delta x^2)$ for apertures, $O(\Delta x)$ for volume fractions).
- The locations of centroids, and the average outward normal to the boundary, are given exactly by

$$\text{Face-centroid: } \mathbf{x}_{\mathbf{i}+(1/2)\mathbf{e}_d} = \frac{1}{|A_{\mathbf{i}+(1/2)\mathbf{e}_d}|} \int_{A_{\mathbf{i}+(1/2)\mathbf{e}_d}} \mathbf{x}\, \mathrm{d}A \tag{28}$$

$$\text{Boundary face-centroid: } \mathbf{x}_{\mathbf{i}}^{\mathrm{B}} = \frac{1}{|A_{\mathbf{i}}^{\mathrm{B}}|} \int_{A_{\mathbf{i}}^{\mathrm{B}}} \mathbf{x}\, \mathrm{d}A \tag{29}$$

$$\text{Outward normal: } \mathbf{n}_{\mathbf{i}}^{\mathrm{B}} = \frac{1}{|A_{\mathbf{i}}^{\mathrm{B}}|} \int_{A_{\mathbf{i}}^{\mathrm{B}}} \mathbf{n}^{\mathrm{B}}\, \mathrm{d}A \tag{30}$$

  where $\mathbf{n}^{\mathrm{B}}$ is the outward normal to $\partial\Omega$, defined for each point on $\partial\Omega$. Again, we assume that we can compute estimates of these quantities that are accurate to $O(\Delta x^2)$.

Using just these quantities, we can define conservative discretizations for all the required operators.

3.1.2. EB examples. Now we present example visualizations to illustrate our ability to approximate complex geometry using the EB method. It is important to note that in this paper we visualize the embedded boundaries as piecewise planar (due to our graphics software), when in fact we use piecewise quadratic approximations for face intersections. Our first example is generated from imaging data of a coral (see Figure 3). This example illustrates our ability to handle arbitrarily complex geometry in a completely automated and efficient way given a distance function (for this example the distance function is given by a CT scan). In the next example (see Figure 4) we present South China Sea bathymetry with AMR boxes (where each box can be assigned to a unique processor in a parallel simulation).

3.2. AMR spatial discretization

Now we describe block-structured AMR in more detail. Our approach will be to express the AMR discretizations in terms of the corresponding uniform grid discretizations at each level (see Sections 3.3–3.5), followed by the appropriate conservative, second-order-accurate inter-level operators that make multilevel discretizations possible (see Section 4.2).

DOI: 10.1002/fld

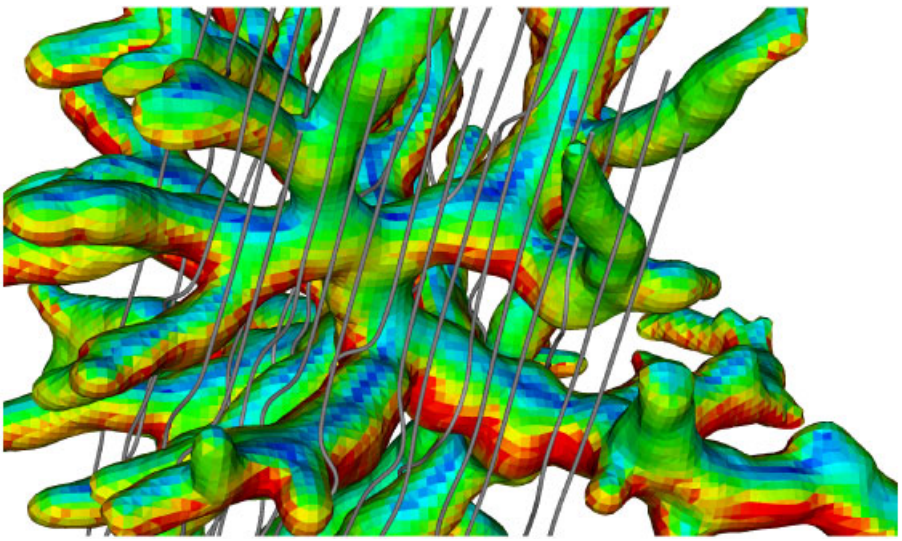

Figure 3. Flow past a coral as computed with the numerical method. Streamtubes indicate flow direction (flow is from top to bottom) and the surface colors indicate current speed near the coral (blue is slow, red is fast). Thanks to Dr Jaap A. Kaandorp for the Madracis Mirabilis CT scan data.

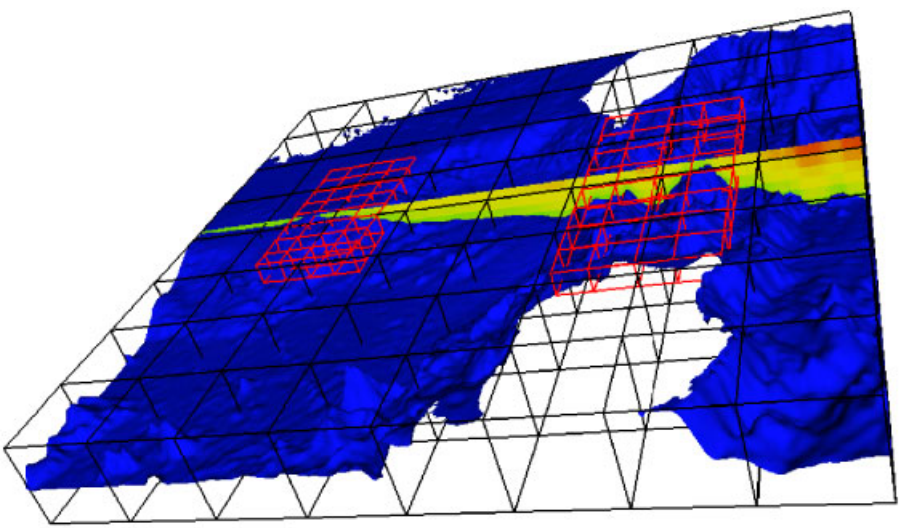

Figure 4. EB approximation of the South China Sea. Black boxes are a decomposition of the coarsest level; red boxes are finer grids. Each box is further sub-divided into individual control volumes. The red boxes refine both the Luzon Strait vicinity (right) and the Dongsha Island region (left). Upper right is Taiwan; lower right is the Philippines; mainland China is upper left. The East–West slice is colored by distance to the bed.

DOI: 10.1002/fld

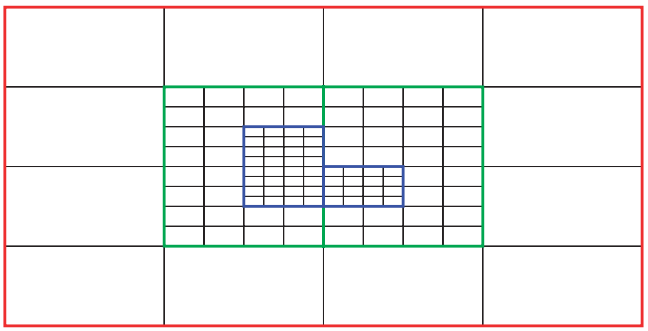

Figure 5. Block-structured adaptive mesh refinement.

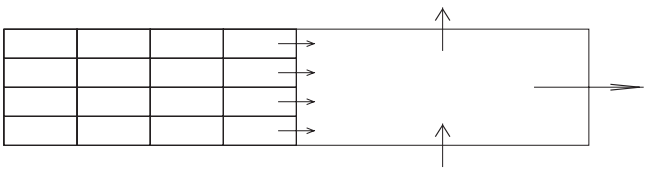

Figure 6. Flux matching at coarse–fine interfaces.

For this paper we have imposed the constraint that the EB interface cannot cross the coarse–fine interface. This additional constraint is for ease of implementation and will be relaxed in future work, similar to [41]. A further constraint imposed for this paper is that all AMR levels are advanced with the same time step; i.e. we are not subcycling the levels as is done in [3, 15, 16]. In the AMR research community, methods that do not subcycle AMR levels are termed composite methods, and as such, 'mass' conservation is easily maintained with a flux-based approach. See [15] for subcycling issues.

In adaptive methods, one adjusts the computational effort locally to maintain a uniform level of accuracy throughout the problem domain. Refined regions are organized into rectangular patches, as in Figures 4 and 5. Refinement is possible in both space and time (though, as stated above, for this work all levels use the same time step). AMR allows the simulation of a range of spatial and temporal scales.

Three basic requirements are necessary to maintain conservation and accuracy with AMR: (1) match fluxes (see Figure 6) conservatively at coarse–fine interfaces (this leads to a refluxing step for the coarse levels); (2) use interpolation to provide ghost cell values for points in the stencil extending outside of the grids at that level; (3) conservatively coarsen and refine data when regridding. We maintain accuracy and strict conservation with embedded boundaries and AMR.

3.3. Divergence of fluxes

Our finite-volume discretizations are formulated in a divergence form, and as such we need to define a conservative divergence operator for discrete fluxes at face-centroids (higher than second-order-accurate finite-volume methods need face-averaged fluxes [48]).

DOI: 10.1002/fld

Using just the geometric quantities from Section 3.1.1, we can define conservative discretizations for the divergence operator. Let $\mathbf{F}=(F^1,\ldots,F^D)$ be a function of $\mathbf{x}$. Using the divergence theorem we have

$$\begin{aligned}\nabla\cdot\mathbf{F} &\approx \frac{1}{|V_\mathbf{i}|}\int_{V_\mathbf{i}}\nabla\cdot\mathbf{F}\,\mathrm{d}V=\frac{1}{|V_\mathbf{i}|}\int_{\partial V_\mathbf{i}}\mathbf{F}\cdot\mathbf{n}\,\mathrm{d}A\\ &\approx \frac{1}{|V_\mathbf{i}|}\left[\left(\sum_{\pm=+,-}\sum_{d=1}^{D}\pm|A_{\mathbf{i}\pm(1/2)\mathbf{e}^d}|F^d(\mathbf{x}_{\mathbf{i}\pm(1/2)\mathbf{e}^d})\right)+|A_\mathbf{i}^\mathrm{B}|\mathbf{n}_\mathbf{i}^\mathrm{B}\cdot\mathbf{F}(\mathbf{x}_\mathbf{i}^\mathrm{B})\right]\end{aligned}\tag{31}$$

where (31) is obtained by replacing the integrals of the normal components of the vector field $\mathbf{F}$ with the values at the face-centroids. It is this form that we use when conservatively discretizing divergence operators.

3.4. Elliptic and parabolic equations

The projections and predictor step require the solution of elliptic and parabolic partial differential equations. The density-weighted projections require the solution of: $\nabla\cdot(1/\rho)\nabla\phi=f$, while the parabolic predictor step requires the solution of $\phi_t=\mu\Delta\phi+S$. In this sub-section we define our implicit solution methodology for the elliptic equations. We use the constant coefficient, parabolic methodology developed in [13] for the parabolic equation solves. A numerical convergence study for the parabolic methodology is given in Section 5.2.3.

Now we define our discretizations of the variable-coefficient, elliptic, partial differential equation on an irregular domain Ω. It is this variable-coefficient elliptic equation that we use in the density-weighted projections (DWPM1 and DWPM2) of Section 3.6.2. The PDE is as follows:

$$\begin{aligned}\nabla\cdot\beta\nabla\psi &= f \quad \text{on } \Omega\\ \beta\frac{\partial\psi}{\partial n} &= g_N \quad \text{on } \partial\Omega\end{aligned}\tag{32}$$

or

$$\psi=g_D \quad \text{on } \partial\Omega$$

where g_D and g_N are Dirichlet and Neumann boundary conditions. To numerically solve (32) we need to spatially discretize ψ, and so we define a discrete variable ϕ, $\phi_\mathbf{i}\approx\psi(\mathbf{i}\Delta x)$. Using the discretization of the divergence defined in (31), we can define a discretization of the variable-coefficient elliptic equation as follows:

$$\nabla\cdot\beta\nabla\phi\approx L^h[\phi,\beta]_\mathbf{i}=f_\mathbf{i}\tag{33}$$

$$L^h[\phi,\beta]_\mathbf{i}=\frac{1}{|V_\mathbf{i}|}\left[\left(\sum_{\pm=+,-}\sum_{d=1}^{D}\pm|A_{\mathbf{i}\pm(1/2)\mathbf{e}^d}|F^d_{\mathbf{i}\pm(1/2)\mathbf{e}^d}\right)+|A_\mathbf{i}^B|F^B\right]\tag{34}$$

where $f_\mathbf{i}=f(\mathbf{i}\Delta x)$ and the fluxes F^d and F^B are linear combinations of $\phi_\mathbf{i}$, $\beta_\mathbf{i}$ and the boundary values ($\mathbf{F}=\beta\nabla\phi$). We avoid problems arising from arbitrarily small values of $|V_\mathbf{i}|$ in the denominator (of (34)) by recalling (25) and solving

$$\kappa_\mathbf{i}L^h[\phi,\beta]_\mathbf{i}=\kappa_\mathbf{i}f_\mathbf{i}\tag{35}$$

DOI: 10.1002/fld

When $D=3$ the fluxes ($F^d_{\mathbf{i}+(1/2)\mathbf{e}^1}$, in direction d) are given by bilinear interpolation of centered differences (or linear when $D=2$). Explicitly, bilinear interpolation of fluxes can be written as an iteration of linear interpolation of fluxes in the two directions that are not normal to the face. For example, given the face with outward unit normal $\mathbf{e}^1$, with centroid $\mathbf{x}$, define the linearly interpolated flux in the d ($d\neq 1$) direction by

$$
\begin{aligned}
F^d_{\mathbf{i}+(1/2)\mathbf{e}^1} &= \eta \frac{\beta_{\mathbf{i}}+\beta_{\mathbf{i}+\mathbf{e}^1}}{2}\frac{(\phi_{\mathbf{i}+\mathbf{e}^1}-\phi_{\mathbf{i}})}{\Delta x_1} + (1-\eta)\frac{\beta_{\mathbf{i}+\mathbf{e}^1\pm\mathbf{e}^d}+\beta_{\mathbf{i}\pm\mathbf{e}^d}}{2}\frac{(\phi_{\mathbf{i}+\mathbf{e}^1\pm\mathbf{e}^d}-\phi_{\mathbf{i}\pm\mathbf{e}^d})}{\Delta x_1} \\
\eta &= 1-\frac{|\mathbf{x}\cdot\mathbf{e}^d|}{\Delta x_d} \\
\pm &= \begin{cases} +, & \mathbf{x}\cdot\mathbf{e}^d>0 \\ -, & \mathbf{x}\cdot\mathbf{e}^d\leqslant 0 \end{cases}
\end{aligned}
\tag{36}
$$

The bilinear interpolation of the flux for the face with normal $\mathbf{e}^1$ is as in [13]. The boundary conditions are also as in [13], but are subject to (32).

