A FOURTH-ORDER EMBEDDED BOUNDARY FINITE VOLUME METHOD FOR THE UNSTEADY STOKES EQUATIONS WITH COMPLEX GEOMETRIES *

NATHANIEL OVERTON-KATZ†, XINFENG GAO‡, STEPHEN GUZIK§, OSCAR ANTEPARA¶, DANIEL T. GRAVES‖, AND HANS JOHANSEN#

Abstract. A fourth-order finite volume embedded boundary (EB) method is presented for the unsteady Stokes equations. The algorithm represents complex geometries on a Cartesian grid using EB, employing a technique to mitigate the "small cut-cell" problem without mesh modifications, cell merging, or state redistribution. Spatial discretizations are based on a weighted least-squares technique that has been extended to fourth-order operators and boundary conditions, including an approximate projection to enforce the divergence-free constraint. Solutions are advanced in time using a fourth-order additive implicit-explicit Runge-Kutta method, with the viscous and source terms treated implicitly and explicitly, respectively. Formal accuracy of the method is demonstrated with several grid convergence studies, and results are shown for an application with a complex bio-inspired material. The developed method achieves fourth-order accuracy and is stable despite the pervasive small cells arising from complex geometries.

Key words. High-Order Finite Volume, Embedded Boundary, Stokes Equations

MSC codes. 65M08, 76D07

Notation.

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

*Submitted to the editors November 1, 2022.

Note: This work builds upon prior conference proceedings [26] from the authors.

Funding: This work was supported by the Applied Mathematics Program of the U.S. DOE Office of Advanced Scientific Computing Research under contract number DE-AC02-05CH11231. Some computations used the resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

†Lawrence Berkeley National Lab, Berkeley, CA, USA(novertonkatz@lbl.gov)

‡Colorado State University, Fort Collins, CO, USA(Xinfeng.Gao@colostate.edu)

§Colorado State University, Fort Collins, CO, USA(Stephen.Guzik@colostate.edu)

¶Lawrence Berkeley National Lab, Berkeley, CA, USA(oantepara@lbl.gov)

‖Lawrence Berkeley National Lab, Berkeley, CA, USA(dtgraves@lbl.gov)

#Lawrence Berkeley National Lab, Berkeley, CA, USA(hjohansen@lbl.gov)

1. Introduction. Finite volume methods (FVMs) can achieve high solution accuracy and efficiency on structured Cartesian grids with a high degree of parallelism [1]. High-order methods have successfully been created for Cartesian grids [23] using polynomial reconstructions from structured data. Despite these advantages, representation of complex geometries on structured grids is a significant challenge. Technologies such as mapped multi-block methods [24, 13] can represent moderately complex geometries using structured grids by combining several blocks of curvilinear grids. However, mapped multi-block grids struggle to represent rough surfaces, and creating quality grids is often time-consuming [31].

Alternatively, the embedded boundary (EB) method [2] can represent complicated geometries with little restriction, while maintaining the advantages of structured grid solvers. EB grids are created by embedding the boundary geometry in a Cartesian grid, and cells that intersect the boundaries are cut, as illustrated by Fig. 1.1. The resulting grid is one that is regular away from the boundary, while near the boundary the grid is made up of partial, or "cut" cells. This approach 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 in parallel. However, the presence of cut-cells introduces challenges for both higher-order accuracy and numerical stability. Specific stencil construction schemes are required near the boundaries, since regular grid-aligned stencils may not be consistent or stable if they use cut-cell values. In addition, volumes of cut-cells may also become arbitrarily small, and maintaining stability of these small cells requires special care.

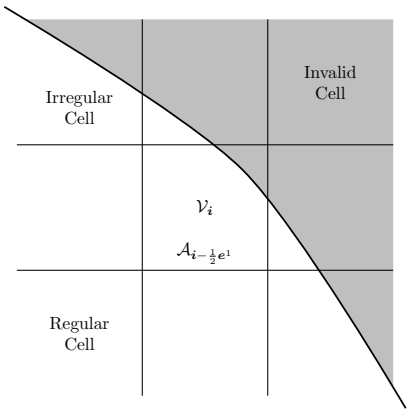


Fig. 1.1: Illustration of a grid for EB methods. The shaded region lies outside the problem domain of interest.

