Towards a High-Order Embedded Boundary Finite Volume Method for the Incompressible Navier-Stokes Equations with Complex Geometries

Nathaniel Overton-Katz *, Xinfeng Gao †, and Stephen Guzik ‡
Computational Fluid Dynamics and Propulsion Laboratory
Colorado State University, Fort Collins, CO, USA

Oscar Antepara §, Daniel T. Graves ¶, and Hans Johansen ‖
Computational Research Division
Lawrence Berkeley National Lab, Berkeley, CA, USA

In this paper, we present components of a fourth-order finite volume method that will eventually allow for solving the incompressible Navier-Stokes equations. The algorithm represents complex geometries on a Cartesian embedded boundary grid, and takes special steps to achieve stable and accurate discretizations. Spatial discretizations are based on a weighted least-squares technique that mitigates the "small cut cell" problem, without mesh modifications, cell merging, or redistribution. Solutions are advanced in time using a projection method coupled with a higher-order additive implicit-explicit (ImEx) Runge-Kutta method, where the diffusion and advection terms are treated implicitly and explicitly, respectively. We demonstrate convergence tests for a scalar advection-diffusion equation and an approximate projection solver and confirm fourth-order accuracy for complex geometries. These methods are key components of solvers for the incompressible Navier-Stokes equations, and indicate feasibility of achieving higher-order accuracy on an arbitrary cut cell mesh.

Notation

D	spatial dimension, indexed with d	$\boldsymbol{i}$	grid indices, e.g., (i, j, k)
$\mathcal{V}_{\boldsymbol{i}}$	volume of a cell $\boldsymbol{i}$	$\boldsymbol{e}^d$	unit vector in direction d
$\mathcal{A}_{\boldsymbol{f}}$	area of a face $\boldsymbol{f}$	$\boldsymbol{f}$	face indices, e.g.
κ	volume fraction relative to the Cartesian cell	$\langle\cdot\rangle$	the cell-averaged or face-averaged quantity
$\hat{\mathbf{n}}$	surface unit normal vector	$\mathbf{A}$	the advection operator
$\mathbf{x}$	location in space, e.g., (x, y, z)	$\mathbf{D}$	the divergence operator
$\vec{\mathbf{F}}$	flux tensor, components $\mathbf{F}_d$	$\mathbf{G}$	the gradient operator
$\mathbf{u}$	velocity vector, components u_d	$\mathbf{L}$	the Laplacian operator
p	pressure scalar	$\mathbb{P}$	projection operator
		$O(h)$	order of accuracy proportional to a cell length

I. Introduction

The finite volume method is inherently conservative in its construction, which is a valuable property when solving for fluid flows with combustion or shock waves present. When operating on structured grids, finite volume methods have been developed to achieve high solution accuracy and code performance. This is demonstrated by the Chombo framework [1], where adaptive mesh refinement is one of the key features for improving solution accuracy with a high

*PhD Student, noverton@colostate.edu, Student AIAA member
†Associate Professor, Xinfeng.Gao@colostate.edu, Senior AIAA member
‡Assistant Professor, Stephen.Guzik@colostate.edu, Senior AIAA member
§Researcher, oantepara@lbl.gov
¶Researcher, dtgraves@lbl.gov
‖Researcher, hjohansen@lbl.gov

degree of parallelism. High-order methods have successfully been created with structured grid solvers [2] due to the convenience of polynomial reconstructions with regular data. Despite these advantages, representation of complex geometries on structured grids is a significant challenge. Technologies such as mapped multi-block methods allow for moderately complex structured grid geometries by combination of several curvilinear grids. However, mapped multi-block grids struggle to represent rough surfaces, and creating quality grids is generally time-consuming.

Alternatively, embedded boundary grids have little restriction in geometries that may be represented, while maintaining the advantages of structured grid solvers. Embedded boundary grids are created by first generating a Cartesian grid independent of boundary geometry. Then, the boundary geometry is overlain on the grid, and cells that intersect the boundaries are cut (Fig. 1). The resulting grid is one that is structured everywhere away from the boundary, while near the boundary the grid is made up of partial, or "cut" cells. The resulting method retains the advantages of solving on structured grids on the interior of the domain, while having relatively little restriction in geometries that may be represented. Additionally, this grid generation process can be done efficiently and quickly. However, the presence of cut cells brings about additional challenges that must be overcome. Elaborate reconstruction schemes are required near the boundaries of embedded boundary methods, since regular or grid-aligned stencils are not accurate or stable in the presence of cut cells. Volumes of cut cells may also be arbitrarily small with respect to the Cartesian cells they originate from, and maintaining stability of these small cells requires special care.

The embedded boundary method has been used with great success in a number of complex fluid dynamics applications [3–6]. Previous time dependent applications have only achieved up to second-order accurate solutions, with first-order or inconsistent results near embedded boundaries [7]. A high-order finite volume embedded boundary method for smooth and kinked (C^0) domains was demonstrated for Poisson's equation by Devendran et al [8]. The goal of this work is to extend the fourth-order embedded boundary method to solve the time dependent incompressible Navier-Stokes equations. Advancing the diffusion terms in time is straightforward using a fourth-order implicit time marching method. Advection can be treated with explicit time marching methods, but doing so requires special consideration for arbitrarily small cut cells. These two schemes can be combined using an additive Runge-Kutta (ARK) scheme coupled with a projection method to solve the incompressible Navier-Stokes equations.

II. The Finite-Volume Method for Embedded Boundaries

Finite volume methods express time dependent PDEs in "flux-divergence" form of

$$\frac{\partial}{\partial t}\mathbf{u}(\mathbf{x},t) + \nabla\cdot\vec{\mathbf{F}}(\mathbf{u}) = 0\,, \tag{1}$$

based on conservation laws over discrete control volumes. To apply the finite volume method, the system of PDEs is converted to integral form over volumes $\mathcal{V}_{\boldsymbol{i}}$, and the divergence theorem is applied to the flux term $\vec{\mathbf{F}}$ yielding

$$\frac{\partial}{\partial t}\int_{\mathcal{V}_i}\mathbf{u}\,\mathrm{d}\mathbf{x} + \int_{\partial\mathcal{V}_i}\vec{\mathbf{F}}\cdot\hat{\mathbf{n}}\,\mathrm{d}\mathbf{x} = 0\,.$$

The flux term may be broken into separate regions over the volume surface $\partial\mathcal{V}_{\boldsymbol{i}}$ as

$$\int_{\partial\mathcal{V}_i}\vec{\mathbf{F}}\cdot\hat{\mathbf{n}}\,\mathrm{d}\mathbf{x} = \sum_{\boldsymbol{f}\in\partial\mathcal{V}_i}\int_{\mathcal{A}_f}\vec{\mathbf{F}}\cdot\hat{\mathbf{n}}_{\boldsymbol{f}}\,\mathrm{d}\mathbf{x}\,,$$

where $\boldsymbol{f}$ indexes a particular face of the cell indexed by $\boldsymbol{i}$, and $\hat{\mathbf{n}}_{\boldsymbol{f}}$ is the corresponding outward normal vector.

Averaged quantities are defined as

$$\langle\mathbf{u}\rangle_{\boldsymbol{i}} = \frac{1}{\mathcal{V}_{\boldsymbol{i}}}\int_{\mathcal{V}_i}\mathbf{u}\,\mathrm{d}\mathbf{x}\,,\qquad \langle\mathbf{F}\rangle_{\boldsymbol{f}} = \frac{1}{\mathcal{A}_{\boldsymbol{f}}}\int_{\mathcal{A}_f}\vec{\mathbf{F}}\cdot\hat{\mathbf{n}}_{\boldsymbol{f}}\,\mathrm{d}\mathbf{x}\,.$$

Using cell averaged notation, the finite volume scheme is expressed in terms of averages as

$$\frac{\partial}{\partial t}\langle\mathbf{u}\rangle_{\boldsymbol{i}} + \sum_{\boldsymbol{f}\in\partial\mathcal{V}_i}\frac{\mathcal{A}_{\boldsymbol{f}}}{\mathcal{V}_{\boldsymbol{i}}}\langle\mathbf{F}\rangle_{\boldsymbol{f}} = 0\,. \tag{2}$$

This is an exact formulation, as no approximations have been made. In practice, the fluxes and time derivatives are not known exactly, so a numerical solution is created by producing approximations for each. Cut cells can potentially be

arbitrarily small in a limit where $\mathcal{V}_i$ approaches zero. This is numerically troublesome, since the flux term is potentially divided by zero, and the other geometric quantities and stencils must have similar limits to avoid this. Thus numerical stability for small cells requires extra care, and is dealt with by scaling with a cell volume fraction, as detailed in Devendran et al [8].