One of the target application areas for this research is studying high-aspect-ratio geophysical flows. These flows typically have horizontal scales that are larger than vertical scales. When solving elliptic partial differential equations for large-aspect-ratio systems it is common to reduce the total number of grid points by setting $\Delta x\neq\Delta z\neq\Delta y$. When the aspect ratio is more than (roughly) 2, traditional multigrid smoothers fail to dampen high-frequency errors and multigrid convergence stalls. This is a well-known problem, and can be easily seen by noting that the elliptic stencil becomes dimensionally decoupled when $\Delta x\gg\Delta y$, see [60]. There are two 'fixes' that are traditionally employed: (1) anisotropic coarsening in multigrid, (2) line solving, or (3) both [60]. For our approach in this research we chose to implement a line solver that is both AMR and EB aware.

We extend our single-grid solution methodology [13] for solving (32), and the heat equation, by using essentially the same second-order-accurate (non-EB) AMR elliptic algorithm as is outlined in [15, 16, 61].

When we have anisotropic control volumes we use a line solver for multigrid smoothing; otherwise we use a colored Gauss–Siedel smoother. For the line solver we have an additional requirement that our disjoint-box layout has one box per (connected) line per level (with as many grids in the horizontal as needed). In the elliptic solve we use a residual-correction form and impose homogeneous boundary conditions at the coarse–fine interface.

A pseudo-code description of our line-smoother algorithm is given in Figure 7. For the line-smoother algorithm we initialize the correction δ to zero; then we evaluate our horizontal elliptic stencil (L^{h}_{xy}) and compute a new right-hand side (g) where $\lambda=1/4\mathrm{diag}(L^{\mathrm{h}})$ and where $(I-\lambda L^{\mathrm{h}}_{xy})0=0$. Subsequently, we do a tridiagonal solve using our new right-hand side ($g=\lambda R$). Then we apply the correction to e (where e is a correction since we are already in the residual-correction form). We use the Thomas Algorithm, a simplified version of Gaussian elimination for tridiagonal systems, when doing the tridiagonal solve. Where we have covered cells (i.e. $\kappa=0$) we set the diagonal term to 1 and the two off-diagonal terms to zero. This decouples covered cells from the non-covered control volumes and permits the line solver to handle arbitrarily complex geometries. A validation of the line solver is given in Section 5.2.2 and [26].

DOI: 10.1002/fld

```
procedure LineSmoother(e, R)
    {
        δ = 0
        g = (I − λL^h_xy)δ + λR
        δ = (I + λL^h_z)^{−1} g
        e = e + δ
    }
```

Figure 7. Pseudo-code description of the line smoother algorithm.

3.5. Hyperbolic equations

We use an explicit second-order-accurate, stable hyperbolic methodology for computing discrete approximations to advective terms. The methodology was developed in [41, 62], and we make a few minor deviations from [41] for incompressible flow. Specifically, we use the single-level hyperbolic methodology with flux matching at the coarse–fine interface, and conservative linear interpolation from coarse levels to fine ghost cells. The method is stable (and accurate) in the absence of physical diffusion for both the transport of momentum (i.e. the Euler equations) and for pure scalar advection.

The procedure for computing the advection terms is outlined as follows:

1. Extrapolate the advective normal velocities to face-centers at $t^{n+1/2}$.
2. MAC-project the advective normal velocities to obtain divergence-free advective velocities.
3. Extrapolate the remaining quantities—tangential velocities, density (ρ) and scalars (s)—to face-centers at $t^{n+1/2}$.
4. Compute the advective terms as a combination of stable and conservative approximations.

In the following sub-sections we describe this in detail. A numerical validation of the hyperbolics is given in Section 5.2.1.

3.5.1. Extrapolate advective velocities to face-centers at $n+\frac{1}{2}$. Now we describe the Godunov methodology used to compute second-order-accurate approximations to the advective velocities. The method is a characteristic extrapolation procedure. For a given scalar s we extrapolate, in direction d, to the low (L) side of the $\mathbf{i}+\frac{1}{2}\mathbf{e}_d$ face-center, and at time $n+\frac{1}{2}$, by

$$\tilde{s}^{L,n+1/2}_{\mathbf{i}+(1/2)\mathbf{e}_d} = s^n_{\mathbf{i}} + \frac{\Delta x_d}{2}\frac{\partial s}{\partial x_d}\bigg|^n_{\mathbf{i}} + \frac{\Delta t}{2}\frac{\partial s}{\partial t}\bigg|^n_{\mathbf{i}} \tag{37}$$

This is a Taylor expansion in time and space. Similarly for component d' of the velocity we extrapolate in the d-direction to the low (L) side of the $\mathbf{i}+\frac{1}{2}\mathbf{e}_d$ face-center,

$$\tilde{u}^{L,n+1/2}_{d',\mathbf{i}+(1/2)\mathbf{e}_d} = u^n_{d',\mathbf{i}} + \frac{\Delta x_d}{2}\frac{\partial u_{d'}}{\partial x_d}\bigg|^n_{\mathbf{i}} + \frac{\Delta t}{2}\frac{\partial u_{d'}}{\partial t}\bigg|^n_{\mathbf{i}} \tag{38}$$

DOI: 10.1002/fld

and on the high (H) side of the $\mathbf{i}-\frac{1}{2}\mathbf{e}_d$ face-center,

$$\tilde{u}^{H,n+1/2}_{d',\mathbf{i}-(1/2)\mathbf{e}_d} = u^n_{d',\mathbf{i}} - \frac{\Delta x_d}{2}\frac{\partial u_{d'}}{\partial x_d}\bigg|^n_{\mathbf{i}} + \frac{\Delta t}{2}\frac{\partial u_{d'}}{\partial t}\bigg|^n_{\mathbf{i}} \tag{39}$$

Since we are extrapolating advective components only, we evaluate (38) and (39) with $d'=d$. We approximate r-order-accurate spatial derivatives in direction d by

$$\frac{\partial \psi}{\partial x_d}\bigg|^n_{\mathbf{i}} \approx \psi^{\lim,r}_{x_d,\mathbf{i}} \tag{40}$$

where the limited slopes are defined in Section 3.5.4. For these advective component extrapolations we define preliminary advective velocities by interpolating to face-centers

$$\mathbf{u}^{\text{pre}}_{\mathbf{i}+(1/2)\mathbf{e}_d} = I^{\mathbf{i}}_{\mathbf{i}+(1/2)\mathbf{e}_d}\mathbf{u}^n_{\mathbf{i}} \tag{41}$$

We then approximate the time-derivative term for each direction d,

$$\frac{\partial u_d}{\partial t}\bigg|^n_{\mathbf{i}} \approx H^n_{d,\mathbf{i}} - \sum_{d'=1}^{D} u^{\text{ave}}_{d',\mathbf{i}}(u_d)^{\lim,r}_{x_{d'},\mathbf{i}} \tag{42}$$

where

$$u^{\text{ave}}_{d,\mathbf{i}} = \tfrac{1}{2}(u^{\text{pre}}_{d,\mathbf{i}+(1/2)\mathbf{e}_d} + u^{\text{pre}}_{d,\mathbf{i}-(1/2)\mathbf{e}_d}) \tag{43}$$

the transverse ($d'\neq d$) derivatives are upwinded and where we define the source term of the dth component,

$$H^n_{d,\mathbf{i}} \equiv \frac{1}{\rho^n}L[u^n_d,\mu]_{\mathbf{i}} - \left[\frac{\nabla^d p}{\rho}\right]^{n-1/2}_{\mathbf{i}} + \mathbf{g}^n_{d,\mathbf{i}} \tag{44}$$

Equations (38) and (39) yield the advective velocities on the low and high side of face-centers. For covered faces in irregular cells we extrapolate both the extrapolations, (38) and (39), and the preliminary advective velocities (41) to covered faces [41].

Now we solve a Riemann problem by upwinding to choose the state for each direction d advective velocity,

$$\tilde{u}^{n+1/2}_{d,\mathbf{i}+(1/2)\mathbf{e}_d} = R(\tilde{u}^{L,n+1/2}_{d,\mathbf{i}+(1/2)\mathbf{e}_d}, \tilde{u}^{H,n+1/2}_{d,\mathbf{i}+(1/2)\mathbf{e}_d}, u^{\text{pre}}_{d,\mathbf{i}+(1/2)\mathbf{e}_d}) \tag{45}$$

where R is defined in (65).

We MAC-project $\tilde{\mathbf{u}}^{n+1/2}_{\mathbf{i}+(1/2)\mathbf{e}_d}$. This results in a divergence-free $\mathbf{u}^{\text{ADV}}_{\mathbf{i}+(1/2)\mathbf{e}_d}$ advective velocity field,

$$\mathbf{u}^{\text{ADV}}_{\mathbf{i}+(1/2)\mathbf{e}_d} = \mathbf{P}^{\text{mac}}_{\rho}(\tilde{\mathbf{u}}^{n+1/2}_{\mathbf{i}+(1/2)\mathbf{e}_d}) \tag{46}$$

where $\mathbf{P}^{\text{mac}}_{\rho}$ is defined in (72).

3.5.2. Extrapolate remaining quantities to face-centers at $n+\frac{1}{2}$. We are now in a position to extrapolate the remaining quantities to half time at face-centers. We do this similarly to how we extrapolated the advective velocities. However, now we use the divergence-free advective velocities

DOI: 10.1002/fld

at face-centroids, with a non-conservative (but stable) discretization of the advective terms in the time-derivative approximation. Here we present the method for a generic scalar s, which will represent the tangential velocities, density, or passive scalars that we are advecting with the flow. We approximate the spatial derivatives in (37) using our limited spatial-derivatives method

$$\left.\frac{\partial s}{\partial x_d}\right|_{\mathbf{i}}^{n} \approx s_{x_d,\mathbf{i}}^{\lim,r} \tag{47}$$

see Equation (63). For the time derivative we use the face-centered advective velocities

$$\left.\frac{\partial s}{\partial t}\right|_{\mathbf{i}}^{n} \approx H_{\mathbf{i}}^{n} - \sum_{d=1}^{D} u_{d,\mathbf{i}}^{\text{ave}} s_{x_d,\mathbf{i}}^{\lim,r} \tag{48}$$

where

$$u_{d,\mathbf{i}}^{\text{ave}} = \tfrac{1}{2}(u_{d,\mathbf{i}+(1/2)\mathbf{e}_d}^{\text{ADV}} + u_{d,\mathbf{i}-(1/2)\mathbf{e}_d}^{\text{ADV}}) \tag{49}$$

and where we define the source term,

$$H_{\mathbf{i}}^{n} = L[s^n, k_s]_{\mathbf{i}} + H_{s,\mathbf{i}}^{n} \tag{50}$$

In Equation (50), L is the Laplacian operator with diffusion constant k_s. For covered faces in irregular cells we extrapolate both the extrapolations (37) (high or low, depending on what is needed) and the advective velocities to covered faces using the method of Colella *et al*. [41].

Subsequently, we solve a Riemann problem for each face direction (d) using the divergence-free face-centered velocities $\mathbf{u}_{\mathbf{i}+(\mathbf{1}/\mathbf{2})\mathbf{e_d}}^{\text{ADV}}$,

$$\tilde{s}_{\mathbf{i}+(1/2)\mathbf{e}_d}^{n+1/2} = R(\tilde{s}_{\mathbf{i}+(1/2)\mathbf{e}_d}^{L,n+1/2}, \tilde{s}_{\mathbf{i}+(1/2)\mathbf{e}_d}^{H,n+1/2}, (u_d)_{\mathbf{i}+(\mathbf{1}/\mathbf{2})\mathbf{e_d}}^{\text{ADV}}) \tag{51}$$

where R is defined in (65). For tangential velocities at face-centers (including covered faces) we enforce a divergence-free condition by correcting with the MAC-gradients

$$u_{\mathbf{i}+(1/2)\mathbf{e}_d}^{d'} = u_{\mathbf{i}+(1/2)\mathbf{e}_d}^{*,d'} - \left[\frac{\nabla^{d'}\phi}{\rho^{n+1/2}}\right]_{\mathbf{i}+(1/2)\mathbf{e}_d}$$

Now we have $\tilde{s}_{\mathbf{i}+(1/2)\mathbf{e}_d}^{n+(1/2)}$ at regular, irregular, and needed covered face-centers.

3.5.3. Compute advective terms. Since the advective velocities are exactly discretely divergence-free, i.e.

$$D^{\text{mac}} \cdot \mathbf{u}^{\text{ADV}} = 0 \tag{52}$$

then the discrete analog of

$$(\mathbf{u}\cdot\nabla)c = \nabla\cdot(c\mathbf{u}) = \nabla\cdot\mathbf{F} \tag{53}$$

is true. We will proceed by selectively using (53) to conservatively discretize the advective terms. Following [41], we compute a stable (S) hybridization of conservative (C) and non-conservative (NC) divergences,

$$(D\cdot\mathbf{F})_{\mathbf{i}}^{\text{S,NC}} = \kappa_{\mathbf{i}}(D\cdot\mathbf{F})_{\mathbf{i}}^{\text{C}} + (1-\kappa_{\mathbf{i}})(D\cdot\mathbf{F})_{\mathbf{i}}^{\text{NC}} \tag{54}$$

DOI: 10.1002/fld

where

$$(D\cdot\mathbf{F})_{\mathbf{i}}^{\mathrm{NC}}=\sum_{\pm=+,-}\sum_{d=1}^{D}\frac{\pm\mathbf{F}_{\mathbf{i}+(1/2)\mathbf{e}^d}}{\Delta x_d} \tag{55}$$

and where $(D\cdot\mathbf{F})_{\mathbf{i}}^{\mathrm{C}}$ is defined below. Note that for (55) we use covered face values for faces with zero apertures, where the covered face values are obtained by extrapolating from the interior as in [41].

For velocity at irregular control volumes and for scalars at all control volumes we define $(D\cdot\mathbf{F})_{\mathbf{i}}^{\mathrm{C}}$ using the flux-based divergence of (31). For velocity component d' at regular control volumes we define $(D\cdot\mathbf{F})_{\mathbf{i}}^{\mathrm{C}}$ as follows:

$$(D\cdot\mathbf{F})_{\mathbf{i}}^{\mathrm{C}}=\sum_{d=1}^{D}\left[\frac{\mathbf{u}_{d,\mathbf{i}+(1/2)\mathbf{e}^d}+\mathbf{u}_{d,\mathbf{i}-(1/2)\mathbf{e}^d}}{2}\left(\frac{\mathbf{u}_{d',\mathbf{i}+(1/2)\mathbf{e}^d}-\mathbf{u}_{d',\mathbf{i}-(1/2)\mathbf{e}^d}}{\Delta x_d}\right)\right] \tag{56}$$

Notice that κ in the denominator of (31) cancels with the κ of (54), eliminating small κ division issues.