While the EB method has been used in a number of complex fluid dynamics applications [12, 2, 28, 8], typical applications have only achieved up to second-order accurate solutions, with first-order or inconsistent results near embedded boundaries [32]. There has been recent progress with improvements in accuracy [26]. For example a high-order finite volume EB method for smooth and kinked (C^0) domains was demonstrated for Poisson's equation by Devendran et al. [11] using the high-order grid generation method by Ligocki et al.[19]. The present work extends that fourth-order EB method to solve the time dependent Stokes equations. Furthermore, a fourth-order additive Runge-Kutta (ARK) scheme is applied to integrate viscous

terms implicitly, but any source terms explicitly. The implicit-explicit (ImEx) time integration is coupled with a projection method [9] to enforce the divergence-free constraint on the velocity.

Similar to EB methods are immersed boundary methods [27, 25, 18] which also cut a Cartesian grid at the boundaries and have been shown at high-order accuracy. However, immersed boundary methods build on finite-difference methods and generally do not conserve solution quantities, an inherent property of finite-volume methods. There are also finite-element based approaches which address cut-cells, such as the implicit mesh discontinuous Galerkin methods [30]. In these methods, the challenges of dealing with cut-cells, or elements, are addressed by formulating quadrature rules that account for the cut boundaries. In contrast, our method uses a weighted least-squares method with higher-order geometric information in order to perform local reconstructions. Most significantly, our approach does not require special small cell treatment, such as cell merging or flux redistribution, and displays good operator stability properties in a finite volume formulation.

This paper is organized as follows. Section 2 describes the FVM for the EB method, and details the approach taken to solve for the Stokes equations. Next, in Section 3, the algorithm for the fourth-order EB method is developed and applied specifically for the Stokes equations. The developed algorithm is then verified and validated in Section 5 for a number of test cases, and results for complicated geometries are presented and discussed in Section 6. Finally, we draw conclusions and propose directions for future work in Section 7

2. The Finite-Volume Method for Embedded Boundaries. The FVM discretizes time dependent partial differential equations (PDEs) in "flux-divergence" form,

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

where $\mathbf{u}$ is a solution vector that varies in space $\mathbf{x}$ and time t, $\vec{\mathbf{F}}$ is a flux dyad, and $\mathbf{s}$ is a source term that may depend on the solution $\mathbf{u}$. To apply the FVM, 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

$$\frac{\partial}{\partial t}\int_{\mathcal{V}_i}\mathbf{u}\,\mathrm{d}\mathbf{x} + \int_{\partial\mathcal{V}_i}\vec{\mathbf{F}}\cdot\hat{\mathbf{n}}\,\mathrm{d}\mathbf{x} = \int_{\mathcal{V}_i}\mathbf{s}\,\mathrm{d}\mathbf{x}\,. \tag{2.1}$$

The volume's surface $\partial\mathcal{V}_{\boldsymbol{i}}$ is split into separate discrete regions $\mathcal{A}_{\boldsymbol{f}}$, indexed with $\boldsymbol{f}$, so that $\partial\mathcal{V}_{\boldsymbol{i}} \equiv \cup_{\boldsymbol{f}}\mathcal{A}_{\boldsymbol{f}}$. The surface flux integral then becomes:

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

where $\hat{\mathbf{n}}_{\boldsymbol{f}}$ is the corresponding outward unit normal vector.

FVMs then define averaged quantities on faces and volumes,

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

and express Eq. (2.1) in terms of averages in the discrete ODE form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle\mathbf{u}\rangle_{\boldsymbol{i}} + \sum_{\boldsymbol{f}\in\partial\mathcal{V}_i}\frac{\mathcal{A}_{\boldsymbol{f}}}{\mathcal{V}_{\boldsymbol{i}}}\langle\mathbf{F}\rangle_{\boldsymbol{f}} = \langle\mathbf{s}\rangle_{\boldsymbol{i}}\,. \tag{2.2}$$