A. Projection Formulation for the Incompressible Navier-Stokes Equations

The incompressible Navier-Stokes equations with constant density are given by

$$\frac{\partial}{\partial t}\mathbf{u} + \nabla \cdot (\mathbf{uu}) = -\nabla p + \nu \Delta \mathbf{u}\,, \tag{3}$$

$$\nabla \cdot \mathbf{u} = 0\,, \tag{4}$$

where $\mathbf{u}$ is the flow velocity, p the pressure, and ν the kinematic viscosity.

A Hodge projection operator $\mathbb{P}$ has been used in the finite volume literature [9, 10] to enforce a divergence-free velocity field:

$$\mathbb{P}(\mathbf{w}) = \mathbf{v}\,, \tag{5}$$

$$\nabla \cdot \mathbf{v} = 0 \text{ where}\,, \tag{6}$$

$$\mathbb{P}(\mathbf{w}) \equiv \left(\mathbb{I} - \nabla \Delta^{-1} \nabla \cdot\right) \mathbf{w}\,. \tag{7}$$

B. Finite Volume Projection Formulation

We choose discretizations for each of the spatial operators in Eqs. (3-4), and write the resulting discrete equations at grid locations $\boldsymbol{i}$ as

$$\frac{\partial}{\partial t}\mathbf{u}_{\boldsymbol{i}} = -\mathbf{A}(\mathbf{u})_{\boldsymbol{i}} - (\mathbf{G}p)_{\boldsymbol{i}} + \nu(\mathbf{L}\mathbf{u})_{\boldsymbol{i}}\,, \tag{8}$$

$$(\mathbf{D}\mathbf{u})_{\boldsymbol{i}} = 0\,, \tag{9}$$

where $\mathbf{A}, \mathbf{D}, \mathbf{G}$, and $\mathbf{L}$ are fourth-order finite volume approximations of advection, divergence, gradient, and Laplacian terms, respectively. (From this point forward, we will drop the subscript $\boldsymbol{i}$ except for clarity.) The goal is to discretize these operators so that

$$\mathbf{A}\mathbf{u} = \nabla \cdot (\mathbf{uu}) + O(h^4)\,, \tag{10}$$

$$\mathbf{D}\mathbf{u} = \nabla \cdot \mathbf{u} + O(h^4)\,, \tag{11}$$

$$\mathbf{G}\mathbf{u} = \nabla \mathbf{u} + O(h^4)\,, \tag{12}$$

$$\nu\mathbf{L}\mathbf{u} = \nu \nabla \cdot \nabla \mathbf{u} + O(h^4) \tag{13}$$

in the regular interior of the domain, with some potential loss of accuracy near boundaries and in cut cells.

Because we are using co-located cell-average velocity and pressure, we take the approach of an *approximate* projection [11]. Instead of a strictly zero discrete divergence, we allow $\mathbf{u}$ to have a divergence that is at the level of the discretization error. The equivalent discrete projection is

$$\mathbb{P}(\mathbf{w}) = \left(\mathbb{I} - \mathbf{G}\mathbf{L}^{-1}\mathbf{D}\right) \mathbf{w}\,, \tag{14}$$

which requires the inversion of the Laplacian operator over the whole domain. Using the projection operator, the incompressible flow equations can be approximated by

$$\frac{\partial}{\partial t}\mathbf{u} = \mathbb{P}\left(-\mathbf{A}(\mathbf{u}) + \nu\mathbf{L}\mathbf{u}\right)\,, \tag{15}$$

$$\mathbf{D}\mathbf{u} = O(h^4)\,. \tag{16}$$

The general procedure for solving this system separates into an intermediate update that integrates the right-hand side of Eq. (16), and then projects that update using Eq. (14), to maintain an approximately-divergence free stage value for the time integrator. This is essentially a higher-order accurate version of the projection operator described in [7].

1. Advection-Diffusion

In the projected form of the incompressible Navier-Stokes equations given by Eq. (16) there are two components required for a solution: the projection operator, and an advection-diffusion time integration scheme. As building blocks for an incompressible Navier-Stokes solver, the projection and advection-diffusion schemes can be developed and tested separately. Showing stability and correctness of the approximate projection is required for the incompressible Navier-Stokes equations. To show algorithm verification, the scalar form of the advection-diffusion equation

$$\frac{\partial}{\partial t}\phi = \mathbb{P}\left(-\mathbf{A}(\phi) + \nu \mathbf{L}\phi\right) , \tag{17}$$

is sufficient to prove the order of accuracy of the high-order embedded boundary algorithm developed. Once the components are proven correct, the process of combining these pieces to form an incompressible Navier-Stokes solver should be straightforward. The only remaining challenge is introducing the non-linear vector advection term, $\mathbf{A}(\mathbf{u})$.

2. Projection Formulation for Open Boundaries

We must specify boundary conditions for the projection operators for open domain boundaries to split a given $\mathbf{w}$ into three separate components: a scalar potential flow solution, ψ, which only satisfies the boundary conditions, and the remainder which is split into pure gradient, ϕ, and divergence-free parts: $\mathbf{w} = \mathbf{G}\psi + \mathbf{v} + \mathbf{G}\phi$, such that

$$\mathbf{L}\psi = \mathbf{D}\mathbf{G}\psi = 0, \quad \mathbf{G}\psi \cdot \hat{\mathbf{n}} = \mathbf{u} \cdot \hat{\mathbf{n}} \text{ (potential)}, \tag{18}$$

$$\mathbf{D}\mathbf{v} = 0, \quad \mathbf{v} \cdot \hat{\mathbf{n}} = 0 \text{ (divergence-free)}, \tag{19}$$

$$\mathbf{L}\phi = \mathbf{D}\mathbf{G}\phi = \mathbf{D}\mathbf{w}, \quad \mathbf{G}\phi \cdot \hat{\mathbf{n}} = 0 \text{ (gradient)}. \tag{20}$$

In the end, our desired divergence-free velocity field that satisfies the correct boundary conditions is just the components $\mathbf{u} = \mathbf{v} + \mathbf{G}\psi$.

III. Embedded Boundary Spatial Discretization

In the embedded boundary (EB) approach, cells fall into one of three categories; regular cells, irregular cells, and invalid cells. This distinction between cell types is illustrated in Fig. 1. Regular cells are those that are full Cartesian cells. Irregular cells, or cut cells, are those which are partial cells because they intersect with the boundary geometry. Invalid cells, as the name indicates, are cells that fall outside the domain boundaries and are thus not in the solution domain. To denote the embedded boundary regions, let Ω be the irregular domain and $\Upsilon_{\boldsymbol{i}}$ be any regular cell, then a particular cell can be denoted by $\mathcal{V}_{\boldsymbol{i}} = \Upsilon_{\boldsymbol{i}} \cap \Omega$.

The challenge of embedded boundary methods is to approximate the flux terms $\langle \mathbf{F} \rangle_{\boldsymbol{f}}$ when regular grid stencils can not be used due to nearby cut cells. To solve this requires knowledge of the flux as a function, so that face averages may be computed appropriately. A general reconstruction is done by creating local polynomials and evaluating their value and derivatives on the face as needed. For a structured grid, reconstructed polynomials lead to grid-aligned regular stencils. When using an embedded boundary method, the cells near the boundary require a consistent approach to generate stencils that depend on the local geometry. On irregular regions, our approach is to use a weighted least-squares polynomial approximation from the cell average values in a local region of neighboring cells. The scheme presented theoretically can produce any order spatial discretization desirable, but for the context of this paper, fourth-order is the focus. The present study continues and extends the work developed by Devendran et al.[8], and the procedure is briefly summarized in this section.