A non-stable (NS) but conservative update for density would be

$$\rho_{\mathbf{i}}^{n+1,\mathrm{NS,C}}=\rho_{\mathbf{i}}^{n}-\Delta t\,(D\cdot\mathbf{F})_{\mathbf{i}}^{\mathrm{C}} \tag{57}$$

while a stable but non-conservative update would be

$$\rho_{\mathbf{i}}^{n+1,\mathrm{S,NC}}=\rho_{\mathbf{i}}^{n}-\Delta t\,(D\cdot\mathbf{F})_{\mathbf{i}}^{\mathrm{S,NC}} \tag{58}$$

We would like to update (58), but we need to account for the missing mass. The missing mass is (57) minus (58) scaled by the volume, and using (54) is

$$\delta M_{\mathbf{i}}=\Delta t\,|V_{\mathbf{i}}|(1-\kappa_{\mathbf{i}})[(D\cdot\mathbf{F})_{\mathbf{i}}^{\mathrm{NC}}-(D\cdot\mathbf{F})_{\mathbf{i}}^{\mathrm{C}}] \tag{59}$$

To enforce exact mass conservation, we use volume-weighted redistribution of this mass to neighboring control volumes

$$\rho_{\mathbf{i}'}^{n+1,\mathrm{S,C}}=\rho_{\mathbf{i}'}^{n}-\Delta t\,(D\cdot\mathbf{F})_{\mathbf{i}'}^{\mathrm{S,NC}}+\frac{\kappa_{\mathbf{i}'}}{\sum_{\mathbf{i}''\in N(\mathbf{i})}\kappa_{\mathbf{i}''}}\frac{\delta M_{\mathbf{i}}}{|V_{\mathbf{i}'}|} \tag{60}$$

or

$$(D\cdot\mathbf{F})_{\mathbf{i}'}^{\mathrm{S,C}}=(D\cdot\mathbf{F})_{\mathbf{i}'}^{\mathrm{S,NC}}-\frac{\kappa_{\mathbf{i}'}}{\sum_{\mathbf{i}''\in N(\mathbf{i})}\kappa_{\mathbf{i}''}}\frac{\delta M_{\mathbf{i}}}{|V_{\mathbf{i}'}|\Delta t} \tag{61}$$

More generally (for velocity too),

$$(D\cdot\mathbf{F})_{\mathbf{i}'}^{\mathrm{S,C}}=(D\cdot\mathbf{F})_{\mathbf{i}'}^{\mathrm{S,NC}}-\frac{1}{\sum_{\mathbf{i}''\in N(\mathbf{i})}\kappa_{\mathbf{i}''}}\kappa_{\mathbf{i}}(1-\kappa_{\mathbf{i}})[(D\cdot\mathbf{F})_{\mathbf{i}}^{\mathrm{NC}}-(D\cdot\mathbf{F})_{\mathbf{i}}^{\mathrm{C}}] \tag{62}$$

where $\mathbf{i}'\in N(\mathbf{i})$, and where $N(\mathbf{i})$ is a set of indices whose components differ from those of $\mathbf{i}$ by no more than one and can be reached by a monotonic path.

DOI: 10.1002/fld

3.5.4. Limited-slope computation. Here we define the limited-slope operator in direction d, used in (40), (42), (47), and (48). The second-order ($r=2$) limited-slope calculation is

$$\psi^{\lim,2}_{x_d,\mathbf{i}} = \begin{cases} \dfrac{\operatorname{sign}(\psi_{\mathbf{i}+\mathbf{e}_d}-\psi_{\mathbf{i}-\mathbf{e}_d})\min(2|\psi_{\mathbf{i}+\mathbf{e}_d}-\psi_{\mathbf{i}}|, 2|\psi_{\mathbf{i}}-\psi_{\mathbf{i}-\mathbf{e}_d}|, \frac{1}{2}|\psi_{\mathbf{i}+\mathbf{e}_d}-\psi_{\mathbf{i}-\mathbf{e}_d}|)}{\Delta x_d}, & b>0 \\ 0, & b\leqslant 0 \end{cases} \tag{63}$$

where

$$b=(\psi_{\mathbf{i}+\mathbf{e}_d}-\psi_{\mathbf{i}})\cdot(\psi_{\mathbf{i}}-\psi_{\mathbf{i}-\mathbf{e}_d}) \tag{64}$$

The fourth-order ($r=4$) accurate slope calculation is as in [41]. Using slope limiters allows the method to robustly handle sharp gradients, with only a local degradation in accuracy.

3.5.5. Upwinding. Our Riemann solution in (45) and (51) is simple upwinding,

$$R(\psi^{\mathrm{L}},\psi^{\mathrm{H}},u_d)=\begin{cases} \psi^{\mathrm{L}} & \text{if } u_d>0 \\ \psi^{\mathrm{H}} & \text{if } u_d<0 \\ \dfrac{1}{2}(\psi^{\mathrm{L}}+\psi^{\mathrm{H}}) & \text{if } u_d=0 \end{cases} \tag{65}$$

3.6. Discretizing projections

We need to define projections for two data types: face-centered and cell-centered. The face-centered operator $\mathbf{P}^{\mathrm{mac}}_{\rho}$ is defined in Section 3.6.1, while the cell-centered projection operator $\mathbf{P}^{\mathrm{cc}}_{\rho}$ is defined in Section 3.6.2.

Velocity boundary conditions for the velocity operators are specified in Section 2.2, and pressure boundary conditions for the pressure operators are specified in Section 2.2. Note that the pressure difference ($p^{n+1/2}-p^{n-1/2}$) elliptic solves (in DWPM2) have a desirable homogeneous boundary condition at outflow, eliminating the need to approximate a pressure (as is the case for DWPM1 at outflow).

3.6.1. Face-centered MAC projection. Now we define a projection that is based on face-centered advective velocities. This is used to correct the advective velocity (e.g. in $\mathbf{u}\cdot\nabla\mathbf{u}$) as was noted at the beginning of this section. The Hodge decomposition of the advective velocities is

$$u^{*,d}_{\mathbf{i}+(1/2)\mathbf{e}_d}=u^{d}_{\mathbf{i}+(1/2)\mathbf{e}_d}+\left[\frac{\nabla^d\phi}{\rho}\right]_{\mathbf{i}+(1/2)\mathbf{e}_d} \tag{66}$$

We first define the discrete operators needed for the projection; then we define the face-centered projection operator.

Face-centered divergence: We define face-centered vector fields $\mathbf{F}=(F_1,\ldots,F_{\mathbf{D}})$, such that $\mathbf{F}_{\mathbf{i}+(1/2)\mathbf{e}_d}$ is at face-centers. For regular control volumes (non-EB) we define a discretized divergence operator on such a vector field as follows:

$$D^{\mathrm{mac}}\cdot\mathbf{F}\equiv\frac{1}{|V_{\mathbf{i}}|}\left[\left(\sum_{\pm=+,-}\sum_{d=1}^{D}\pm|A_{\mathbf{i}\pm(1/2)\mathbf{e}^d}|F^d(\mathbf{x}_{\mathbf{i}\pm(1/2)\mathbf{e}^d})\right)\right] \tag{67}$$

DOI: 10.1002/fld

where (67) is obtained by replacing the integrals of the normal components of the vector field $\mathbf{F}$ with the values at face-centers. For irregular control volumes we define a discretized divergence operator on such a vector field as follows:

$$D^{\mathrm{mac}} \cdot \mathbf{F} \equiv \frac{1}{|V_{\mathbf{i}}|} \left[\left(\sum_{\pm=+,-} \sum_{d=1}^{D} \pm |A_{\mathbf{i}\pm(1/2)\mathbf{e}^d}| F^d(\mathbf{x}_{\mathbf{i}\pm(1/2)\mathbf{e}^d}) \right) + |A_{\mathbf{i}}^B| \mathbf{n}_{\mathbf{i}}^B \cdot \mathbf{F}(\mathbf{x}_{\mathbf{i}}^B) \right] \tag{68}$$

where (68) is obtained by replacing the integrals of the normal components of the vector field $\mathbf{F}$ with the values at the face-centroids.

Face-centered gradient: We define the face-centered gradient $G^{\mathrm{mac},d}[\phi]_{\mathbf{i}+(1/2)\mathbf{e}_d}$ for each direction d using

$$G^{\mathrm{mac},d}[\psi]_{\mathbf{i}+(1/2)\mathbf{e}_d} = \frac{\psi_{\mathbf{i}+\mathbf{e}} - \psi_{\mathbf{i}}}{\Delta x_d} \tag{69}$$

To scale the MAC-gradient by density ($\rho_{\mathbf{i}+(1/2)\mathbf{e}_d}$) we divide as follows:

$$\left[\frac{G^{\mathrm{mac},d}\phi}{\rho} \right]_{\mathbf{i}+(1/2)\mathbf{e}_d} = (\phi_{x_d,\mathbf{i}+(1/2)\mathbf{e}_d}) \Big/ \left(\frac{\rho_{\mathbf{i}} + \rho_{\mathbf{i}+\mathbf{e}_d}}{2} \right) \tag{70}$$

For irregular faces, we use a linear interpolation operator $I_{\mathbf{fc}}^{\mathbf{i}+(1/2)\mathbf{e}_d}$ to move data from face-centers to face-centroids, see [13].

MAC-projection: Our face-centered MAC projection operator does the following:

$$u_{\mathbf{i}+(1/2)\mathbf{e}_d}^d = \mathbf{P}_{\rho}^{\mathrm{mac}}(u_{\mathbf{i}+(1/2)\mathbf{e}_d}^{*,d}) \tag{71}$$

We use the face-centered, discrete operators to define the MAC-projection

$$\mathbf{P}_{\rho}^{\mathrm{mac}} \equiv \mathbf{I} - \frac{1}{\rho} G^{\mathrm{mac}} \left[D^{\mathrm{mac}} \frac{1}{\rho} G^{\mathrm{mac}} \right]^{-1} D^{\mathrm{mac}} \tag{72}$$

The first step is to compute the MAC-divergence of (66) and then solve a variable-coefficient (see Section 3.4) elliptic equation for $\phi_{\mathbf{i}}$ given boundary conditions on ϕ (see Section 2.2),

$$D^{\mathrm{mac}} \left[\frac{1}{\rho_{\mathbf{i}+(1/2)\mathbf{e}_d}} G_d^{\mathrm{mac}} \phi_{\mathbf{i}} \right] = D^{\mathrm{mac}}[u_{\mathbf{i}+(\mathbf{1}/\mathbf{2})\mathbf{e_d}}^{*,d}] \tag{73}$$

We subsequently compute the divergence-free advective velocities

$$u_{\mathbf{i}+(\mathbf{1}/\mathbf{2})\mathbf{e_d}}^d = u_{\mathbf{i}+(\mathbf{1}/\mathbf{2})\mathbf{e_d}}^{*,d} - \left[\frac{G^{\mathrm{mac},d}\phi}{\rho} \right]_{\mathbf{i}+(\mathbf{1}/\mathbf{2})\mathbf{e_d}} \tag{74}$$

A numerical convergence study for the density-weighted MAC-projection is given in Section 5.2.4.

3.6.2. Cell-centered approximate projections. Now we define an approximate projection that is based on cell-centered discrete velocity data. This is used to correct the predictor velocity as

DOI: 10.1002/fld

was noted at the beginning of this section. Now our Hodge decomposition of the cell-centered velocities is

$$\mathbf{u}_{\mathbf{i}}^{*} = \mathbf{u}_{\mathbf{i}} + \left[\frac{\nabla\phi}{\rho}\right]_{\mathbf{i}} \tag{75}$$

Below we define the cell-centered discretizations needed to define our cell-centered projection operator.

Average cell to face: In what follows we need to interpolate the cell-centered $\mathbf{w}_{\mathbf{i}}^{*}$ (advective components only) using a second-order-accurate interpolant to face-centers (A), then second-order to face-centroids (I),

$$w_{\mathbf{i}+(1/2)\mathbf{e}_d}^{*,d} = I_{\mathbf{fc}}^{\mathbf{i}+(1/2)\mathbf{e}_d}(A^{\mathrm{C}\to\mathrm{F}}(\mathbf{w}^{*})) \tag{76}$$

So we need to define our averaging operator to move data from cell-centers to face-centers

$$[A^{\mathrm{C}\to\mathrm{F}}\psi]_{\mathbf{i}+(1/2)\mathbf{e}_d} \equiv \frac{\psi_{\mathbf{i}} + \psi_{\mathbf{i}+\mathbf{e}_d}}{2} \tag{77}$$

Cell-centered gradient: In what follows we also need to define $G^{\mathrm{cc}}[\psi]_{\mathbf{i}}$. For component d, the cell-centered gradient is

$$G_d^{\mathrm{cc}}[\psi]_{\mathbf{i}} = \frac{G_d^{\mathrm{mac}}[\psi]_{\mathbf{i}+(1/2)\mathbf{e}_d} + G_d^{\mathrm{mac}}[\psi]_{\mathbf{i}-(1/2)\mathbf{e}_d}}{2} \tag{78}$$

where the face-centered gradients for uncovered faces are computed by (69). For covered faces, we extrapolate to the covered face-center by using uncovered values, i.e. (69). For regular control-volumes this procedure reduces to a traditional centered difference.

Cell-centered projection: Here we are solving a similar problem as in the MAC case. To do this, we average cell-centered velocities to faces, apply the MAC projection, and average the gradient field back to cell-centers. Our cell-centered projection operator does the following:

$$\mathbf{u}_{\mathbf{i}} = \mathbf{P}_{\rho}^{\mathrm{cc}}(\mathbf{w}_{\mathbf{i}}^{*}) \tag{79}$$

Now we define our cell-centered projection

$$\mathbf{P}_{\rho}^{\mathrm{cc}} \equiv \mathbf{I} - \mathbf{Q}_{\rho}^{\mathrm{cc}} \tag{80}$$

$$\mathbf{Q}_{\rho}^{\mathrm{cc}} \equiv \frac{1}{\rho} G^{\mathrm{cc}} \left[D^{\mathrm{mac}} \frac{1}{\rho} G^{\mathrm{mac}} \right]^{-1} [D^{\mathrm{mac}} \cdot A^{\mathrm{C}\to\mathrm{F}}] \tag{81}$$

and where $A^{\mathrm{C}\to\mathrm{F}}$ is an averaging operator that moves data from cell-centers (C) to face-centers (F).

The cell-centered projection uses a MAC-projection to obtain the face-centered gradient, which we then use to compute G^{cc}. A numerical convergence study for the density-weighted MAC-projection is given in Section 5.2.4.

Filtering divergent velocity modes: Since our cell-centered approximate projection methods have a null space where divergent velocity fields can exist [63], we damp those modes with a divergence sensitive filter:

$$\mathbf{u}_{\mathbf{i}}^{n+1} := \mathbf{u}_{\mathbf{i}}^{n+1} + \lambda G D \mathbf{u}_{\mathbf{i}}^{n+1} \tag{82}$$

DOI: 10.1002/fld

We discretize the divergence D using a face-centered discretization, and the gradient G using a cell-centered discretization, with λ as a damping coefficient. GD is an approximation of the matrix-valued operator $\partial_{x_i}\partial_{x_j}$. This is algebraically equivalent to a cell-centered projection in which, instead of solving an elliptic equation, only one Jacobi iteration is taken toward the solution.