Note that Eq. (2.2) has a term of $\mathcal{V}_{\boldsymbol{i}}^{-1}$ that must be carefully balanced to avoid numerical stability issues. As a measure of how small cells are, we define the volume fraction κ so that $\kappa h^{\mathrm{D}} = \mathcal{V}_{\boldsymbol{i}}$, where h is the Cartesian grid spacing. Because cut-cells can be arbitrarily small in our discretization, Eq. (2.2) must be well-defined in terms of how $\mathcal{A}_{\boldsymbol{f}}$ and $\langle \mathbf{F} \rangle_{\boldsymbol{f}}$ are evaluated on a given small cell as $\mathcal{V}_{\boldsymbol{i}}$ (and thus κ) approach zero. Regardless, this remains an exact formulation with no approximations, providing the averages and geometric quantities are exact. In practice, the fluxes and time derivatives require numerical approximations, which characterize the accuracy and stability of the method.

2.1. Projection Form of the Unsteady Stokes Equations. The unsteady Stokes equations with constant density are given by

$$\frac{\partial}{\partial t}\mathbf{u} = -\nabla p + \nu \Delta \mathbf{u}\,, \tag{2.3}$$

$$\nabla \cdot \mathbf{u} = 0\,, \tag{2.4}$$

where $\mathbf{u}$ is the flow velocity, p the pressure, and ν is the constant kinematic viscosity. Boundary conditions for inflows prescribe a velocity $\mathbf{u} = \mathbf{u}_{\text{in}}$ and pressure $\nabla p \cdot \hat{\mathbf{n}} = 0$, while outflows are specified by $\nabla \mathbf{u} \cdot \hat{\mathbf{n}} = 0$ and pressure $p = 0$ or constant reference pressure. Viscous boundary conditions prescribe a boundary velocity $\mathbf{u} = \mathbf{u}_{\text{wall}}$ and pressure $\nabla p \cdot \hat{\mathbf{n}} = 0$.

A Hodge projection operator $\mathbb{P}$ has been used in the finite volume literature [9, 7] to enforce the divergence-free velocity field constraint. We can define this acting on any vector field $\mathbf{w}$ that is not divergence free:

$$\mathbb{P}(\mathbf{w}) = \mathbf{v} \tag{2.5}$$

$$\nabla \cdot \mathbf{v} = 0 \tag{2.6}$$

$$\mathbb{P}(\mathbf{w}) \equiv \left(\mathbb{I} - \nabla \Delta^{-1} \nabla \cdot\right) \mathbf{w}\,, \tag{2.7}$$

where $\mathbf{v}$ is the divergence free component of $\mathbf{w}$, with appropriate boundary conditions. This allows us to write (2.3) as $\frac{\partial}{\partial t}\mathbf{u} = \mathbb{P}(\nu \Delta \mathbf{u})$, with the pressure term eliminated [16, 7]. Thus, the projection enforces the pressure constraint and boundary conditions; we discuss this further in Section 2.2.1.

2.2. Finite Volume Projection Formulation. We choose discretizations for each of the spatial operators in Eqs. (2.3-2.4), and write the resulting discrete equations at cell location $\boldsymbol{i}$ as

$$\frac{\partial}{\partial t}\mathbf{u}_{\boldsymbol{i}} = -(\mathbf{G}p)_{\boldsymbol{i}} + \nu(\mathbf{L}\mathbf{u})_{\boldsymbol{i}}\,, \tag{2.8}$$

$$(\mathbf{D}\mathbf{u})_{\boldsymbol{i}} = 0\,, \tag{2.9}$$

where $\mathbf{D}, \mathbf{G}$, and $\mathbf{L}$ are fourth-order finite volume approximations of divergence, gradient, and Laplacian terms, respectively (note that from here on, we will drop the subscript $\boldsymbol{i}$ for simplicity). The goal is to discretize these operators so that

$$\mathbf{D}\mathbf{u} = \nabla \cdot \mathbf{u} + O(h^4)\,, \tag{2.10}$$

$$\mathbf{G}\mathbf{u} = \nabla \mathbf{u} + O(h^4)\,, \tag{2.11}$$

$$\mathbf{L}\mathbf{u} = \nabla \cdot \nabla \mathbf{u} + O(h^4)\,, \tag{2.12}$$

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

We use co-located cell-average velocity and pressure, for which a traditional marker-and-cell ("MAC") staggered-grid discretization of the projection is not available, so we instead use an *approximate* projection [21]. This implies that 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

$$\mathbf{P}(\mathbf{w}) = \left(\mathbb{I} - \mathbf{G}\mathbf{L}^{-1}\mathbf{D}\right)\mathbf{w}\,, \tag{2.13}$$

which requires the inversion of the Laplacian operator over the entire domain. Using this projection operator, our approximation of the unsteady Stokes equations is