A. Multi-Dimensional Taylor Expansion

A multi-dimensional polynomial can be defined using

$$(\mathbf{x} - \bar{\mathbf{x}})^{\mathbf{q}} = \prod_{d=1}^{\mathrm{D}} (x_d - \bar{x}_d)^{\mathrm{q}_d} , \qquad \mathbf{q}! = \prod_{d=1}^{\mathrm{D}} \mathrm{q}_d ! , \qquad |\mathbf{q}| = \sum_{d=1}^{\mathrm{D}} \mathrm{q}_d ,$$

where $\mathbf{q}$ is a *multi-index* or D dimensional non-negative integer vector, and $\bar{\mathbf{x}}$ is a given point in space. For a sufficiently smooth scalar function ϕ, its multi-dimensional Taylor series of order Q can be written as

$$\phi(\mathbf{x}) = \sum_{|\mathbf{q}|<\mathrm{Q}} \frac{1}{\mathbf{q}!} \phi^{(\mathbf{q})}(\bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^{\mathbf{q}} + O(h^{\mathrm{Q}}) , \tag{21}$$

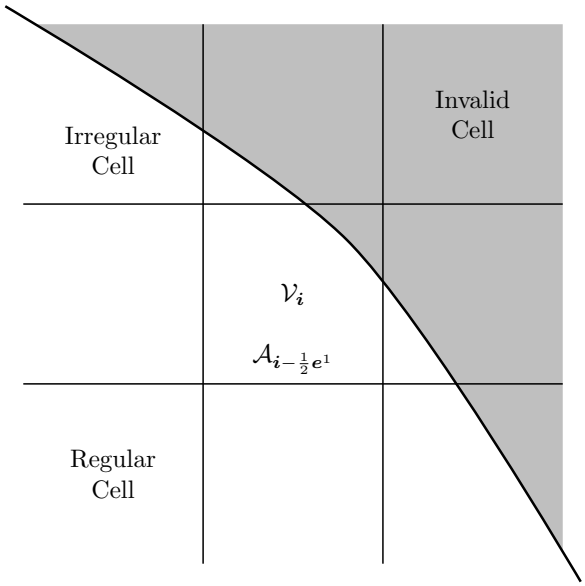


Fig. 1 Illustration of notation used for embedded boundary methods. The shaded region lies outside the problem domain of interest.

where the convention $0! = 1$ is used, and $\phi^{(\mathbf{q})}$ is the multi-index partial derivative notation

$$\phi^{(\mathbf{q})}(\mathbf{x}) = \left(\prod_{d=1}^{\mathrm{D}} \frac{\partial^{\mathrm{q}_d}}{\partial x_d^{\mathrm{q}_d}} \right) \phi(\mathbf{x}) \,,$$

and any $\mathrm{q}_d = 0$ implies no derivative. In formulating multi-dimensional polynomials to fit cell averaged data, integration over cells is needed. Moments are defined to be the integration of basis polynomials over a region, as

$$m_{\boldsymbol{i}}^{\mathbf{q}}(\bar{\mathbf{x}}) = \int_{\mathcal{V}_{\boldsymbol{i}}} (\mathbf{x} - \bar{\mathbf{x}})^{\mathbf{q}} \, \mathrm{d}\mathbf{x} \,, \tag{22}$$

for volume moments, while for face moments

$$m_{\boldsymbol{f}}^{\mathbf{q}}(\bar{\mathbf{x}}) = \int_{\mathcal{A}_{\boldsymbol{f}}} (\mathbf{x} - \bar{\mathbf{x}})^{\mathbf{q}} \, \mathrm{d}\mathbf{x} \,. \tag{23}$$

B. Flux Reconstruction

To reconstruct a high-order solution from cell averaged data, we use a Taylor expansion about cell centers $\bar{\mathbf{x}}_{\boldsymbol{i}}$. By defining the multi-dimensional polynomial coefficients $c_{\boldsymbol{i}}^{\mathbf{q}} = \frac{1}{\mathbf{q}!} \mathrm{u}_d^{(\mathbf{q})}(\bar{\mathbf{x}}_{\boldsymbol{i}})$, approximated cell averages can be represented as

$$\begin{aligned} \langle \mathrm{u}_d \rangle_{\boldsymbol{j}} &= \frac{1}{\mathcal{V}_{\boldsymbol{j}}} \int_{\mathcal{V}_{\boldsymbol{j}}} \mathrm{u}_d \, d\mathbf{x} \\ &= \frac{1}{\mathcal{V}_{\boldsymbol{j}}} \int_{\mathcal{V}_{\boldsymbol{j}}} \sum_{|\mathbf{q}|<\mathrm{Q}} \frac{1}{\mathbf{q}!} \mathrm{u}_d^{(\mathbf{q})}(\bar{\mathbf{x}}_{\boldsymbol{i}})(\mathbf{x} - \bar{\mathbf{x}}_{\boldsymbol{i}})^{\mathbf{q}} + O(h^{\mathrm{Q}}) \, d\mathbf{x} \\ &= \frac{1}{\mathcal{V}_{\boldsymbol{j}}} \sum_{|\mathbf{q}|<\mathrm{Q}} c_{\boldsymbol{i}}^{\mathbf{q}} m_{\boldsymbol{j}}^{\mathbf{q}}(\bar{\mathbf{x}}_{\boldsymbol{i}}) + O(h^{\mathrm{Q}}) \,. \end{aligned} \tag{24}$$

The number of unknown coefficients is determined by $p = \frac{(\mathrm{D}+\mathrm{Q})!}{\mathrm{D}!\,\mathrm{Q}!}$. Determining the p coefficients requires solving a linear system of equations including at least the same number of cell values $\langle \mathrm{u}_d \rangle_{\boldsymbol{j}}$. If the values $\langle \mathrm{u}_d \rangle_{\boldsymbol{j}}$ are more than

enough linearly independent values to determine each coefficient, this is an *over-determined* linear least-squares system. Note that there are degenerate cases where values are *almost* linearly dependent, in which case the least-squares system will typically distribute the solution between those values.

Thus, given a vector U that is a vector of neighboring cell-average velocities $\langle \mathrm{u}_d \rangle_{\boldsymbol{j}}$, using matrix notation we can write the approximation as

$$U \approx M\, C\,,$$

where the geometric moment matrix M comes from Eq. (24) for each of the neighboring cells $\boldsymbol{j}$, and the multi-dimensional coefficients vector C is to be determined.

A diagonal weighting matrix W is added to enable further control of the least-squares fit with the goal to improve stability as in [8]; further discussion is in section III.C. With the weighting matrix, this yields the system and solution for the coefficients as

$$C = \arg\min_{\tilde{C}} \left\| WU - WM\tilde{C} \right\|_2 \rightarrow C = (W\,M)^{\dagger}\, W\,U\,, \tag{25}$$

where $(W\,M)^{\dagger}$ indicates the pseudo-inverse of $(W\,M)$, which is equivalent to a true inverse when the matrix is square.