4. TIME DISCRETIZATION

In this section we describe the time discretizations necessary to solve the incompressible Navier–Stokes equations (1)–(4).

4.1. Semidiscrete version of projection formulation

Using the hyperbolic methodology of Section 3.5, the elliptic and parabolic methodology of Section 3.4, and the projection ideas from Section 2.3, we are now in a position to describe our solution strategy for the incompressible Navier–Stokes equations. The procedure is outlined as follows:

1. Compute the advective terms ($[\mathbf{u}\cdot\nabla\mathbf{u}]^{n+1/2}$ and $[\mathbf{u}\cdot\nabla s]^{n+1/2}$) as a combination of stable and conservative approximations (using an exactly divergence-free advective velocity).
2. Update passive scalars by solving a parabolic system (this reduces to an explicit update if there is no scalar diffusion).
3. Predict velocity field ($\mathbf{u}^*$) by solving a parabolic system.
4. Correct the predicted velocity by approximately projecting it onto a divergence-free space.

The detailed algorithm for advancing a single time step is presented sequentially in the following sub-sections.

4.1.1. Compute advective terms. Compute the cell-centered advective terms: $[\mathbf{u}\cdot\nabla\mathbf{u}]^{n+1/2}$, $[\mathbf{u}\cdot\nabla\rho]^{n+1/2}$, and $[\mathbf{u}\cdot\nabla s]^{n+1/2}$. See Section 3.5.

4.1.2. Update scalars. We discretize the scalar parabolic systems in time using the L_0-stable method of Twizell, Gumel, and Arigu (TGA) [14], which was also described in [12, 13]. We use TGA because the Crank–Nicolson method is unstable for EB and for locally refined meshes (see [12, 15] for a discussion). The scalar update is

$$s^{n+1} = (I-\mu_1 L)^{-1}(I-\mu_2 L)^{-1}[(I+\mu_3 L)s^n + (I+\mu_4 L)\, f_s^{n+1/2}\Delta t] \tag{83}$$

where

$$f_s^{n+1/2} \equiv -[\mathbf{u}\cdot\nabla s]^{n+1/2} + H_s^{n+1/2} \tag{84}$$

and where μ_1–μ_4 are defined in [13]. Note that for simplicity in (83), the Laplacian operators (L) include the diffusion constant (k_s).

4.1.3. Predict velocity. We predict the new velocity by solving parabolic systems for the cell-centered $\mathbf{u}^*$. The parabolic time-discretized momentum equations are

$$\mathbf{u}^{n+1} = (I-\mu_1 L)^{-1}(I-\mu_2 L)^{-1}[(I+\mu_3 L)\mathbf{u}^n + (I+\mu_4 L)\, f^{n+1/2}\Delta t] \tag{85}$$

DOI: 10.1002/fld

where

$$f^{n+1/2} \equiv -[\mathbf{u}\cdot\nabla\mathbf{u}]^{n+1/2} - \frac{\nabla p^{n+1/2}}{\rho^{n+1/2}} + \mathbf{g}^{n+1/2} \tag{86}$$

The discretizations of the advective terms are defined in Section 3.5. Since we do not know the new gradient of pressure and cannot ensure a divergence-free $\mathbf{u}^{n+1}$, we take a predictor step for $\mathbf{u}^*$ (using TGA),

$$\mathbf{u}^* = (I-\mu_1 L)^{-1}(I-\mu_2 L)^{-1}[(I+\mu_3 L)\mathbf{u}^n + (I+\mu_4 L) f^{n-1/2}\Delta t] \tag{87}$$

where

$$f^{n-1/2} \equiv -[\mathbf{u}\cdot\nabla\mathbf{u}]^{n+1/2} - \frac{\nabla p^{n-1/2}}{\rho^{n+1/2}} + \mathbf{g}^{n+1/2} \tag{88}$$

Note that for simplicity in (85) and (87) the Laplacian operators (L) include the diffusion constant (ν).

4.1.4. Correct predicted velocity. We correct $\mathbf{u}^*$ by approximately projecting it onto a divergence-free space. We do this by using $\mathbf{P}^{cc}_{\rho}$, a cell-centered discretization of the projection operator, and update the pressure gradient. Following [4, 11, 20, 64] and (85)–(88) we have

$$\mathbf{u}^* = \mathbf{u}^{n+1} + \Delta t \frac{\nabla p^{n+1/2} - \nabla p^{n-1/2}}{\rho^{n+1/2}} + O(\Delta t^3) \tag{89}$$

We present two different density-weighted approximate projection formulations. The first method we term DWPM1 and is given by

$$\mathbf{u}^{n+1} = \mathbf{P}^{cc}_{\rho^{n+1/2}}\left(\mathbf{u}^* + \Delta t\left(\frac{\nabla p^{n-1/2}}{\rho^{n+1/2}} - \mathbf{g}\right)\right) + \Delta t\mathbf{g} \tag{90}$$

$$\frac{\nabla p^{n+1/2}}{\rho^{n+1/2}} = \frac{1}{\Delta t}\mathbf{Q}^{cc}_{\rho^{n+1/2}}\left(\mathbf{u}^* + \Delta t\left(\frac{\nabla p^{n-1/2}}{\rho^{n+1/2}} - \mathbf{g}\right)\right) \tag{91}$$

The second method we term DWPM2 and is given by

$$\mathbf{u}^{n+1} = \mathbf{P}^{cc}_{\rho^{n+1/2}}(\mathbf{u}^*) \tag{92}$$

$$\frac{\nabla p^{n+1/2}}{\rho^{n+1/2}} = \frac{\nabla p^{n-1/2}}{\rho^{n+1/2}} + \frac{1}{\Delta t}\mathbf{Q}^{cc}_{\rho^{n+1/2}}(\mathbf{u}^*) \tag{93}$$

These approximate cell-centered projections differ from that in the first method, DWPM1, we solve an elliptic equation for the pressure ($p^{n+1/2}$), while in the second method, DWPM2, we solve an elliptic equation for the pressure change ($p^{n+1/2} - p^{n-1/2}$). With DWPM1 we have the benefit of an implicit solve for pressure at every time step (at $n+\frac{1}{2}$). The DWPM1 drawback is that it is not clear how to extend the pressure outflow boundary conditions with variable density. With DWPM2 we have desirable $p_t = 0$ outflow boundary conditions, but we have an incremental update for the pressure ($p^{n+1/2}$) at every time step. A secondary drawback to DWPM2 is that the pressure and pressure gradient are subject to accumulation of high-frequency error for flows

DOI: 10.1002/fld

where the pressure is nearly constant in time. This is due to the fact that there is no mechanism to damp these errors with an incremental update (DWPM1 damps the errors via the elliptic solve for $p^{n+1/2}$; see [64] for a relevant discussion). Nonetheless, DWPM2 produces good results as is seen in Section 5.5.1.

As part of the cell-centered projection, our last step is to apply a filter to damp divergent modes that are in the null space of P_ρ^{cc} (see Section 3.6.2).

4.2. AMR discretization issues

We extend our single-level solution methodology (as described in the previous sections) for solving the incompressible Navier–Stokes equations, by extending the single-level operators in a conservative and second-order-accurate manner. The adaptive pseudo-code is nearly identical to the single-grid algorithm with the exception of regridding and coarse–fine boundary conditions. Specifically, we match fluxes at coarse–fine interfaces (e.g. in the MAC-projection) and use conservative second-order-accurate coarsening and refinement when regridding. Regridding is done using second-order-accurate conservative linear interpolation from coarse regions to newly fine regions, and second-order-accurate conservative coarsening (i.e. classic averaging) from fine to newly coarse regions. With this accurate regridding step, we maintain second-order accuracy of the entire method. Refinement is restricted to a fixed number of levels as is specified before a simulation is run. Recall that for this paper we do not permit the coarse–fine interface to cross the EB, and we do not subcycle the AMR levels (subcycling maintains the same advective CFL condition on each level) as is done in [3, 15, 16].

A pseudo-code description of our composite EB AMR INS algorithm is presented in Figure 8. For the PiecewiseLinearFillPatch() function we do conservative linear interpolation based on coarse data to fine ghost cells, as in [3, 16]. For the ExtrapToFacesAtHalfTime() function we use our single-level infrastructure. Our discrete composite (comp) operator $\mathbf{P}_\rho^{\mathrm{mac,comp}}$ is the multilevel analog to the single-level $\mathbf{P}_\rho^{\mathrm{mac}}$ operator. Through a multi-level coupled solve, $\mathbf{P}_\rho^{\mathrm{mac,comp}}$ maintains conservation at coarse–fine interfaces by setting coarse fluxes equal to the average of the fine fluxes at coarse–fine interfaces (as in Figure 6). The Hyperbolic() function computes the advective terms by level. Our CompositeParabolic() function is as in [13], except we use coarse–fine interpolation and refluxing methodology to couple the levels. The $\mathbf{P}_\rho^{\mathrm{cc,comp}}$ is related to the $\mathbf{P}_\rho^{\mathrm{mac,comp}}$ in the same way as for the single-level solves (where we interpolate data back and forth between cell- and face-centers).

In our Regrid() step we loop through coarse AMR levels, tagging control volumes needing refinement, typically with a user-supplied threshold based on vorticity magnitude or density gradients (other tagging criteria are possible, such as geometric locations or passive scalars); cluster the tags into disjoint blocks using [65]; copy or conservatively linearly interpolate the old data to the new data structures; and project the regridded velocity, as in [3, 16]. When a tagging threshold is specified, the refinement algorithm refines regions that exceed the threshold (leaving other regions for possible coarsening). The refinement algorithm is sensitive to the tagging criteria in the sense that if one is not careful, excessive or no refinement is possible; it typically takes a few trial runs to set the threshold correctly. In future work, we aim to take a more mathematical approach to refinement by utilizing an error-based refinement criterion (see [16, Chapter 5]). Note that a maximum number of refinement levels is specified before the start of a simulation, so endless refinement does not occur. The regridding frequency is specified by a CFL-type constraint, where we do not want tagged fluid regions to escape to coarser control volumes.

DOI: 10.1002/fld

```
Initialize(u^0, ρ^0, ∇p^{-1/2})
for n = 0, ..., NumTimeSteps
{
    PiecewiseLinearFillPatch(ρ^n)
    PiecewiseLinearFillPatch(u^n)
    for l = 0, ..., l^max
        {
            ũ^{n+1/2,l} = ExtrapToFacesAtHalfTime(u^{n,l}, [∇p/ρ]^{n-1/2,l})
        }
    u^{ADV} = P_ρ^{mac,comp}(ũ^{n+1/2})
    for l = 0, ..., l^max
        {
            [[u·∇u]^{n+1/2,l}, [u·∇ρ]^{n+1/2,l}] = Hyperbolic(u^{n,l}, u^{ADV,l}, ρ^{n,l}, [∇p/ρ]^{n-1/2,l})
            ρ^{n+1,l} = ρ^{n,l} − Δt[u·∇ρ]^{n+1/2,l}
        }
    u* = CompositeParabolic(u^n, [u·∇u]^{n+1/2}, [∇p/ρ]^{n-1/2})
    [u^{n+1}, [∇p/ρ]^{n+1/2}] = P_ρ^{cc,comp}(u*, ρ^{n+1/2}, [∇p/ρ]^{n-1/2})
    Regrid()
}
```

Figure 8. Pseudo-code description of the adaptive incompressible Navier–Stokes algorithm. Notation for this figure is described in Section 4.2.

5. RESULTS

Here we demonstrate the accuracy of the method for studying variable-density incompressible flows with a series of test problems. The test problems build in difficulty, starting with the accuracy of the main operators followed by several fluid mechanics examples.

5.1. Accuracy measures

To test the convergence properties of the method, we conduct convergence studies. Where our test problems have unknown exact solutions, we conduct asymptotic convergence studies using at least three sets of simulations, coarse (c), medium (m) and fine (f), and compute errors ($e_{\mathbf{i}}$) between simulation solutions (ψ). In the absence of an exact solution, the asymptotic solution errors are computed as follows:

$$e^{\mathrm{c}} \equiv C^{\mathrm{m}\to\mathrm{c}}(\psi^{\mathrm{medium}}) - \psi^{\mathrm{coarse}} \tag{94}$$

$$e^{\mathrm{m}} \equiv C^{\mathrm{f}\to\mathrm{m}}(\psi^{\mathrm{fine}}) - \psi^{\mathrm{medium}} \tag{95}$$

where $C^{\mathrm{m}\to\mathrm{c}}$ and $C^{\mathrm{f}\to\mathrm{m}}$ are discrete coarsening operators (that coarsen discrete cell-centered values from fine grids to coarser grids) with accuracy greater than or equal to the expected solution error accuracy (in this paper we use second-order-accurate coarsening operators; i.e. traditional 'sum and divide' averaging). For further discussion of coarsening operators, we refer the reader to [48] where higher-order-accurate discrete coarsening operators are necessary. We then compute norms

DOI: 10.1002/fld

of these errors, and subsequently the asymptotic convergence rate of the method for the particular test problem and norm, as in [11, 26].

We compute volume-weighted p-norms as follows:

$$L_p \equiv \|e\|_p = \frac{1}{|\Omega|}\left(\sum_l \sum_{\mathbf{i}\in\Omega^l_{\text{valid}}} |e^l_{\mathbf{i}}|^p |V^l_{\mathbf{i}}|\right)^{1/p} \tag{96}$$

where $|\Omega|$ is the volume of the flow domain, $e^l_{\mathbf{i}}$ is the discrete error at control volume location $\mathbf{i}$ and level l, and $|V^l_{\mathbf{i}}|$ are the individual control volume magnitudes for level l, see (25). When comparing medium (e^{m}) and coarse (e^{c}) errors, we compute convergence rates as follows:

$$r_p = \frac{\log\left(\dfrac{\|e^{\text{c}}\|_p}{\|e^{\text{m}}\|_p}\right)}{\log\left(\dfrac{h^{\text{c}}}{h^{\text{m}}}\right)} \tag{97}$$

The expected asymptotic solution error convergence rates for the whole method are $r_1 = 2$, $r_2 = 1.5$, and $r_\infty = 1$, as in [26, 41].

5.2. *Convergence properties of the main operators*

In this section we present numerical converge studies for the main operators that we use to solve the governing equations: hyperbolic, elliptic, and parabolic. We also present convergence results for the density-weighted projections.

5.2.1. Hyperbolic equations. Our test problem here is scalar advection ($s_t + \mathbf{u}\cdot\nabla s = 0$) of a Gaussian bump down an inclined channel. The channel is inclined so as to test a non-trivial embedded boundary (i.e. the control volumes span a range of geometries). The channel is parallel to the vector $\mathbf{n} = [1.13, 1.14, 1.0]$ (where the third component is used only for the 3D case). In 3D the channel is a cylinder, in 2D it is a constant-width channel. The channel has a radius of 0.212, and passes through the origin of the unit square/cube domain. The constant, unit magnitude velocity field is parallel to the channel axis. The exact solution is $s(\mathbf{x}, t) = e^{-r^2}$, where $r(\mathbf{x}, t)$ is the distance from the (moving) Gaussian origin $\mathbf{x}_b(t)$ to a point $\mathbf{x}$ in the channel. The Gaussian origin is initialized as the domain origin and is translated down the channel with $\mathbf{u}$; i.e. $\mathbf{x}_b = t\mathbf{u}$.