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{u} = \mathbf{P}\left(\nu\mathbf{L}\mathbf{u}\right) + O(h^4) \tag{2.14}$$

$$\mathbf{D}\mathbf{u} = O(h^4)\,, \tag{2.15}$$

and we have eliminated the pressure from the time evolution equations. The procedure for advancing this system in time is to first do an intermediate update for the viscous terms, and then apply the projection Eq. (2.13), which produces an approximately-divergence free solution at the end of each time step. This is essentially a higher-order accurate version of the projection operator described by Trebotich et al [32].

2.2.1. Projection Boundary Conditions. Boundary conditions for the projection operator can be specified for for any vector field, so that the Hodge decomposition has three distinct components [10]: $\mathbf{w} = \mathbf{G}\psi + \mathbf{v} + \mathbf{G}\phi$. First, ψ is the scalar potential flow solution which satisfies only the boundary conditions in the normal direction ($\nabla\psi\cdot\hat{\mathbf{n}} = \mathbf{u}\cdot\hat{\mathbf{n}}$ for inflow and walls) and is divergence-free inside the domain. The two other parts satisfy homogeneous boundary for inflow and wall conditions: ϕ such that $\mathbf{G}\phi\cdot\hat{\mathbf{n}} = 0$, and the divergence-free part, $\mathbf{v}$, such that $\mathbf{D}\mathbf{v} = 0$ and $\mathbf{v}\cdot\hat{\mathbf{n}} = 0$. Each of these components is determined from $\mathbf{w}$ by solving the equations:

$$\mathbf{L}\psi = 0, \quad \mathbf{G}\psi\cdot\hat{\mathbf{n}} = \mathbf{u}\cdot\hat{\mathbf{n}} \qquad \text{(potential flow with BCs)}, \tag{2.16}$$

$$\mathbf{L}\phi = \mathbf{D}\mathbf{w}, \quad \mathbf{G}\phi\cdot\hat{\mathbf{n}} = 0 \qquad \text{(interior gradient)}, \tag{2.17}$$

$$\mathbf{u} = \mathbf{v} + \mathbf{G}\psi = \mathbf{w} - \mathbf{G}\phi \qquad \text{(divergence-free velocity)}. \tag{2.18}$$

This allows $\mathbf{u}$ to be approximately divergence-free and satisfy the correct domain boundary conditions. Although the Hodge decomposition has three components, the projection $\mathbf{u} = \mathbf{P}(\mathbf{w})$ only requires solving for ϕ.

Some modifications to this procedure are required for outflow [32], given that flow back into the domain may occur. For the divergence operator on the right-hand side of Eq. (2.17) the boundary conditions are: $\mathbf{w}$ matches the velocity at inflow $\mathbf{w} = \mathbf{u}_{\mathrm{in}}$, solid walls require no normal-flow $\mathbf{w}\cdot\hat{\mathbf{n}} = 0$, and outflow boundaries specify no boundary condition. Eq. (2.16) and Eq. (2.17) use outflow boundary conditions $\psi = 0$ and $\phi = 0$, respectively. The boundary condition of ϕ correspond to the pressure conditions for inflow and walls, since the projection takes the place of the pressure gradient [16].

3. Embedded Boundary Spatial Discretization. In the embedded boundary approach, there are three categories of cells: regular, irregular, and invalid. This distinction between cell types is illustrated in Fig. 1.1. Regular cells are those that are full Cartesian cells, and do not contain a portion of the boundary. Irregular 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 EB regions, we intersect the irregular domain, Ω, and $\Upsilon_{\boldsymbol{i}}$, any regular cell, to denote a particular cell by $\mathcal{V}_{\boldsymbol{i}} = \Upsilon_{\boldsymbol{i}} \cap \Omega$.