1. Higher-order EB Viscous Flux Stencil

Using this method of reconstruction gives a way to evaluate fluxes for Eq. (2). In the convenient case when the flux is linear, such as the diffusion term, a Taylor expansion is applied as

$$\begin{aligned}
\mathcal{A}_{\boldsymbol{f}} \langle \mathrm{F}_d^V \rangle_{\boldsymbol{f}} &= \int_{\mathcal{A}_{\boldsymbol{f}}} \nabla \mathrm{u}_d \cdot \hat{\mathbf{n}}_{\boldsymbol{f}}\, \mathrm{d}\mathbf{x} + O(h^{\mathrm{Q}}) \\
&= \int_{\mathcal{A}_{\boldsymbol{f}}} \nabla \left(\sum_{|\mathbf{q}|<\mathrm{Q}} c_{\boldsymbol{f}}^{\mathbf{q}} (\mathbf{x} - \bar{\mathbf{x}}_{\boldsymbol{f}})^{\mathbf{q}} \right) \cdot \hat{\mathbf{n}}_{\boldsymbol{f}}\, \mathrm{d}\mathbf{x} \\
&\equiv \sum_{|\mathbf{q}|<\mathrm{Q}} c_{\boldsymbol{f}}^{\mathbf{q}} G_{\boldsymbol{f}}^{\mathbf{q}}\,,
\end{aligned}$$

This too is a linear equation, which can be expressed in matrix form as in Eq. (25) to derive an expression for the corresponding stencil as in [8]:

$$\begin{aligned}
\mathcal{A}_{\boldsymbol{f}} \langle \mathrm{F}_d^V \rangle_{\boldsymbol{f}} &= G\, C \\
&= G\, (W\,M)^{\dagger}\, W\,U \\
&\equiv S_{\boldsymbol{f}}^{\mathrm{T}}\, U\,.
\end{aligned}$$

where C is the same coefficient as before, and $G = [\cdots G_{\boldsymbol{f}}^{\mathbf{q}} \cdots]$, and

$$G_{\boldsymbol{f}}^{\mathbf{q}} = \sum_{d=1}^{\mathrm{D}} \int_{\mathcal{A}_{\boldsymbol{f}}} \frac{\partial}{\partial_{x_d}} \left((\mathbf{x} - \bar{\mathbf{x}}_{\boldsymbol{f}})^{\mathbf{q}} \right) \hat{\mathbf{n}}_{\boldsymbol{f},d}\, \mathrm{d}\mathbf{x} = \sum_{d=1}^{\mathrm{D}} \mathrm{q}_d m_{\boldsymbol{f},d}^{\mathbf{q}-\boldsymbol{e}^d}\,. \tag{26}$$

The viscous flux stencil $S_{\boldsymbol{f}}$ depends only on the local neighbors selected, their moments, and flux function, so it may be computed once per grid at initialization and cached as a sparse matrix operator.

2. Higher-order EB Advection Flux Stencil

Unlike the diffusion term, the scalar advection term does not use boundary conditions since the solution is defined only by velocity characteristics. However, we can still evaluate the velocity average on faces as

$$\begin{aligned}
\langle \mathrm{u}_d \rangle_{\boldsymbol{f}} &= M_{\boldsymbol{f}}\, C \\
&= M_{\boldsymbol{f}}\, (W_{\mathrm{up}}\, M)^{\dagger}\, W_{\mathrm{up}}\, U \\
&= R_{\boldsymbol{f}}\, U\,.
\end{aligned}$$

Where $M_{\boldsymbol{f}}$ is the row vector of moments of the face $\boldsymbol{f}$, and W_{up} is an upwind-weight matrix that is evaluated using the face-centroid velocity field for simplicity. The reconstruction vector $R_{\boldsymbol{f}}$ may still be cached, since it only depends upon the grid.

For the advection term $\langle \nabla \cdot (\mathbf{uu}) \rangle$, we combine upwind polynomial approximations for $\mathbf{u}$ to create face-average fluxes using a *convolution* formula (see [12]):

$$\begin{aligned}
\langle \vec{\mathbf{F}} \cdot \hat{\mathbf{n}}_d \rangle &= \langle \mathrm{u}_d \mathbf{u} \rangle + O(h^4) \\
&\equiv \left(s_{\mathrm{avg}}^{\mathrm{T}} \mathrm{u}_d \right) \left(s_{\mathrm{avg}}^{\mathrm{T}} \mathbf{u} \right) + \frac{\mathcal{A}_f^2}{12} \left(s_1^{\mathrm{T}} \mathrm{u}_d \right) \left(s_1^{\mathrm{T}} \mathbf{u} \right) && (27) \\
s_{\mathrm{avg}} &= W_{\mathrm{up}} \left(M W_{\mathrm{up}} \right)^{\dagger} f_{\mathrm{avg}} && (28) \\
s_1 &= W_{\mathrm{up}} \left(M W_{\mathrm{up}} \right)^{\dagger} f_1 && (29) \\
f_{\mathrm{avg}} &= \text{ vector that evaluates coefficients to get fourth-order } \langle \mathbf{u} \rangle_{\boldsymbol{f}} && (30) \\
f_1 &= \text{ vector that evaluates coefficients to get second-order } \langle \partial_{d'} \mathbf{u} \rangle_{\boldsymbol{f}}, \text{ tangential to the face.} && (31)
\end{aligned}$$

In 2D, Eq. (27) holds because odd moments of the face around the centroid $\bar{\mathbf{x}}$ are 0, so the convolution formula only involves even moments, which are simply powers of $\mathcal{A}_f$. With some algebraic manipulation (see [12]) for $O(h^4)$, it reduces to this formula. However, in 3D or higher-order in 2D, there are additional terms that must be included for the general case without this cancellation.

3. Approximate Projection

The higher-order projection operator must start with cell average velocities, $\langle \mathbf{u} \rangle_{\boldsymbol{i}}$, and correct them to be *approximately* divergence-free, Eq. (16). To accomplish this, we require discretizations and boundary conditions for each operator in Eq. (14).

First, the Laplacian operator $\mathbf{L}$ is essentially the same as the viscous flux in section III.B.1, but with different boundary conditions based on Eq. (18), that are homogeneous Neumann for inflow or solid walls, and homogeneous Dirichlet for outflow.

For the cell-average gradient operator, $\mathbf{G}$, the least-squares polynomial fit is again used to evaluate a stencil just as in Eq. (26). In this case, as it is not a finite volume flux, cell moments are used instead of face moments:

$$G_{d,\boldsymbol{i}}^{\mathbf{q}} = \sum_d \int_{\mathcal{V}_{\boldsymbol{i}}} \frac{\partial}{\partial_{x_d}} \left((\mathbf{x} - \bar{\mathbf{x}}_{\boldsymbol{i}})^{\mathbf{q}} \right) \mathrm{d}\mathbf{x} = \mathrm{q}_d m_{\boldsymbol{i}}^{\mathbf{q} - \boldsymbol{e}^d} \,. \tag{32}$$

The gradient operator uses the same boundary conditions as the Laplacian.

Finally, we define a cell-average Divergence operator using the divergence theorem,

$$\int_{\mathcal{V}_{\boldsymbol{i}}} \nabla \cdot \mathbf{u} \, \mathrm{d}\mathbf{x} = \int_{\partial \mathcal{V}_{\boldsymbol{i}}} \mathbf{u} \cdot \hat{\mathbf{n}} \, \mathrm{d}\mathbf{x} \,,$$

so the cell average of the divergence of the velocity field can be determined from face average quantities as

$$\langle \nabla \cdot \mathbf{u} \rangle_{\boldsymbol{i}} = \sum_{\boldsymbol{f} \in \partial \mathcal{V}_{\boldsymbol{i}}} \frac{\mathcal{A}_{\boldsymbol{f}}}{\mathcal{V}_{\boldsymbol{i}}} \langle \mathbf{u} \cdot \hat{\mathbf{n}}_{\boldsymbol{f}} \rangle_{\boldsymbol{f}} \,.$$

These face averages are similar to those in the advection flux, however, without any upwind weighting for the reconstruction. The terms on regular faces have single component normal vectors $\hat{\mathbf{n}}_{\boldsymbol{f}} = \boldsymbol{e}^d$, and as a result $\langle \mathbf{u} \cdot \hat{\mathbf{n}}_{\boldsymbol{f}} \rangle_{\boldsymbol{f}} = \langle \mathbf{u}_d \rangle_{\boldsymbol{f}}$. Boundary conditions for the divergence are $\mathbf{u} \cdot \hat{\mathbf{n}} = 0$ at any solid walls, including EB boundaries. At inflow boundaries, the velocity is specified, while on outflow no boundary conditions are used, as in [7].