A 2D isotropic convergence study for this problem is shown in Table I. A 2D AMR isotropic convergence study for this problem is shown in Table II. A 2D anisotropic convergence study for this problem is shown in Table III. A 3D anisotropic convergence study for this problem is shown in Table IV. For hyperbolic problems we expect the convergence rates for our method to be $r_1 = 2.0, r_2 = 1.5, r_\infty = 1.0$, where we indicate convergence rates r_p as in (97). The convergence tables in this section indicate that we are achieving the expected rates.

5.2.2. Elliptic equations. To illustrate our ability to solve anisotropic elliptic problems, here we present results from a simple test case, where the exact solution is $\phi = \sin(1.3x)\sin(2.2y)[\sin(3.1z)]$. Consider a sphere of radius $\frac{1}{4}$ centered in a unit domain (i.e. a cube with sides of length 1).

DOI: 10.1002/fld

Table I. Hyperbolic test problem: isotropic grid, solution error and convergence rates: $\Delta x_c = \frac{1}{256} = 2\Delta x_m = 4\Delta x_f$; a 2D calculation.

Variable	Medium–coarse error	Fine–medium error	Rate
L_1 norm of scalar error	2.2876e−05	5.6418e−06	2.0196e+00
L_2 norm of scalar error	6.6815e−05	1.9502e−05	1.7765e+00
l_∞ norm of scalar error	8.3294e−03	4.1744e−03	9.9664e−01

Table II. Hyperbolic test problem: isotropic grid; solution error and convergence rates: $\Delta x_c = \frac{1}{16} = 2\Delta x_m = 4\Delta x_f$; a 2D calculation with three AMR levels having nref = 4.

Variable	Medium–coarse error	Fine–medium error	Rate
L_1 norm of scalar error	1.5316e−04	3.5190e−05	2.1218e+00
L_2 norm of scalar error	5.7150e−04	1.3468e−04	2.0852e+00
L_∞ norm of scalar error	1.3367e−02	5.7748e−03	1.2109e+00

Table III. Hyperbolic test problem: anisotropic grid ($\Delta x = 4\Delta y$); solution error and convergence rates: $\Delta x_c = \frac{1}{128} = 2\Delta x_m = 4\Delta x_f$; a 2D calculation.

Variable	Medium–coarse error	Fine–medium error	Rate
L_1 norm of scalar error	1.1499e−04	3.0643e−05	1.9078e+00
L_2 norm of scalar error	5.7566e−04	2.0620e−04	1.4812e+00
L_∞ norm of scalar error	1.6868e−02	8.4677e−03	9.9426e−01

Table IV. Hyperbolic test problem: anisotropic grid ($\Delta x = \Delta y = 2\Delta z$), solution error and convergence rates: $\Delta x_c = \frac{1}{16} = 2\Delta x_m = 4\Delta x_f$; a 3D calculation.

Variable	Medium–coarse error	Fine-medium error	Rate
L_1 norm of scalar error	8.4448e−03	2.1650e−03	1.9637e+00
L_2 norm of scalar error	1.7622e−02	4.4385e−03	1.9893e+00
L_∞ norm of scalar error	5.4763e−02	3.2235e−02	7.6459e−01

Computed using two parallel processors.

Our 2D fine grid has 32×640 (i.e. 20:1 aspect ratio) level-zero cells (including covered cells) and our AMR refinement ratio is two. The coarse solution is on the same grid layout as the fine solution, but coarsened by a factor of two.

We solve an elliptic problem using our AMR multigrid method and present the expected multigrid convergence history in Figure 9. Notice that the error is reduced by a constant factor by each v-cycle, as is expected by multigrid theory [60]. The method maintains second-order accuracy (see Table V for 2D AMR results, and Table VI for 3D single-grid results).

DOI: 10.1002/fld

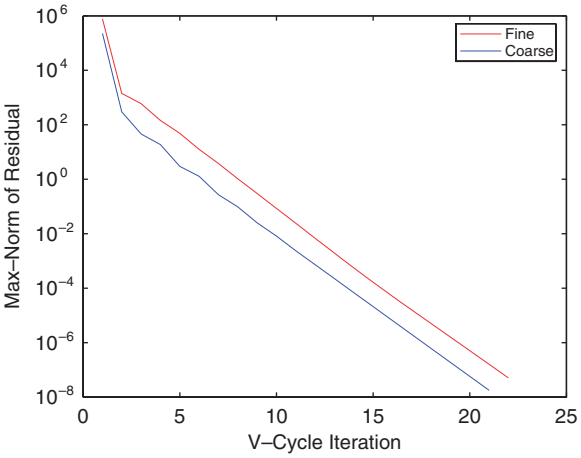

Figure 9. Multigrid convergence for a 20:1 aspect ratio 2D AMR solve.

Table V. Elliptic test problem: solution error convergence rates for a 20:1 aspect ratio AMR solve.

Variable	Coarse error	Fine error	Rate
L_1 norm error	9.717318e−05	2.456552e−05	1.983924e+00
L_2 norm error	1.909830e−04	4.851628e−05	1.976903e+00
L_∞ norm error	1.051167e−03	2.836212e−04	1.889955e+00

$\Delta x_\mathrm{f} = \frac{1}{32}$ and $\Delta x_\mathrm{c} = 2\Delta x_\mathrm{f}$, 2D.

Table VI. Elliptic test problem: solution error convergence rates for a 4:4:1 aspect ratio single-grid solve.

Variable	Coarse error	Fine error	Rate
L_1 norm error	6.459656e−05	1.403111e−05	2.202828e+00
L_2 norm error	9.121400e−05	2.166892e−05	2.073628e+00
L_∞ norm error	3.873316e−03	1.019814e−03	1.925263e+00

$\Delta x_\mathrm{f} = \frac{1}{64}$ and $\Delta x_\mathrm{c} = 2\Delta x_\mathrm{f}$, 3D.

For elliptic problems we expect the convergence rates for our method to be $r_1 = 2.0, r_2 = 2.0, r_\infty = 2.0$. The convergence tables in this section indicate that we are achieving the expected rates.

5.2.3. Parabolic equations. Our test problem here is scalar diffusion ($s_t = \nu \Delta s + H$) in a unit domain with a centered, embedded sphere of radius $\frac{1}{4}$. Our exact solution $s = \sin(5x)\sin(5y)\sin(5z)\cos(t)$ is imposed as an initial condition (at $t = 0$) and as a Dirichlet boundary condition. The source term H is computed from the exact solution given $\nu = 0.01$. A 2D anisotropic convergence study for this problem is shown in Table VII. A 3D anisotropic convergence study for this problem is shown in Table VIII.

DOI: 10.1002/fld

Table VII. Parabolic test problem: solution error convergence rates ($\Delta x = 2\Delta y$).

Variable	Coarse error	Fine error	Rate
L_1 norm of error	5.242795e−03	1.335954e−03	1.972465e+00
L_2 norm of error	7.953677e−03	2.038763e−03	1.963928e+00
L_∞ norm of error	3.628538e−02	1.074547e−02	1.755660e+00

$\Delta x_f = \frac{1}{16}$ and $\Delta x_c = 2\Delta x_f$; a 2D calculation with two AMR levels having nref=4.

Table VIII. Parabolic test problem: solution error convergence rates ($\Delta x = \Delta y = 2\Delta z$).

Variable	Coarse error	Fine error	Rate
L_1 norm of error	4.869321e−03	1.212166e−03	2.006133e+00
L_2 norm of error	7.659393e−03	1.922383e−03	1.994334e+00
L_∞ norm of error	4.814209e−02	1.298819e−02	1.890098e+00

$\Delta x_f = \frac{1}{16}$ and $\Delta x_c = 2\Delta x_f$; a 3D calculation with one AMR levels having nref=4.

For parabolic problems we expect the convergence rates for our method to be $r_1 = 2.0, r_2 = 2.0, r_\infty = 2.0$. The convergence tables in this section indicate that we are achieving the expected rates.

5.2.4. Density-weighted projections. Here we project a given velocity field onto a divergence-free space. Our given divergent velocity field is $\mathbf{u} = (\sin(\pi x), \sin(2\pi y), \sin(3\pi z))$. For the density-weighted projections, which use DWPM1 of Section 4.1.4, we prescribe a density field of $\rho = 2 + \sin(x) + \sin(y) + \sin(z)$. We have a unit domain with a centered embedded sphere of radius $\frac{1}{4}$, and our boundary conditions are $\mathbf{u} \cdot \mathbf{n} = 0$.

Density-weighted MAC projection: 2D and 3D anisotropic convergence studies for this problem were performed, and resulted in discretely exact, divergence-free velocity fields (as determined by the multigrid solver tolerance). For MAC-projections we expect exact projections, and this is what we are achieving.

Density-weighted cell-centered projection: A 2D anisotropic convergence study for this problem is shown in Table IX. A 3D anisotropic convergence study for this problem is shown in Table X. For approximate cell-centered projections we expect the (velocity divergence) convergence rates for our method to be: $r_1 = 2.0, r_2 = 1.5, r_\infty = 1.0$, where we indicate convergence rates r_p as in (97). The convergence tables in this section indicate that we are achieving the expected rates.

5.3. Sphere-driven cavity

To test the convergence properties of our incompressible Navier–Stokes algorithm (using DWPM1 from Section 4.1.4), we simulate the 3D evolution of the flow field within a rigidly rotating sphere, as shown in Figure 10. This is a good problem because both initial and boundary conditions are smooth–desirable characteristics for numerical convergence studies. This is also a good test problem because it allows us to test for symmetry in our solution. The sphere has a radius of 0.5 m and is centered at the origin. The rigid rotation of the sphere is around the z-axis and is

DOI: 10.1002/fld

Table IX. Cell-centered projection: divergence and convergence rates ($\Delta x = 2\Delta y$).

Variable	Coarse divergence	Fine divergence	Rate
L_1 norm divergence	3.527372e−03	9.168277e−04	1.943871e+00
L_2 norm divergence	6.020600e−03	2.022201e−03	1.573981e+00
L_∞ norm divergence	1.050206e−01	5.342729e−02	9.750238e−01

$\Delta x_\mathrm{f} = \frac{1}{128}$ and $\Delta x_\mathrm{c} = 2\Delta x_\mathrm{f}$; a 2D calculation with two AMR levels having nref = 2.

Table X. Cell-centered projection: divergence and convergence rates ($\Delta x = \Delta y = 2\Delta z$).

Variable	Coarse divergence	Fine divergence	Rate
L_1 norm divergence	1.519918e−01	4.303184e−02	1.820517e+00
L_2 norm divergence	2.056353e−01	6.222668e−02	1.724483e+00
L_∞ norm divergence	1.299411e+00	6.439040e−01	1.012940e+00

$\Delta x_\mathrm{f} = \frac{1}{32}$ and $\Delta x_\mathrm{c} = 2\Delta x_\mathrm{f}$; a 3D calculation with two AMR levels having nref = 2.

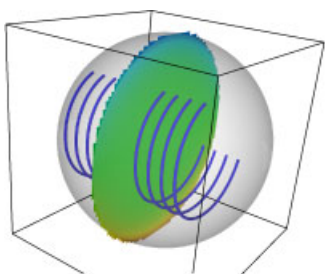

Figure 10. Sphere-driven cavity. The slice is colored by u-velocity, and instantaneous streamtubes aid in visualizing the flow.

started smoothly with a cubic polynomial as follows:

$$\mathbf{u}^{\mathrm{Sphere}} = [-y, x, 0] f(t) \tag{98}$$

$$f(t) = \begin{cases} r\left(-2\left(\dfrac{t}{T}\right)^3 + 3\left(\dfrac{t}{T}\right)^2\right) & \text{if } t < T \\ r & \text{if } t \geqslant T \end{cases} \tag{99}$$

We set $r = 0.1$ and $T = 100\,\mathrm{s}$, and our kinematic viscosity is $v = 1\,\mathrm{m}^2\,\mathrm{s}^{-1}$, giving an approximately unit Reynolds number for $t/T = 0.06$, the stopping time for this test. We impose a no-slip boundary condition on the sphere. This no-slip boundary condition, combined with the rigid rotation of the sphere, slowly drives the interior fluid into motion, as in Figure 10. The long-term state for this flow is rigid rotation of the entire fluid, with constant vorticity parallel to the axis of rotation.

DOI: 10.1002/fld

Table XI. Solution error convergence rates: $\Delta x_c = \frac{1}{64} = 2\Delta x_m = 4\Delta x_f$.

Norm	Variable	Medium–coarse error	Fine–medium error	Rate
L_1	U-velocity	2.7969e−06	7.9661e−07	1.8119e+00
L_1	V-velocity	2.7969e−06	7.9661e−07	1.8119e+00
L_1	W-velocity	4.1303e−08	8.7800e−09	2.2340e+00
L_1	Pressure	1.2735e−05	2.8007e−06	2.1850e+00
L_1	Scalar	5.6574e−05	1.4727e−05	1.9417e+00
L_2	U-velocity	4.6086e−06	1.3910e−06	1.7282e+00
L_2	V-velocity	4.6086e−06	1.3910e−06	1.7282e+00
L_2	W-velocity	9.4183e−08	2.3178e−08	2.0227e+00
L_2	Pressure	2.1915e−05	4.6073e−06	2.2499e+00
L_2	Scalar	3.0519e−04	1.0762e−04	1.5038e+00
L_∞	U-velocity	5.2068e−05	2.5166e−05	1.0489e+00
L_∞	V-velocity	5.2068e−05	2.5166e−05	1.0489e+00
L_∞	W-velocity	3.0969e−06	7.5950e−07	2.0277e+00
L_∞	Pressure	2.3083e−04	6.3841e−05	1.8543e+00
L_∞	Scalar	4.0330e−03	2.0201e−03	9.9740e−01

A 3D viscous calculation.

In order to test both the spatial and temporal accuracy of the method, we focus our attention on early times, $t/T = [0, 0.06]$. We computed 4, 8, and 16 time steps for the coarse, medium, and fine runs over this time frame, using 64 parallel processors. We use an isotropic mesh spacing (i.e. $\Delta x = \Delta y = \Delta z$).

Convergence results from this calculation are presented in Table XI. The results are as we would expect ($r_1 = 2$, $r_2 = 1.5$, and $r_\infty = 1$) from such a smooth problem. The results are symmetric as is apparent by comparing the U-velocity and V-velocity errors.

5.4. Lab-scale density-driven flows

To demonstrate the ability of the method to simulate highly nonlinear, variable-density flows, we present a classic density-driven exchange flow. This flow is numerically highly demanding due to the sharp gradients and multiscale nature. With AMR we are able to accurately resolve the important features.