The challenge of EB 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. A general reconstruction evaluates local polynomials and their derivatives on faces to calculate the face-average fluxes in Eq. (2.2). For a structured grid, reconstructed polynomials lead to grid-aligned regular stencils. When using an EB method, stencils near the boundary depend on the local geometry. In those cases, we use a weighted least-squares polynomial approximation in a local region of neighboring cells. Theoretically, this can produce any order spatial discretization (see Devendran et al.[11]), but demonstrating a fourth-order method is the focus of this paper. In the following section, the necessary operators needed for the fourth-order EB method are described.

3.1. Multi-Dimensional Taylor Expansion. Stencils for the high-order EB method are produced from a multi-dimensional polynomial defined as

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

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

$$\phi(\mathbf{x}) = \sum_{|\mathbf{q}|<\mathrm{Q}} \frac{1}{\mathbf{q}!} \phi^{(\mathbf{q})}(\bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^{\mathbf{q}} + O(h^{\mathrm{Q}})\,, \tag{3.1}$$

where $\phi^{(\mathbf{q})}$ 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. To fit a multi-dimensional polynomial to cell-averaged data, integration over (regular or irregular) cells is needed. Using moments to define integration of basis polynomials over a region, we define volume moments and face moments as

$$m_{\boldsymbol{i}}^{\mathbf{q}}(\bar{\mathbf{x}}) = \int_{\mathcal{V}_{\boldsymbol{i}}} (\mathbf{x} - \bar{\mathbf{x}})^{\mathbf{q}}\, \mathrm{d}\mathbf{x}\,, \qquad m_{\boldsymbol{f}}^{\mathbf{q}}(\bar{\mathbf{x}}) = \int_{\mathcal{A}_{\boldsymbol{f}}} (\mathbf{x} - \bar{\mathbf{x}})^{\mathbf{q}}\, \mathrm{d}\mathbf{x}\,. \tag{3.2}$$

In this work, moments are computed with sufficient accuracy from a boundary geometry represented by implicit functions, as in [11]. This is achieved from application of the divergence theorem and Taylor expansions as detailed by Ligocki et al. [19]. Implicit function representations are general enough that they can accommodate geometries that may come from analytic descriptions, constructive solid geometry, or interpolated 3D imaging data.

3.2. Flux Reconstruction. To reconstruct a high-order solution from cell-averaged data, we use a Taylor expansion about cell centers $\bar{\mathbf{x}}_{\boldsymbol{i}}$ of each uncut-cell

$\Upsilon_{\boldsymbol{i}}$. For velocity component u_d, we can define multi-dimensional polynomial coefficients, $c_{d\boldsymbol{i}}^{\mathbf{q}} = \frac{1}{\mathbf{q}!}\mathrm{u}_d^{(\mathbf{q})}(\bar{\mathbf{x}}_{\boldsymbol{i}})$, and approximate each neighboring $\boldsymbol{j}$ cell-average as

$$\begin{aligned}\langle \mathrm{u}_d \rangle_{\boldsymbol{j}} &= \frac{1}{\mathcal{V}_{\boldsymbol{j}}} \int_{\mathcal{V}_j} \mathrm{u}_d \, d\mathbf{x} \\ &= \frac{1}{\mathcal{V}_{\boldsymbol{j}}} \int_{\mathcal{V}_j} \sum_{|\mathbf{q}|<\mathrm{Q}} \frac{1}{\mathbf{q}!} \mathrm{u}_d^{(\mathbf{q})}(\bar{\mathbf{x}}_{\boldsymbol{i}})(\mathbf{x}-\bar{\mathbf{x}}_{\boldsymbol{i}})^{\mathbf{q}} + O(h^{\mathrm{Q}}) \, d\mathbf{x} \\ &= \frac{1}{\mathcal{V}_{\boldsymbol{j}}} \sum_{|\mathbf{q}|<\mathrm{Q}} c_{d\boldsymbol{i}}^{\mathbf{q}} m_{\boldsymbol{j}}^{\mathbf{q}}(\bar{\mathbf{x}}_{\boldsymbol{i}}) + O(h^{\mathrm{Q}}) \, . \end{aligned} \tag{3.3}$$