4. Physical Boundary Conditions

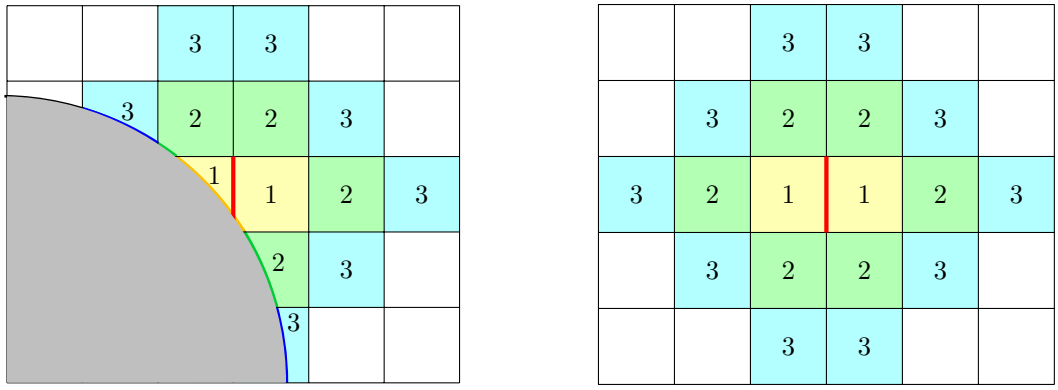
For regions where stencils encompass a Dirichlet boundary condition, the boundary value is defined as $\mathbf{u}_{\text{bc}} = \langle \mathbf{u} \rangle_{\boldsymbol{f}}$ when $\boldsymbol{f}$ corresponds to a face with a prescribed value. A polynomial can be reconstructed to fit the face value using

$$\begin{aligned} \mathcal{A}_{\boldsymbol{f}} \mathrm{u}_{\mathrm{bc},d} &= \int_{\mathcal{A}_{\boldsymbol{f}}} \mathrm{u}_d \, \mathrm{d}\mathbf{x} \\ &= \int_{\mathcal{A}_{\boldsymbol{f}}} \sum_{|\mathbf{q}|<\mathrm{Q}} c_{\boldsymbol{f}}^{\mathbf{q}} (\mathbf{x} - \bar{\mathbf{x}}_{\boldsymbol{f}})^{\mathbf{q}} \, \mathrm{d}\mathbf{x} \\ &= \sum_{|\mathbf{q}|<\mathrm{Q}} c_{\boldsymbol{f}}^{\mathbf{q}} m_{\boldsymbol{f}}^{\mathbf{q}} \, . \end{aligned}$$

Including this additional equation and value into the stencil system will match the boundary condition in a least-squares sense, with a similar method for Neumann boundaries. These extra boundary condition equations are used when any neighboring cell included in the reconstruction contains a portion of the boundary, and so can accommodate different parts of the boundary (such as corners, etc.).

5. Interpolation Neighborhoods

An important aspect of the weighted least squares method is neighborhood selection, choosing which neighboring cells to include in the stencil, which becomes important when dealing with embedded boundaries. In this paper, the focus is on fourth-order accurate stencils in two dimensions, which require a minimum of 10 neighbors for flux reconstruction in the most general case, which must span both the x and y directions to estimate higher-order derivatives. To achieve this with the developed weighted least squares approach, stencils of radius 3 cells from the reconstructed face are used (see Fig. 2). These stencils encompass 18 cells in regular regions. For stencils in embedded boundary regions, the same radius of 3 cells is used along with the inclusion of boundary conditions.



(a) An example of an irregular flux stencil, using 13 cells and 6 boundary conditions.

(b) Regular flux stencils with 18 cells an no boundary conditions.

Fig. 2 An illustration of stencil neighborhoods for flux construction. The stencils are for the red highlighted faces, with cells numbered according to the neighbor distance from the face.

C. The Problem of Small Cut Cells

To reconstruct the stencils required for the spatial discretization, the weighted least-squares method Eq. (25) is used to simplify generation of stencils near the embedded boundaries. First, a large neighborhood is created to interpolate from, which includes many potential neighboring cells, some of which may be very small cut cells. These cut cells with potential small cell volumes can result in a poorly conditioned system of equations. Rather than exhaustively choosing an interpolation neighborhood which is better conditioned, a weighting scheme is chosen to assign relative importance to each cell's entry in the WLS system. Smaller relative weights mean that cells has less importance and a smaller absolute stencil value. An effective weighting for cell $\boldsymbol{i}$ with target construction face $\boldsymbol{f}$ of

$$W_{\boldsymbol{i},\boldsymbol{f}} = D_{\boldsymbol{i},\boldsymbol{f}}^{-5} \, , \tag{33}$$

where $D_{\boldsymbol{i},f}$ is the Euclidean distance, has been shown to increase solution stability and spectral properties [8], especially when interpolation neighborhoods are large and there are many possible consistent stencils.

IV. Time Marching Method

When solving the Navier-Stokes equations, time constraints for the advection and diffusion portions of the fluxes can be significantly different. The advection terms are hyperbolic in nature and generally non-linear and less stiff, making them well suited for explicit time marching methods. In contrast, the diffusion fluxes are parabolic, more stiff, and linear, making them well suited for implicit time marching methods using fast linear solvers. In the context of embedded boundary methods, where cut cells can become arbitrarily small, this difference of time step stability constraints between advection and diffusion terms is can be orders of magnitude different. To maintain reasonable time step sizes, a hybrid implicit-explicit (ImEx) Runge-Kutta method is chosen for the embedded boundary algorithm [13]. This allows for the advection terms to be updated using an explicit method, and the diffusion terms with an implicit method. As a result, the time evolution is generally not limited by the small time steps required for the diffusion physics. At each stage of the implicit RK time marching, a large sparse matrix system must be solved. To do this efficiently, we use a geometric multigrid solver included in Chombo [14] with PETSC [15, 16] as a bottom solver.

To achieve the desired fourth-order accuracy of the presented algorithm, the ARK4(3)6L[2]SA time marching method [17] is used. This method has successfully been demonstrated for stiff, higher-order finite volume methods that use AMR for advection-diffusion problems [18].

V. Algorithm Verification

To show verification of the algorithm developed, the method of manufactured solutions is used. By producing an artificial exact solution, solution error can be computed and used for testing grid convergence. The method of manufactured solutions is explained in detail by Salari et al. [19]. First, a manufactured solution form is chosen. Once the manufactured solution is chosen, a source term and boundary conditions are derived by substituting the manufactured solution into the governing equations.

For algorithm verification with more complicated solutions is to use Richardson extrapolation to evaluate the order of convergence, by comparing coarser solutions to the finest solution in a series, and evaluate the relative errors for convergence rates.

To verify our algorithm, convergence tests are used to demonstrate the claimed solution order of accuracy. The solution error is computed, and convergence rates evaluated using each of the L_∞, L_2, and L_1 norms. The solution error is computed from cell averages as

$$E_{\boldsymbol{i}} = \langle \phi(\mathbf{x}, t) \rangle_{\boldsymbol{i}} - \langle \phi_{\text{exact}}(\mathbf{x}, t) \rangle_{\boldsymbol{i}} \, .$$

The solution norms are computed as

$$L_\infty = \max_{\boldsymbol{i} \in \Omega} |E_{\boldsymbol{i}}| \, ,$$

$$L_2 = \left(\frac{1}{N_c} \sum_{\boldsymbol{i} \in \Omega} E_{\boldsymbol{i}}^2 \right)^{\frac{1}{2}} \, ,$$

$$L_1 = \frac{1}{N_c} \sum_{\boldsymbol{i} \in \Omega} |E_{\boldsymbol{i}}| \, ,$$

where N_c is the number of cells in the domain Ω. Convergence can be tested by the expectation on a grid having cell size h that error $E^h = O(h^{\mathrm{Q}})$, and thus $\mathrm{Q} = \log \frac{E^{rh}}{E^h} / \log r$ should be observed when r is the refinement ratio.