We simulate flow inside a 0.5 m tall, by 3 m wide tank. On the left side of the tank we start with light water, on the right is heavy water. The density ratio of light to heavy fluid is $\frac{1000}{1030}$, and our kinematic viscosity is $v = 10^{-6}\,\mathrm{m^2\,s^{-1}}$. A resulting snapshot of the flow as it evolves using four AMR levels is (zoomed in and) shown in Figure 11(a), with the adaptive control volumes in Figure 11(b). Notice that we are resolving the boundary layer at the top and bottom of the tank, and are generating Kelvin-Helmholtz-type instability billows along the sheared density interface; all while the finest control volumes track the areas with strongest vorticity. A 3D version of this simulation, using three AMR levels, is shown in Figure 11(c). Notice the boundary layers developing on the walls; the lobe and cleft structure forming at the fronts; and the 3D turbulent structure along the shear layer—these are all expected for this well-understood flow (see [66] for a more detailed study of these types of flows).

To asses the performance of the EB AMR implementation, we compute a series of runs with the same initial conditions as above, but now the problem is 8 m long by 0.5 m deep with a centered

DOI: 10.1002/fld

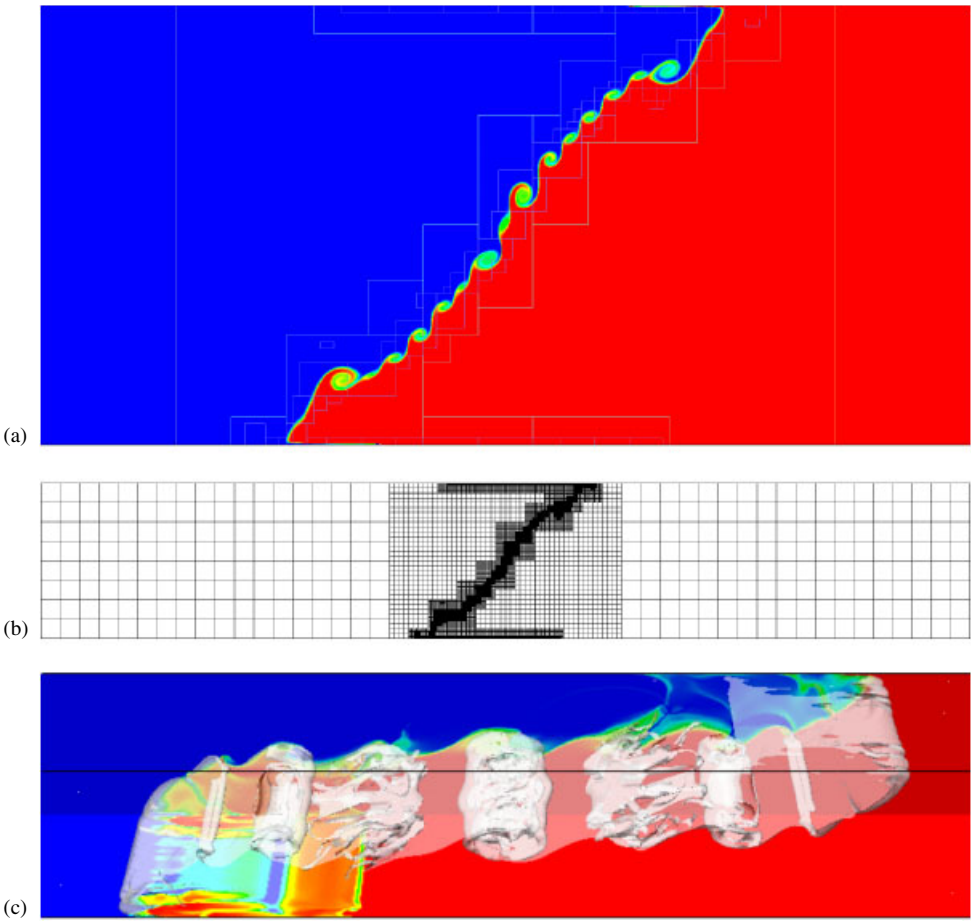

Figure 11. (a) and (b) are from a 2D lock-exchange calculation and have four AMR levels. (a) is of density and (b) shows all of the control volumes. (c) is a 3D lock-exchange calculation with three AMR levels; density is both isosurfaced and colored on the bottom and back walls.

Gaussian sill. The Gaussian is described by $H\exp^{-x^2/2\sigma^2}$, with $\sigma^2 = 0.0625\,\mathrm{m}^2$ and the sill height $H = 0.1\,\mathrm{m}$. Again, the density ratio is $\frac{1000}{1030}$ and our kinematic viscosity is $\nu = 10^{-6}\,\mathrm{m}^2\,\mathrm{s}^{-1}$. We vary the coarsest domain and number of AMR levels, and monitor the time spent to compute 10 time steps.

Table XII presents the performance of the method as computed on a serial workstation. Notice that we obtain a factor of roughly 9 speedup in execution time for the finest runs. Also notice that as the finest domain becomes finer our speedup improves. Memory savings are directly related to the grid saturation, where we define grid saturation to be the ratio of the number of control volumes used in an AMR calculation to the number of control volumes in an equivalent resolution single-level calculation. Again, as the finest domain gets finer, the grid saturation gets smaller, indicating significant memory savings. The combination of algorithmic speedup and memory savings due to low grid saturation enables simulations that are not otherwise possible.

DOI: 10.1002/fld

Table XII. Performance of the method for 10 time steps of the 2D lock-exchange on a sill problem.

Number of levels	Finest domain	Grid saturation	Speedup
1	2048×128	1	1
2	2048×128	0.120	2.70
3	2048×128	0.088	3.01
1	4096×256	1	1
2	4096×256	0.107	4.06
3	4096×256	0.0435	5.35
1	8192×512	1	1
2	8192×512	0.102	5.19
3	8192×512	0.0325	9.20

Finest domain is the number of control volumes for an equivalent single-level calculation; grid saturation is the number of control volumes in the AMR hierarchy divided by the finest domain; speedup is the amount of time taken to solve 10 time steps, normalized by the equivalent single-level calculation. The refinement ratio is 4 between levels.

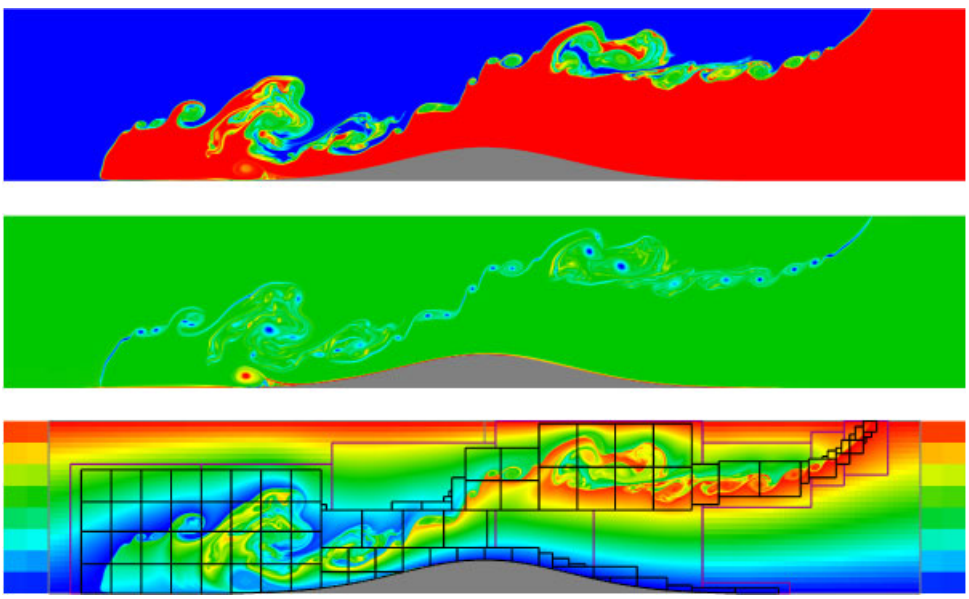

Figure 12. A 2D lock-exchange over a Gaussian sill. The top plot is colored by density, the middle plot is vorticity, and the bottom plot is a passive scalar initialized to depth.

A snapshot of a well-resolved (four AMR levels) 2D run is presented in Figure 12. It is clear that the calculation is able to resolve boundary layers, shear instabilities on the interface, and the subsequent mixing that ensues. The 3D effects become significant as the simulation progresses, as shown in Figure 13.

5.5. Internal-wave generation and dissipation

Oceanic surface tides can induce currents over topography, which in turn generate internal gravity waves that propagate along density gradients beneath the surface. While the amplitude of oceanic

DOI: 10.1002/fld

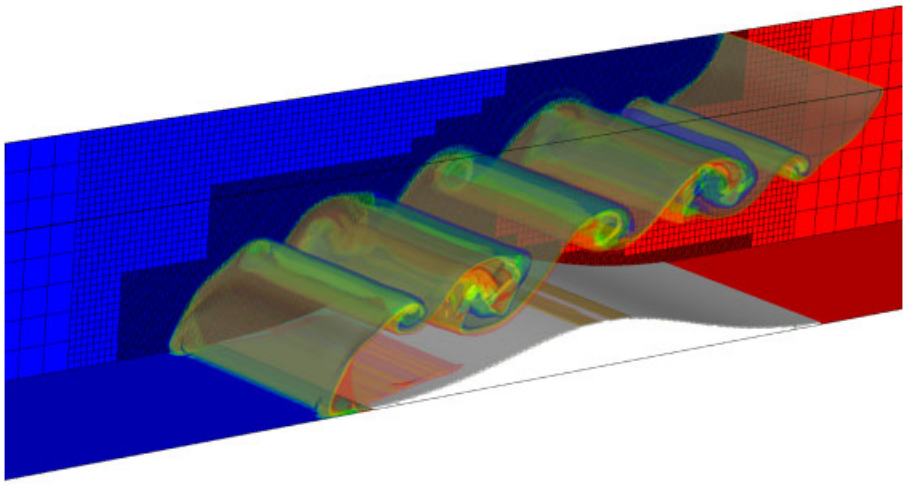

Figure 13. A 3D lock-exchange over a Gaussian sill: density is both isosurfaced and colored on the bottom and back walls.

surface waves due to the tides is typically less than meters, resulting internal-wave amplitudes can be larger than 100 m, and their associated currents can alter surface characteristics enough to make them visible from space. Their ubiquitous nature is documented in the Global Ocean Associates Internal Wave Atlas [67], which is a compilation of satellite images of internal waves. Like surface waves, internal waves can propagate until they dissipate at bathymetric boundaries, and in the oceanic context there is strong evidence in support of the idea that internal-wave breaking at boundaries results in mixed fluid which propagates out into the ocean interior [68–70].

In this section we test the ability of the method to generate and dissipate internal waves. This is not meant to be an exhaustive internal-wave study, but rather to demonstrate that the method is capable of capturing the important physical processes, where the goal is to test the accuracy of the method in simulating highly nonlinear, non-hydrostatic, density-driven flows.

5.5.1. Stratified flow past a sill. Using the 2D version of DWPM2 from Section 4.1.4, we simulate a field-scale, idealized, internal-wave-generating sill. This type of problem has been studied by Cummins *et al.* [71], Farmer and Armi [72] and Lamb [73] and is common in oceanographic (and even atmospheric) settings where currents force a stratified profile past topographic features.

The 2D domain is 256 m deep (D) by 4096 m long (L), with a Gaussian sill centered 1024 m from the left side. The Gaussian is described by $H\exp^{-x^2/2\sigma^2}$, with $\sigma^2 = 10^5\,\mathrm{m}^2$ and the sill height $H = 196\,\mathrm{m}$. This makes the shallowest spot above the sill 60 m deep. The domain is forced by a constant inflowing current (U) from the left face at $0.2\,\mathrm{m\,s}^{-1}$ (which is in the range of tidal currents). The stably stratified initial and inflowing density profiles are given by $\rho = 1001 - \exp^{0.0673z}\,\mathrm{kg\,m}^{-3}$, where $z = 0$ is at the top of the domain. The kinematic viscosity is $\nu = 10^{-6}\,\mathrm{m}^2\,\mathrm{s}^{-1}$. Significant features of the developed flow are shown in Figure 14.

An internal hydraulic jump forms at the downstream side of the sill, as can be seen in the density plot. The bifurcation point (at the hydraulic jump), where the strong downslope flow detaches from the surface, remains relatively stationary throughout the calculation. The downslope currents entrain lighter waters from the 'stagnant' pool just downstream of the bifurcation point, with

DOI: 10.1002/fld

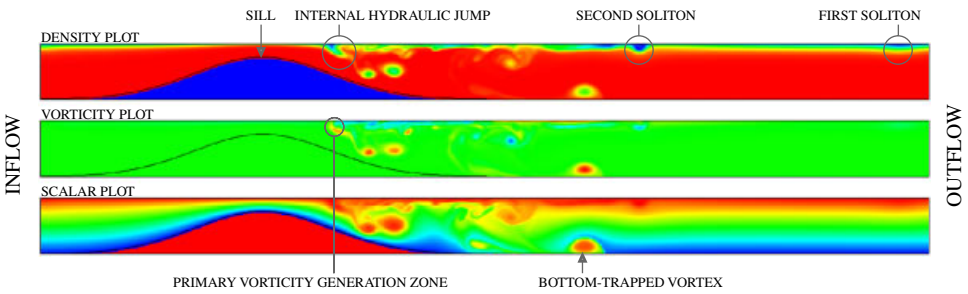

Figure 14. Stratified flow past a sill: significant features at $t/T = 0.232$.

entrained lighter waters advected downstream at the cores of vortices. The vortices are generated at the shear layer downstream of the bifurcation point. Note that the method robustly handles sharp gradients in the flow due to the slope limiters of Section 3.5.4 as part of the higher-order upwind Godunov hyperbolic methodology. At these sharp gradients, the method locally reduces to first-order accuracy (i.e. in L_∞), as is true with traditional shock-capturing schemes.

Three coherent structures are ejected from the hydraulic jump region. First, a long-wavelength soliton propagates as a wave of depression along the pycnocline (region of rapidly changing density) and exits the domain at $t/T \approx 0.24$, where we define the advective timescale $T \equiv L/U = 20480\,\mathrm{s}$ (labeled 'first soliton' in Figure 14). This long wave is also clearly evident in the density plot. Next, a shorter wavelength, solitary wave of depression is ejected and propagates downstream along the pycnocline (labeled 'second soliton' in Figure 14). Finally, a large vortex patch is shed from the shear layer at the internal hydraulic jump and propagates downstream along the bed (labeled 'bottom-trapped vortex' in Figure 14). This vortex core has lighter waters in its core, but is temporarily trapped along the bottom due to pressure gradients that are stronger than the buoyancy forcing.