VI. Results

To demonstrate the developed algorithm, solutions of projections, scalar advection, and diffusion were evaluated on a set of simple embedded boundary geometries. To begin with, verification is performed for each part of the developed algorithm. First, a potential flow case is examined to verify correctness of the projection operator, verify solution accuracy in space, and determine stability. Next, manufactured solutions for the scalar diffusion equation are tested to verify solution order of accuracy in space and time in the presence of boundary conditions. Following, analytic solutions for the scalar advection equations are used to verify the solution order of accuracy in space and time, and

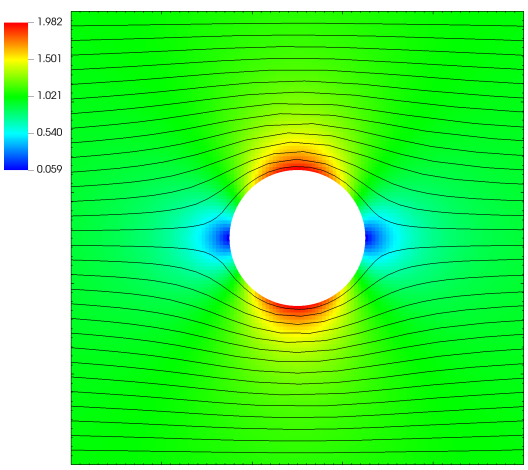
stability. Finally, the scalar advection and scalar diffusion equations are joined together, and again verified for solution order of accuracy.

A. Potential Flow for a Circle in a Channel

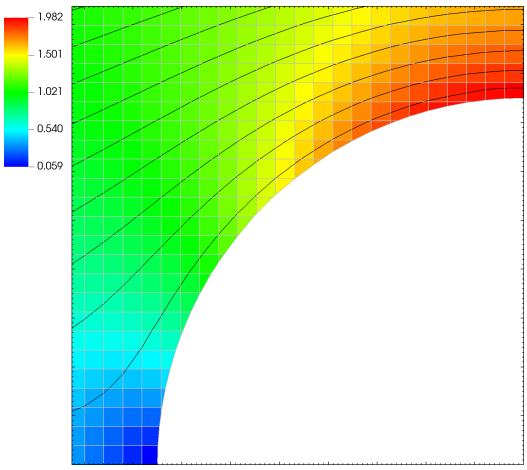
The approximate projection operator $\mathbb{P}(\mathbf{u})$ is demonstrated to be both fourth-order accurate and stable by solving for bounded potential flow. A geometry is generated by creating an embedded boundary grid with a circle of radius 0.15 placed in the center of a square domain of unit length, shown with a cell size of $h = 1/128$ in Fig. 3. The left boundary is specified as an inlet condition with unit velocity in the x-direction, the right boundary is specified by an outlet condition, and the remaining boundaries are slip-walls with zero normal velocity. The initial solution is a potential flow in an infinitely long domain evaluated for cell averages.

Stability of the approximate projection operator is demonstrated by showing that repeatedly applications of the projection on a velocity field reduces the divergence towards zero. The velocity divergence **Du** and pressure gradient $\mathbf{G}\phi$ from the projection are computed on a grid with cells size $h = 1/128$ for 100 iterations and plotted in Fig. 4. These quantities are shown to strictly decrease, showing stability of the high-order approximate projection.

Convergence tests are performed to ensure the spacial discretization achieves the targeted fourth-order accuracy. The projection is applied once for grids of decreasing refinement, and the divergence field **Du** is used to evaluate the solution error. The finest level is chosen with cell size $h = 1/128$, and subsequent coarser levels each double h. The L_1, L_2, and L_∞ errors and convergence rates are calculated and listed in Table 2. The tabulated convergence rates approach the expected fourth-order accuracy, but fail to maintain it consistently. Error in the solution is found to be focused around both the regular and embedded boundaries. The failure to reach consistent fourth-order accuracy is attributed to using an initial condition which does not exactly match the boundary conditions, but further investigation is required.



(a) The entire potential flow solution.

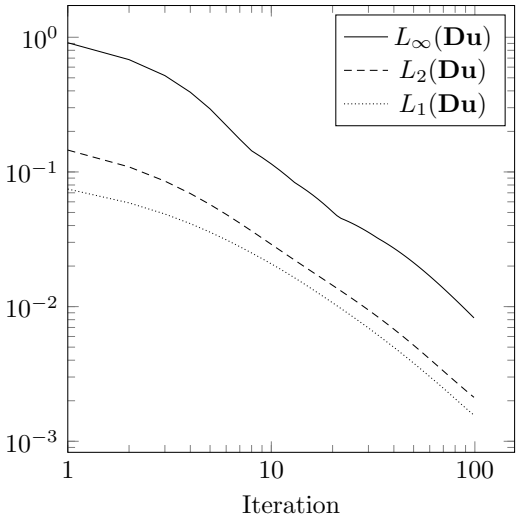


(b) A closeup view of potential flow, showing the grid lines in gray.

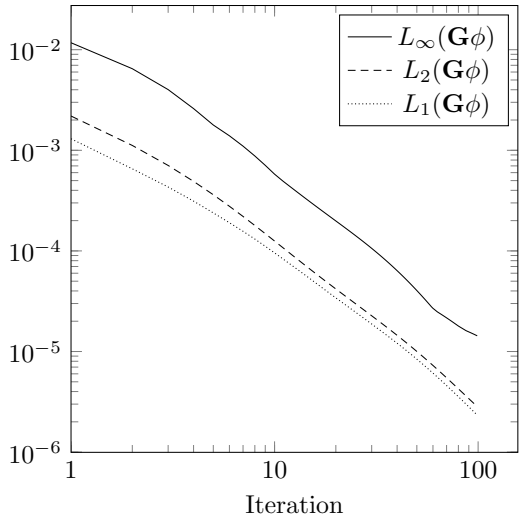
Fig. 3 Potential flow around a circle in a channel, where the left boundary is a uniform inlet, the right an outlet, and all other boundaries slip walls. The streamlines are shown in black, and the contours plot the velocity magnitude.

Table 1 Projection errors and convergence rates

N	L_1	Rate(L_1)	L_2	Rate(L_2)	L_∞	Rate(L_∞)
32	3.02930758e-03		1.04956527e-02		6.93443497e-02	
64	1.65516269e-04	4.193	8.52493945e-04	3.622	1.02405856e-02	2.759
128	1.48544088e-05	3.478	8.25666001e-05	3.368	1.15735534e-03	3.145



(a) Error norms of the divergence of velocity.



(b) Error norms of the pressure gradient.

Fig. 4 The L_1, L_2, and L_∞ norms from repeated projections of the channel bounded potential flow over a cylinder of grid size $h = 1/128$.

B. Manufactured Solution for Diffusion inside a Circle

To demonstrate the accuracy of the diffusion equation $\frac{\partial}{\partial t}\phi = \nu\mathbf{L}\phi$, a manufactured solution is employed. The algorithm targets fourth-order in space and time using the high-order stencils described in section IV and the ImEx scheme in section III. A circular domain is created of radius 0.3 centered about the point $(0.5, 0.5)$ The manufactured solution is defined on this domain by

$$\phi(\mathbf{x}, t) = \sin(2\pi t)\sin(R^2 - (\mathbf{x} - \mathbf{x}_0)^2) \tag{34}$$

where $R = 0.3$ matches the domain radius, and $\mathbf{x}_0 = (0.5, 0.5)$ to match the domain center. The embedded boundaries are specified by Dirichlet conditions with values determined by the manufactured solution. Following the method of manufactured solutions, a source term is added to balance the equation. The source term is evaluated explicitly in time, while the Laplacian term is evaluated implicitly. The solution is initialized to the fourth-order cell averaged form of the manufactured solution at time 0.125, and uses a dissipation rate of $\nu = 1$. On the finest level, of grid size $h = 1/128$, a time step of $\delta t = 0.1$ is taken to advance the solution forward for 128 steps. Subsequent coarser levels double the cell size and time step, while halving the time steps to reach the same end time. Using the chosen exact solution in Eq. (34), the L_1, L_2, and L_∞ errors and convergence rates are calculated and compiled in Table 2. The expected fourth-order accuracy is demonstrated in all error norms, verifying the algorithm for scalar diffusion.

Table 2 Scalar diffusion convergence errors and rates

N	L_1	Rate(L_1)	L_2	Rate(L_2)	L_∞	Rate(L_∞)
32	8.95629002-02		1.08337606-01		1.88167528-01	
64	7.83876744-03	3.514	9.09994654-03	3.574	1.65467909-02	3.507
128	3.31932144-04	4.562	4.02585953-04	4.499	7.89145300-04	4.390

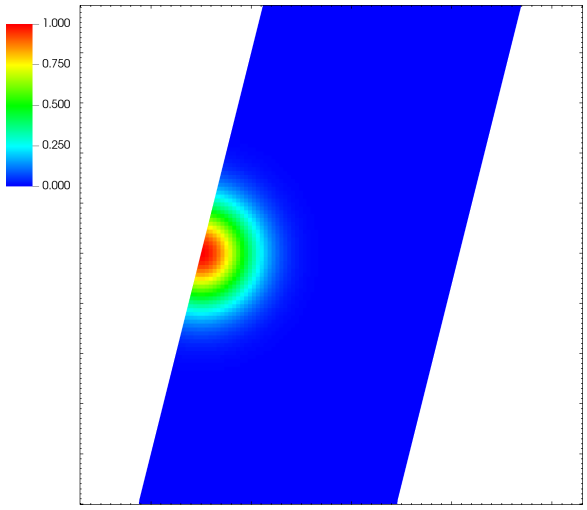
C. Advection inside a Periodic Channel

To show verification of the scalar advection equation $\frac{\partial}{\partial t}\phi = -\mathbf{A}(\phi)$, where in this case $\mathbf{A} = \nabla \cdot (\mathbf{u}\phi)$, comparison to an analytic solution is used. Analytic solutions to scalar advection are defined purely by shifts in the initial profile as a function of time. The algorithm targets fourth-order in space and time using the high-order stencils described in section IV and the explicit portion of the ImEx scheme in section III. The channel geometry is generated in a rectangle of unit

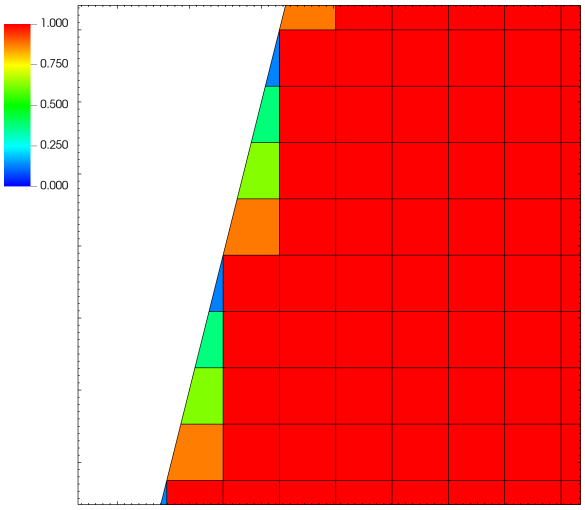
height. The left plane is defined by the normal vector $(-4, 1)$ and intersects the point $(0.5, 0.5)$. The right plane uses the same normal vector, but intersects the point $(1, 0.5)$. The top and bottom boundaries are periodic, and the left and right boundaries are slip walls. The initial and exact solution is a Gaussian profile is defined by

$$\phi(\mathbf{x}) = ae^{-\frac{|(\mathbf{x}-\mathbf{x}_0^2|}{\sigma^2}}. \tag{35}$$

The profile in Eq. (35), and shown in Fig. 5, is initialized along the left boundary about the point $\mathbf{x}_0 = (0.5, 0.5)$ with $a = 1$ and $\sigma^2 = 1e-2$, with velocity vector $\mathbf{u}$ set to unit magnitude aligned with the channel. In the finest case, the grid size is $h = 1/256$, and the time step corresponds to a CFL number of 1 based on the grid spacing. Each coarser level has the grid size and time step doubled from the finest. The profile is advected for a total of one unit in time, after which the solution should return to the initial profile. Using the exact solution, L_1, L_2, and L_∞ errors and convergence rates are calculated and compiled in Table 3. The expected fourth-order accuracy is observed in all error norms, even in small cells, verifying the algorithm for scalar advection. Of particular note, this solution achieves a stable high-order explicit advection scheme at CFL of 1 without the use of cell merging or redistribution. This is a significant achievement for embedded boundary methods, where small cell problems are typically problematic for explicit advection schemes.



(a) Solution ϕ of the advected Gaussian profile after one periodic cycle in the angled channel, which is visually indistinguishable from the initial condition.



(b) A close up view of the angled channel mesh and cell volume fraction κ.

Fig. 5 Advection of a Gaussian profile in an angled channel. The channel is periodic in the vertical direction, and imposes slip walls on the left and right boundaries.

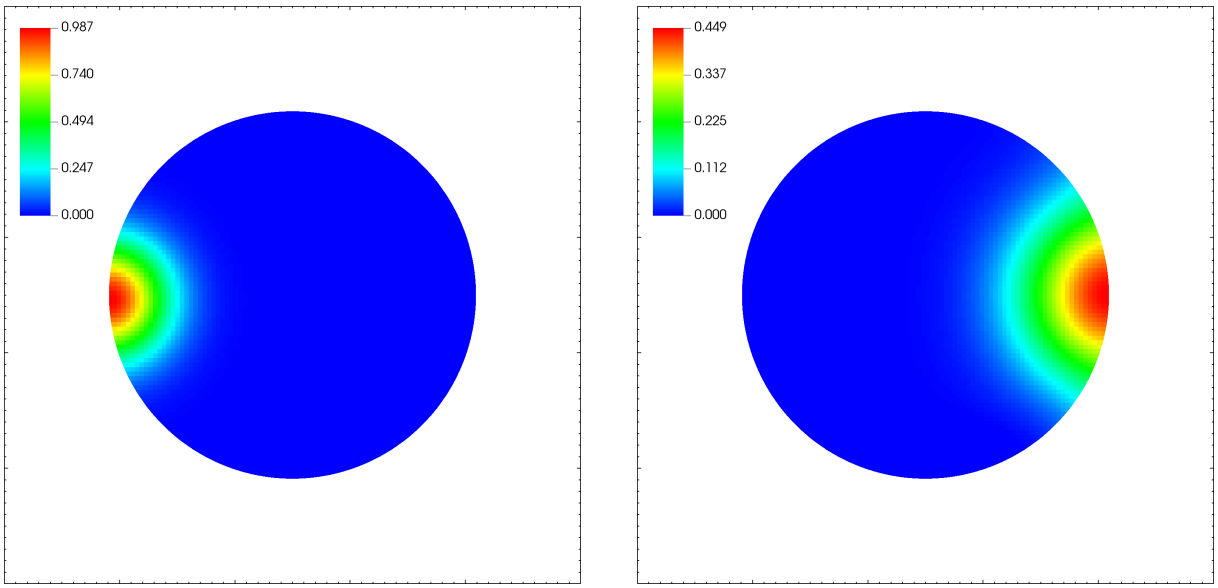
Table 3 Scalar advection errors and convergence rates

N	L_1	Rate(L_1)	L_2	Rate(L_2)	L_∞	Rate(L_∞)
64	8.13151344e-05		3.24451671e-04		3.90291704e-03	
128	2.91882209e-06	4.800	1.27778440e-05	4.666	2.58432941e-04	3.917
256	1.02664699e-07	4.829	4.94748718e-07	4.691	1.56887807e-05	4.042