Eventually, we expect the system to reach a dynamic equilibrium state, with continued vortex shedding, entrainment of lighter waters and internal-wave generation. Note that the initial conditions are idealized and the flow is likely highly 3D (especially downstream of the sill) in realistic oceanic settings. Nonetheless, we expect similar features to develop for similar flows past sills.

A convergence study (using 16 processors) for $t/T \leqslant 0.012$ of this problem is presented in Table XIII. We use an isotropic mesh spacing for these runs. The convergence study shows that we are getting the expected accuracy for this complex field-scale variable–density flow.

5.5.2. Internal-wave breaking on a uniform slope. Here we test our ability to simulate internal-waves breaking on a slope using DWPM1. For this test case we follow the lab-scale experiments of [74] and simulate lab-scale breaking of a solitary internal wave on a slope. The experimental tank is 3 m long by 0.5 m wide and tall, with the 8:1 (vertical:horizontal) slope meeting the vertical wall of the tank halfway up. We set free-slip boundary conditions on all walls except on the slope where we impose a no-slip condition. We initialize the density field as in the top panel of Figure 15, with salt water, $\rho = 1030\,\mathrm{kg\,m^{-3}}$, on the bottom and fresh water, $\rho = 1000\,\mathrm{kg\,m^{-3}}$, on the top, and a perturbed interface following $\phi = -0.25 - 0.15\mathrm{e}^{-3x^2}$. We smooth the interface over 0.1 m using a Heaviside smoothing kernel, Equation (60) in [23]. The kinematic viscosity is $\nu = 10^{-6}\,\mathrm{m^2\,s^{-1}}$.

We present our well-resolved 2D AMR calculations in the time sequence of Figure 15. We present plots from our 3D single-level calculations in Figure 16. In these figures, blue indicates

DOI: 10.1002/fld

Table XIII. Solution error convergence rates: $\Delta x_c = \frac{1}{256} = 2\Delta x_m = 4\Delta x_f$; a 2D calculation.

Norm	Variable	Medium–coarse error	Fine–medium error	Rate
L_1	U-velocity	6.3982e−05	1.6330e−05	1.9701e+00
L_1	V-velocity	4.7190e−05	1.3562e−05	1.7989e+00
L_1	Pressure	3.3547e−04	8.8763e−05	1.9182e+00
L_1	Density	2.6976e−03	6.6357e−04	2.0233e+00
L_1	Scalar	4.2447e−02	1.2221e−02	1.7963e+00
L_2	U-velocity	2.4874e−04	8.1063e−05	1.6175e+00
L_2	V-velocity	3.3952e−04	1.2351e−04	1.4589e+00
L_2	Pressure	5.2176e−04	2.1079e−04	1.3076e+00
L_2	Density	7.2390e−03	1.8842e−03	1.9419e+00
L_2	Scalar	2.0823e−01	7.5751e−02	1.4589e+00
L_∞	U-velocity	1.1622e−02	5.9232e−03	9.7246e−01
L_∞	V-velocity	1.0837e−02	5.7026e−03	9.2630e−01
L_∞	Pressure	4.0891e−03	2.5460e−03	6.8354e−01
L_∞	Density	5.7153e−02	1.9954e−02	1.5182e+00
L_∞	Scalar	5.9958e+00	2.9695e+00	1.0137e+00

fresh water and red is for salt water. In the 2D figures we included the AMR disjoint-block outlines. In the 3D figures we included an isosurface of the density interface and streamtubes to aid in visualizing the simulation.

As the stratification perturbation rebounds, it propagates to the right as a wave of depression. The depression wave begins to shoal on the slope as is evident in the second frame in Figure 15. Subsequently, the wave continues to propagate to the right but the interaction with the slope becomes stronger and the wave transforms into a wave of elevation. The wave then breaks on the slope with a large mass of dense fluid propagating up slope in a large vortex core. As the wave continues up slope, the breaking becomes more intense and results in stronger and stronger mixing. Coherent vortex structures are still evident in the final two frames of Figure 15 as the breaking wave is dissipated.

3D features are illuminated in Figure 16. In this sequence we can see that the wave maintains a 2D form until the fifth frame when side wall effects (likely triggered by side wall bounded Görtler like instabilities [75], but a detailed analysis is beyond the scope of this paper) propagate into the interior. Subsequently, the vortex tube bends into a horseshoe-like shape in the sixth frame with complete turbulent breaking and mixing quickly following.

The AMR boxes in Figure 15 are tracking density gradients and areas of strong vorticity. With this method we can recursively nest finer grids, resolving important physical processes. The ability to resolve internal-wave lifecyles with an adaptive method that tracks the important flow physics is new, and enables simulations that are not otherwise possible.

6. CONCLUSION

6.1. Summary and conclusions

This paper presents an adaptive, Cartesian-grid projection method that is suitable for the accurate numerical study of incompressible environmental flows in complex domains.

DOI: 10.1002/fld

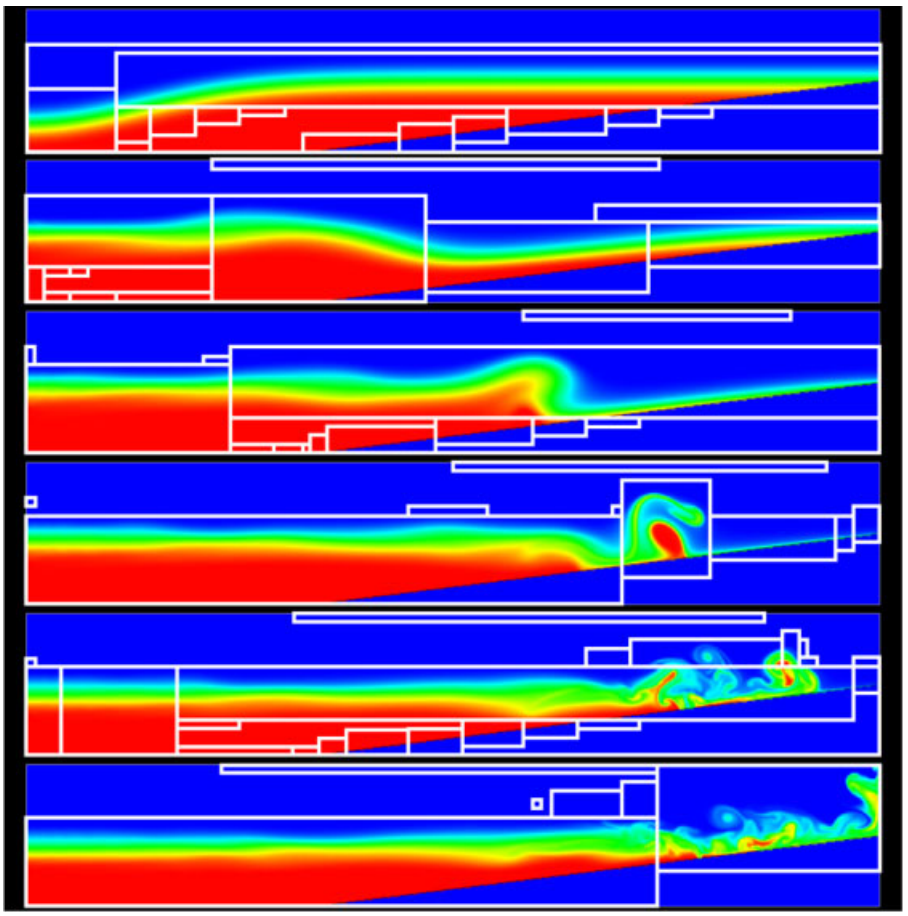

Figure 15. Internal-wave 2D breaking density plots. Blue is freshwater, red is saltwater. Boxes indicate refined regions.

We presented a spatial discretization, including the geometric description and methodologies for solving elliptic, parabolic and hyperbolic PDEs (the component PDEs that make up the incompressible Navier–Stokes equations). The finite-volume cut-cell discretization is relatively new, and is efficient and robust for all problems we have tried, even where the mesh spacing is anisotropic. This is because the geometry is generated in $O(N^{(D-1)/D})$ calculations from easily defined distance functions, is 'water-tight' and therefore allows exactly conservative finite-volume discretizations, handles arbitrarily complex geometries, and is easily coarsened/refined for use in multigrid and AMR.

The adaptive algorithm for solving the incompressible Navier–Stokes equations in complex geometry with buoyancy forcing extends the work of [3, 4, 7–9, 11, 16, 17] while maintaining second-order accuracy. This method enables the efficient and accurate simulation of multiscale, environmental fluid mechanics in realistic complex geometry. For example, the greatest impediment to the numerical simulation of oceanic internal waves is the broad range of length scales over which they exist. AMR makes the simulation of multiscale, highly nonlinear internal waves a tractable problem.

DOI: 10.1002/fld

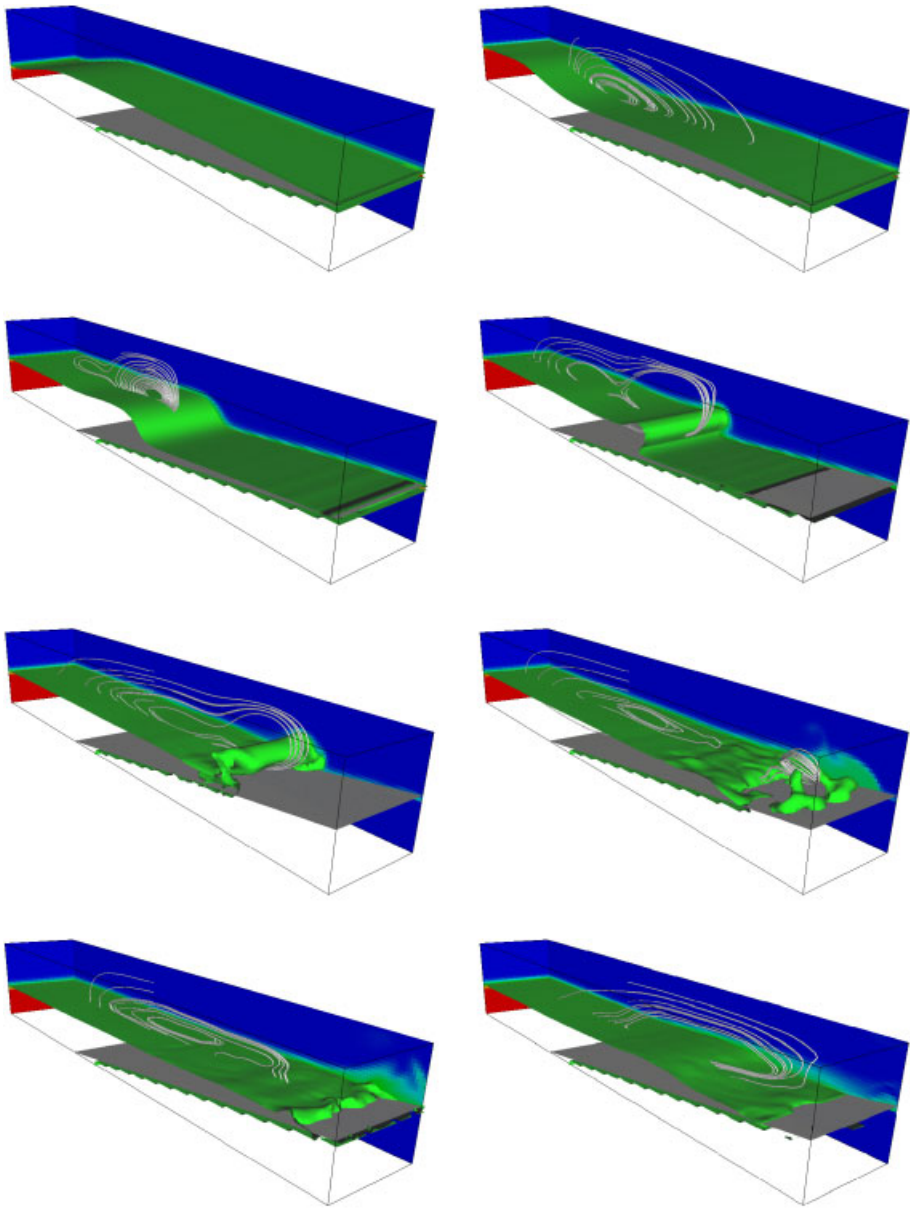

Figure 16. Internal-wave breaking 3D: initial conditions, shoaling, breaking, mixing, and dissipating. Blue is freshwater, red is saltwater. Iso-contour of salt–fresh interface, streamtubes for visualization.

We presented results from a range of test cases to validate that our method is in fact second-order accurate. We first established that the hyperbolic, elliptic, parabolic, and projection operators performed to the expected accuracy. Subsequently, the lab-scale sphere-driven cavity test showed that our method is performing as expected. To test the density-weighted projection methods we simulated lab-scale lock-exchange flows, field-scale stratified flow past a sill, and breaking internal

DOI: 10.1002/fld

waves on a lab-scale slope, with and without AMR. For the flow past a sill test we validated that our method is second-order accurate, and produced some interesting results (an internal hydraulic jump, vortex shedding, bottom-trapped vortices, and the generation of internal solitary waves of depression). Note that in the work of [26] we presented further validation of the method, and the reader should be aware of this effort.

6.2. Future algorithm work

Our first task for future work is to further optimize the code. Serial and parallel performance optimizations will benefit the elliptic solver, the slowest part of the algorithm. Optimizations of our current solver may include reducing the parallel communication costs by (a) utilizing additional ghost cells, and (b) improving our load-balancing algorithm to account for the complex geometry overhead. Recent performance results have shown that AMR elliptic solvers, in the absence of complex geometry, currently scale to $O(10^4)$ parallel processors [76]. We anticipate that the method presented in this work will have similar parallel scaling characteristics.

Extension of this work to the case where coarse–fine interfaces are allowed to cross the EB interface would be useful. This would remove the constraint that all irregular control volumes have maximum refinement (the hyperbolic case has already been successfully addressed in [41]). The elliptic and parabolic cases are relatively straightforward and at the interface crossing requires modification of (1) the Dirichlet EB condition, (2) the coarse–fine interpolation step, and (3) conservative coarse–fine refluxing. In addition, the crossing algorithm needs high-order coarsening, as in [48]. An initial implementation of the EB-AMR crossing algorithm for Poisson's equation is promising.

An additional useful extension is the ability to subcycle the levels to maintain consistent CFL numbers across all control volumes. This extension would follow the works of [3, 15, 16].

We anticipate extending the method to higher-order accuracy by following the work of [48]. The quadrature formulas in [48] provide a systematic mechanism for distinguishing between averages over cells, averages over faces, and point values, to fourth-order accuracy. This can be combined with the ideas here and in [77, 78] to obtain fourth-order in space finite-volume discretizations for mixed hyperbolic/parabolic/elliptic problems on a locally refined grid. It is not obvious how to extend the Mehrstellen discretizations in [48] to the case where the right-hand side includes a time derivative, particularly in the case where implicit differencing in time is required. We will consider a variety of possible approaches here, including fully implicit methods and predictor–corrector approximations to such methods in which the Mehrstellen correction is treated explicitly. It is also necessary to compute higher moments of the cut-cell intersections. The fourth-order spatial approach is straightforward to pursue in conjunction with second-order-accurate temporal discretizations. However, the extension to higher order in time is still an active research issue [79].