D. Advection-Diffusion for Solid Body Rotation inside a Circle

Prior results in sections VI.B and VI.C verify scalar advection and diffusion separately. Approaching the incompressible Navier-Stokes equations, these two equations must be coupled and the resulting algorithm verified. The scalar advection-diffusion equation is given by $\frac{\partial}{\partial t}\phi = -\mathbf{A}(\phi) + \nu\mathbf{L}\phi$. Since few analytic solutions are available, the Richardson extrapolation method is used with a "best" solution to compare against. The algorithm targets fourth-order

accuracy in space and time using the high-order stencils described in section IV and the ImEx scheme in section III where the advection term is solved explicitly and the diffusion term implicitly. A circular geometry centered about $(0.5, 0.5)$ with radius of 0.3183 is specified, and zero valued Neumann boundary conditions are enforced. The initial profile is in defined by the Gaussian profile in Eq. (35), and shown in Fig. 6, is initialized along the left most boundary centered about the point $\mathbf{x}_0 = (0.1817, 0.5)$ with $a = 1$ and $\sigma^2 =$ 1e-2. The velocity field for the solid body rotation is specified as $u_1 = \omega(x_1 - 0.5)$ $u_2 = -\omega(x_0 - 0.5)$. The rotation rate ω and diffusion rate ν are both with unit values. At the finest case, the grid size is $h = 1/128$, and the time step is set corresponding to a CFL number of 1. Each coarser level has the grid size and time step doubled from the finest. The profile is advanced in time for 3.14159 time units, or half of a rotation by the velocity field. In Fig. 6 the initial and final solution profiles are shown. It is seen that the initial profile diffuses while it travels along the boundary. Using the Richardson interpolated solution at resolution $h = 1/256$ as a reference, the L_1, L_2, and L_∞ errors and convergence rates are calculated and compiled in Table 4. The expected fourth-order accuracy is shown for in all error norms, verifying the algorithm for scalar advection-diffusion. Conservation of the scalar with Neumann boundary conditions can be confirmed near round-off. As in the pure advection case, it is particularly noteworthy that this algorithm achieves a stable high-order semi-explicit scheme at a CFL of 1 without the use of cell merging or redistribution.



(a) Initial profile of ϕ. **(b) Profile of ϕ run for half a rotation.**

Fig. 6 Advection-diffusion of a Gaussian profile in a counter-clockwise rotating velocity field. The boundary conditions are specified as zero valued Neumann boundaries.

Table 4 Advection-Diffusion convergence rate

N	L_1	Rate(L_1)	L_2	Rate(L_2)	L_∞	Rate(L_∞)
32	1.41938306e-04		3.97170178e-04		2.57621273e-03	
64	5.11025648e-06	4.795	1.40567604e-05	4.820	9.69850826e-05	4.731
128	2.51271473e-07	4.346	7.09506437e-07	4.308	6.33940341e-06	3.935

VII. Conclusions and Future Work

In this work, we demonstrate that our embedded boundary algorithm is fourth-order accurate. Additionally, our algorithm is shown to be stable when encountering small cells, without cell merging, redistribution, or grid remediation to avoid small cells. These results demonstrate the feasibility of high-order embedded boundary methods, and provide confidence that the developed code can be extended to correctly solve engineering problems.

The immediate next step is to extend this algorithm for the incompressible Navier-Stokes equations. The primary challenge of this will be implementing and testing the non-linear advection term, and then coupling that with the existing diffusion and projection methods. Although this work only demonstrates two-dimensional results, the algorithm should be dimension independent and we plan to verify it next in three-dimensions. We also hope to extend this higher-order algorithm to the compressible Navier-Stokes equations, building on second-order methods previously published [3]. This introduces a number of additional challenges, such as computing more complicated non-linear upwind fluxes as well as introducing limiters into weighted least-squares stencils.

A. Acknowledgements

LBNL co-authors were supported by the Applied Mathematics Program of the U.S. DOE Office of Advanced Scientific Computing Research under contract number DE-AC02-05CH11231.

References

[1] Adams, M., Colella, P., Graves, D. T., Johnson, J. N., Johansen, H. S., Keen, N. D., Ligocki, T. J., Martin, D. F., McCorquodale, P. W., Modiano, D., Schwartz, P. O., Sternberg, T. D., and Van Straalen, B., "Chombo Software Package for AMR Applications—Design Document," Tech. Rep. LBNL-6616E, Lawrence Berkeley National Laboratory, 2015.

[2] Guzik, S. M., Gao, X., Owen, L. D., McCorquodale, P., and Colella, P., "A Freestream-Preserving Fourth-Order Finite-Volume Method in Mapped Coordinates with Adaptive-Mesh Refinement," *Comput. Fluids*, Vol. 123, 2015, pp. 202–217.

[3] Graves, D. T., Colella, P., Modiano, D., Johnson, J., Sjogreen, B., , and Gao, X., "A Cartesian Grid Embedded Boundary Method for the Compressible Navier Stokes Equations," *Comm. App. Math. Comp. Sci.*, Vol. 8, No. 1, 2013, pp. 99–122.

[4] Aftosmis, M., Berger, M., and Adomavicius, G., *A parallel multilevel method for adaptively refined Cartesian grids with embedded boundaries*, 2000. doi:10.2514/6.2000-808.

[5] Richards, K., Senecal, P., and Pomraning, E., "CONVERGE 3.0*," Convergent Science, Madison, WI, 2021.

[6] Berger, M. J., and Aftosmis, M. J., "Progress Towards a Cartesian Cut-Cell Method for Viscous Compressible Flow," *50th AIAA Aerospace Sciences Meeting*, AIAA, 2012. https://arc.aiaa.org/doi/10.2514/6.2012-1301.

[7] Trebotich, D., and Graves, D. T., "An adaptive finite volume method for the incompressible Navier-Stokes equations in complex geometries," *Comm. App. Math. and Comp. Sci.*, Vol. 10, No. 1, 2015, pp. 43–82.

[8] Devendran, D., Graves, D. T., Johansen, H., and Ligocki, T., "A fourth-order Cartesian grid embedded boundary method for Poisson's equation," *Comm. App. Math. and Comp. Sci.*, Vol. 12, No. 1, 2017, pp. 51–79.

[9] Chorin, A. J., "Numerical solution of the Navier-Stokes equations," *Mathematics of Computation*, Vol. 22, No. 104, 1968, pp. 745–745. doi:10.1090/s0025-5718-1968-0242392-2, URL https://doi.org/10.1090/s0025-5718-1968-0242392-2.

[10] Bell, J. B., Colella, P., and Glaz, H. M., "A second-order projection method for the incompressible navier-stokes equations," *Journal of Computational Physics*, Vol. 85, No. 2, 1989, pp. 257–283. doi:10.1016/0021-9991(89)90151-4, URL https://doi.org/10.1016/0021-9991(89)90151-4.

[11] Martin, D. F., Colella, P., and Graves, D., "A cell-centered adaptive projection method for the incompressible Navier–Stokes equations in three dimensions," *Journal of Computational Physics*, Vol. 227, No. 3, 2008, pp. 1863–1886. doi:https://doi.org/10.1016/j.jcp.2007.09.032.

[12] McCorquodale, P., and Colella, P., "A high-order finite-volume method for conservation laws on locally refined grids," *Comm. App. Math. Comput. Sci.*, Vol. 6, No. 1, 2011, pp. 1–25.

[13] Ascher, U. M., Ruuth, S. J., and Spiteri, R. J., "Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations," *Applied Numerical Mathematics*, Vol. 25, No. 2, 1997, pp. 151 – 167. Special Issue on Time Integration.

[14] Martin, D. F., and Cartwright, K. L., "Solving Poisson's equation using adaptive mesh refinement," Tech. Rep. UCB/ERI M96/66, University of California, Berkeley, 1996.

[15] Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., May, D. A., McInnes, L. C., Mills, R. T., Munson, T., Rupp, K., Sanan, P., Smith, B. F., Zampini, S., Zhang, H., and Zhang, H., "PETSc Users Manual," Tech. Rep. ANL-95/11 - Revision 3.9, Argonne National Laboratory, 2018.

[16] Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F., "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries," *Modern Software Tools in Scientific Computing*, edited by E. Arge, A. M. Bruaset, and H. P. Langtangen, Birkhäuser Press, 1997, pp. 163–202.

[17] Kennedy, C., and Carpenter, M., "Additive Runge-Kutta schemes for convection-diffusion-reaction equations," *Applied Numerical Mathematics*, Vol. 44, 2003, pp. 139–181.

[18] Zhang, Q., Johansen, H., and Colella, P., "A fourth-order accurate finite-volume method with structured adaptive mesh refinement for solving the advection-diffusion equation," *SIAM J. Sci. Comput.*, Vol. 34, 2012, pp. B179–B201.

[19] Salari, K., and Knupp, P., "Code Verification by the Method of Manufactured Solutions," Technical Report SAND2000-1444, Sandia National Laboratories, June 2000.