ACKNOWLEDGEMENTS

MFB acknowledges the support of the National Science Foundation MSPRF program and the Department of Energy CSGF program. PC acknowledges support by the U.S. Department of Energy Office of Advanced Scientific Computing under contract No. DE-AC02-05CH11231. SGS acknowledges the U.S. Environmental Protection Agency's Coastal Intensive Sites Network Program (Grant no. R826940-01-0). We thank Caroline Gatti-Bono, Dan Graves, Terry Ligocki, Dan Martin, Peter Schwartz, David Serafini, Chip Smith, Ted Sternberg, David Trebotich, and Brian Van Straalen; without their efforts this work would not have been possible. Thanks also to Professor Oliver Fringer. This research used resources of

DOI: 10.1002/fld

the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract no. DE-AC02-05CH11231. This research also used resources at the Stanford Center for Computational Earth & Environmental Science.

REFERENCES

1. Fringer OB, Gerritsen M, Street RL. An unstructured-grid, finite-volume, nonhydrostatic, parallel coastal ocean simulator. *Ocean Modelling* 2006; **14**:139–173.
2. Smith PE. A three-dimensional, finite-difference model for estuarine circulation. *Ph.D. Thesis*, University of California, Davis, 1997.
3. Almgren AS, Bell JB, Colella P, Howell LH, Welcome ML. A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations. *Journal of Computational Physics* 1998; **142**:1–46.
4. Bell JB, Marcus DL. A second-order projection method for variable-density flows. *Journal of Computational Physics* 1992; **101**:334–348.
5. Chorin AJ. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics* 1997; **135**(2):118–125.
6. Chorin AJ. Numerical study for slightly viscous flow. *Journal of Fluid Mechanics* 1973; **57**:785–796.
7. Chorin AJ. Numerical solution of incompressible flow problems. *Studies in Numerical Analysis* 1968; **2**:64–71.
8. Chorin AJ. Numerical solutions of the Navier–Stokes equations. *Mathematics of Computation* 1968; **22**:745–762.
9. Chorin AJ. On the convergence of discrete approximations to the Navier–Stokes equations. *Mathematics of Computation* 1969; **23**:341–353.
10. Chorin AJ, Marsden JE. *A Mathematical Introduction to Fluid Mechanics* (3rd edn). Springer: New York, 1993.
11. Bell JB, Colella P, Glaz HM. A second-order projection method for the incompressible Navier–Stokes equations. *Journal of Computational Physics* 1989; **85**:257–283.
12. McCorquodale P, Colella P, Johansen H. A Cartesian grid embedded boundary method for the heat equation on irregular domains. *Journal of Computational Physics* 2001; **173**:620–635.
13. Schwartz PO, Barad MF, Colella P, Ligocki TJ. A Cartesian grid embedded boundary method for the heat equation and Poisson's equation in three dimensions. *Journal of Computational Physics* 2006; **211**(2):531–550.
14. Twizell EH, Gumel AB, Arigu MA. Second-order, l_0-stable methods for the heat equation with time-dependent boundary conditions. *Advances in Computational Mathematics* 1996; **6**:333–352.
15. Martin DF, Colella P, Graves D. A cell-centered adaptive projection method for the incompressible Navier–Stokes equations in three dimensions. *Journal of Computational Physics* 2008; **227**:1863–1886.
16. Martin DF. An adaptive cell-centered projection method for the incompressible Euler equations. *Ph.D. Thesis*, Department of Mechanical Engineering, University of California, Berkeley, 1998.
17. VanStraalen B, Trebotich D, Schwartz PO, Ligocki TJ, Graves DT, Colella P, Barad MF. A Cartesian grid embedded boundary method for the incompressible Navier–Stokes equations. In preparation.
18. Gatti-Bono C, Colella P. An anelastic all speed projection method for gravitationally stratified flows. *Journal of Computational Physics* 2006; **216**(2):589–615.
19. Graves DT. An approximate projection method suitable for the modeling of rapidly rotating flows. *Ph.D. Thesis*, Department of Mechanical Engineering, University of California, Berkeley, December 1996.
20. Lai MF. A projection method for reacting flow in the zero mach number limit. *Ph.D. Thesis*, University of California, Berkeley, 1994.
21. Minion M. Two methods for the study of vortex patch evolution on locally refined grids. *Ph.D. Thesis*, University of California, Berkeley, May 1994.
22. Puckett EG, Almgren AS, Bell JB, Marcus DL, Rider WJ. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. *Journal of Computational Physics* 1997; **130**:269–282.
23. Sussman M, Puckett EG. A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows. *Journal of Computational Physics* 2000; **162**(2):301–337.
24. Trebotich DP. A projection method for incompressible viscous flow on a deformable domain. *Ph.D. Thesis*, Department of Mechanical Engineering, University of California, Berkeley, December 1998.
25. Brown DL, Cortez R, Minion ML. Accurate projection methods for the incompressible Navier–Stokes equations. *Journal of Computational Physics* 2001; **168**(2):464–499.
26. Barad MF. An adaptive Cartesian grid projection method for environmental flows. *Ph.D. Thesis*, University of California, Davis, 2006.

DOI: 10.1002/fld

27. Cummins SJ, Rudman M. An sph projection method. *Journal of Computational Physics* 1999; **152**(2):584–607.
28. Hirt CW, Cook JL, Butler TD. A lagrangian method for calculating the dynamics of an incompressible fluid with free surface. *Journal of Computational Physics* 1970; **5**(1):103–124.
29. Longuet-Higgins MS, Cokelet ED. The deformation of steep surface waves on water. i. A numerical method of computation. *Proceedings of the Royal Society of London*, *Series A*, *Mathematical and Physical Sciences* 1976; **350**(1660):1–26.
30. Monaghan JJ. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics* 1992; **30**: 543–574.
31. Brooks AN, Hughes TJR. Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations. *Computational Methods in Applied Mechanical Engineering* 1990; 199–259.
32. Diamessis PJ, Domaradzki JA, Hesthaven JS. A spectral multidomain penalty method model for the simulation of high Reynolds number localized incompressible stratified turbulence. *Journal of Computational Physics* 2005; **202**:298–322.
33. Fischer PF. An overlapping Schwarz method for spectral element solution of the incompressible Navier–Stokes equations. *Journal of Computational Physics* 1997; **133**:84–101.
34. Giraldo FX, Rosmond TE. A scalable spectral element Eulerian atmospheric model (SEE-AM) for NWP: dynamical core tests. *Monthly Weather Review* 2004; **132**:133–153.
35. Gresho PM, Chan ST, Lee RL, Upson CD. A modified finite element method for solving the time-dependent, incompressible Navier–Stokes equations. I Theory. *International Journal for Numerical Methods in Fluids* 1984; **4**:557–598.
36. Özgökmen TM, Fischer PF, Duan J, Iliescu T. Three-dimensional turbulent bottom density currents from a high-order nonhydrostatic spectral element model. *Journal of Physical Oceanography* 2004; **34**:2006–2026.
37. Shahbazi K, Fischer PF, Ethier CR. A high-order discontinuous Galerkin method for the unsteady incompressible Navier–Stokes equations. *Journal of Computational Physics* 2007; **222**:391–407.
38. Gibou F, Fedkiw RP, Cheng L-T, Kang M. A second-order-accurate symmetric discretization of the Poisson equation on irregular domains. *Journal of Computational Physics* 2002; **176**(1):205–227.
39. Griffith BE, Peskin CS. On the order of accuracy of the immersed boundary method: higher order convergence rates for sufficiently smooth problems. *Journal of Computational Physics* 2005; **208**(1):75–105.
40. Peskin CS. The immersed boundary method. *Acta Numerica* 2002; **11**:479–517.
41. Colella P, Graves DT, Keen BJ, Modiano D. A Cartesian grid embedded boundary method for hyperbolic conservation laws. *Journal of Computational Physics* 2006; **211**:347–366.
42. Johansen H, Colella P. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *Journal of Computational Physics* 1998; **147**(1):60–85.
43. Bell JB, Day MS, Rendleman CA, Woosley SE, Zingale MA. Adaptive low Mach number simulations of nuclear flames. *Journal of Computational Physics* 2004; **195**:677–694.
44. Crockett RK, Colella P, Fisher R, Klein RI, McKee CF. An unsplit, cell-centered Godunov method for ideal mhd. *Journal of Computational Physics* 2005; **203**:422–448.
45. Bell JB, Day MS, Shepherd ID, Johnson MR, Cheng RK, Grcar JF, Beckner VE, Lijewski MJ. From the cover: numerical simulation of a laboratory-scale turbulent V-flame. *Proceedings of the National Academy of Sciences* 2005; **102**(29):10006–10011.
46. Pember RB, Howell LH, Bell JB, Colella P, Crutchfield WY, Fiveland WA, Jessee JP. An adaptive projection method for unsteady, low-mach number combustion. *Combustion Science Technology* 1998; **140**:123–168.
47. Skamarock WC, Klemp JB. Adaptive grid refinement for two-dimensional and three-dimensional nonhydrostatic atmospheric flow. *Monthly Weather Review* 1993; **121**:788–804.
48. Barad MF, Colella P. A fourth-order accurate local refinement method for Poisson's equation. *Journal of Computational Physics* 2005; **209**(1):1–18.
49. Bell J, Berger M, Saltzman J, Welcome M. Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM Journal on Scientific Computing* 1994; **15**(1):127–138.
50. Bell JB, Shubin GR. An adaptive grid finite difference method for conservation laws. *Journal of Computational Physics* 1983; **52**(3):569–591.
51. Berger MJ, Colella P. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics* 1989; **82**(1):64–84.
52. Berger MJ, Oliger JE. Adaptive mesh refinement for hyperbolic partial differential equations. *Technical Report*, Stanford, CA, U.S.A., 1983.

DOI: 10.1002/fld

53. Sussman MM, Almgren AS, Bell JB, Colella P, Howell LH, Welcome M. An adaptive level set approach for incompressible two-phase flows. *Journal of Computational Physics* 1999; **148**:81–124.
54. Berger MJ, Leveque RJ. An adaptive Cartesian mesh algorithm for the euler equations in arbitrary geometries. *AIAA Paper 89-1930CP*, 1989; 1–7.
55. Penven P, Debreu L, Marchesiello P, McWilliams JC. Evaluation and application of the roms 1-way embedding procedure to the central California upwelling system. *Ocean Modelling* 2006; **12**:157–187.
56. Fox AD, Maskell SJ. Two-way interactive nesting of primitive equation ocean models with topography. *Journal of Physical Oceanography* 1995; **25**:2977–2996.
57. Zhang D-L, Chang H-R, Seaman NL, Warner TT, Fritsch JM. A two-way interactive nesting procedure with variable domain resolution. *Monthly Weather Review* 1986; **114**:1330–1339.
58. Barad M, Colella P, Graves DT, Ligocki TJ, Modiano D, Schwartz PO, Van Straalen B. *EBChombo Software Package for Cartesian Grid*, *Embedded Boundary Applications*, unpublished, 2007.
59. Colella P, Graves DT, Ligocki TJ, Martin DF, Modiano D, Serafini DB, Van Straalen B. *Chombo Software Package for AMR Applications—Design Document*, unpublished, 2000.
60. Briggs WL, Henson VE, McCormick SF. *A Multigrid Tutorial* (2nd edn). Society for Industrial and Applied Mathematics: Philadelphia, PA, U.S.A., 2000.
61. Martin DF, Cartwright KL. Solving Poisson's equation using adaptive mesh refinement. *Technical Report UCB/ERI M96/66*, UC, Berkeley, 1996.
62. Pember RB, Bell JB, Colella P, Crutchfield WY. An adaptive Cartesian grid method for unsteady compressible flow in irregular regions. *Journal of Computational Physics* 1995; **120**(2):278–304.
63. Rider WJ. Filtering non-solenoidal modes in numerical solutions of incompressible flows. *International Journal for Numerical Methods in Fluids* 1998; **28**(5):789–814.
64. Almgren AS, Bell JB, Crutchfield WY. Approximate projection methods: part I. Inviscid analysis. *SIAM Journal on Scientific Computing* 2000; **22**:1139–1159.
65. Berger MJ, Rigoutsos I. An algorithm for point clustering and grid generation. *IEEE Transactions on Systems*, *Man*, *and Cybernetics* 1991; **21**(5):1278–1286.
66. Hartel C, Meiburg E, Necker F. Analysis and direct numerical simulation of the flow at a gravity-current head. Part 1. Flow topology and front speed for slip and no-slip boundaries. *Journal of Fluid Mechanics* 2000; **418**:189–212.
67. Jackson C. *An Atlas of Internal Solitary-like Waves and their Properties* (2nd edn). 2004. Available from: www.internalwaveatlas.com.
68. Munk W. Abyssal recipes. *Deep-Sea Research* 1966; **13**:707–730.
69. Munk W, Wunsch C. Abyssal recipes II: energetics of tidal and wind mixing. *Deep-Sea Research* 1998; **45**:1977–2010.
70. Thorpe SA. Recent developments in the study of ocean turbulence. *Annual Review of Earth and Planetary Sciences* 2004; **32**:91–109.
71. Cummins PF, Armi L, Vagle S. Upstream internal hydraulic jumps. *Journal of Physical Oceanography* 2006; **36**(5):753–769.
72. Farmer D, Armi L. The generation and trapping of solitary waves over topography. *Science* 1999; **283**(5399): 188–190.
73. Lamb KG. Numerical experiments of internal wave generation by strong tidal flow across a finite amplitude bank edge. *Journal of Geophysical Research* 1994; **99**(C1):843–864.
74. Michallet H, Ivey GN. Experiments on mixing due to internal solitary waves breaking on uniform slopes. *Journal of Geophysical Research* 1999; **104**:13 467–13 477.
75. Saric WS. Görtler vortices. *Annual Review of Fluid Mechanics* 1994; **26**:379–409.
76. Colella P, Bell J, Keen N, Ligocki T, Lijewski M, van Straalen B. Performance and scaling of locally-structured grid methods for partial differential equations. *Journal of Physics* (*Conference Series*, *Proceedings of SciDAC*), 2007; **78**. DOI: 10.1088/1742-6596/78/1/012013.
77. Colella P, Sekora MD. A limiter for PPM that preserves accuracy at smooth extrema. *Journal of Computational Physics* 2008; **227**(15):7069–7076.
78. Colella P, Woodward PR. The piecewise parabolic method (PPM) for gas-dynamical simulations. *Journal of Computational Physics* 1984; **54**:174–201.
79. A. Bourlioux, Layton AT, Minion ML. High-order multi-implicit spectral deferred correction methods for problems of reacting fluid flow. *Journal of Computational Physics* 2003; **189**:651–675.