The number of coefficients, $c_{d\boldsymbol{i}}^{\mathbf{q}}$, is $N_p = \frac{(\mathrm{D}+\mathrm{Q})!}{\mathrm{D}!\,\mathrm{Q}!}$. Determining these N_p coefficients requires solving a linear system of equations with at least that many linearly independent values $\langle \mathrm{u}_d \rangle_{\boldsymbol{j}}$. We select more cells than coefficients, to establish an *over-determined* least-squares system, which provides some robustness when there are small cells present in the interpolation neighborhood.

Adapting linear algebra notation, we write the vector u of neighboring cell-average velocities $\langle \mathrm{u}_d \rangle_{\boldsymbol{j}}$, so our approximation is

$$u \approx M \, c \, ,$$

where the geometric moment matrix M comes from Eq. (3.3) for each $\boldsymbol{j}$, and the vector c contains the multi-dimensional polynomial coefficients $c_{d\boldsymbol{i}}^{\mathbf{q}}$.

If we add a diagonal weighting matrix W, making a *weighted least-squares* (WLS) algorithm, we may improve stability of the stencils (as in [11]); further discussion is in Section 3.3. With the weighting matrix, this yields the system to determine coefficients as

$$c = \arg\min_{\tilde{c}} \left\| W u - W M \tilde{c} \right\|_2 \rightarrow c = (W \, M)^{\dagger} \, W \, u \, , \tag{3.4}$$

where $(W \, M)^{\dagger}$ indicates the pseudo-inverse of $(W \, M)$.

3.2.1. Higher-order EB Viscous Flux Stencil. We can use the WLS approach to calculate face-average fluxes for Eq. (2.2) from cell-average quantities (as in [11]). This is accomplished by determining the stencil $s_{\boldsymbol{f}}$ applied to neighbor values u, derived from the coefficient vector c:

$$\begin{aligned}\mathcal{A}_{\boldsymbol{f}} \langle \mathrm{F}_d \rangle_{\boldsymbol{f}} &\equiv s_{\boldsymbol{f}}^{\intercal} \, u \\ &= b^{\intercal} \, c \\ &= \left(b^{\intercal} \, (W \, M)^{\dagger} \, W \right) u \, , \end{aligned}$$

where b is a vector that approximates any face-average flux (*or other quantity*) from the known coefficients. For a linear flux, such as the viscous term $\mathrm{F}_d = \nu \nabla \mathrm{u}_d$, we determine b (and thus $s_{\boldsymbol{f}}$) using a Taylor expansion from the face center $\bar{\mathbf{x}}_{\boldsymbol{f}}$:

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

Each $b_{\boldsymbol{f}}^{\mathbf{q}}$ can be expressed as a sum of *normal-weighted* face moments $m_{\boldsymbol{f},d}^{\mathbf{q}}$:

$$b_{\boldsymbol{f}}^{\mathbf{q}} = \sum_{d=1}^{\mathrm{D}} \int_{\mathcal{A}_{\boldsymbol{f}}} \frac{\partial}{\partial x_d} (\mathbf{x} - \bar{\mathbf{x}}_{\boldsymbol{f}})^{\mathbf{q}} \, \hat{\mathbf{n}}_{\boldsymbol{f},d} \, \mathrm{d}\mathbf{x} = \sum_{d=1}^{\mathrm{D}} \mathrm{q}_d m_{\boldsymbol{f},d}^{\mathbf{q}-\boldsymbol{e}^d} , \tag{3.5}$$

where $\boldsymbol{q} - \boldsymbol{e}^d$ is required to be positive (derivatives of constants are zero).

For grid-aligned cell faces, normals $\boldsymbol{e}^d$ are constant, so the normal-weighted moments are simply the face moments. However for curved embedded boundaries, all the components of the normal play a role in the interpolation, especially in the case of inhomogeneous boundary conditions. Ultimately, the viscous flux stencil $s_{\boldsymbol{f}}$ depends only on the local neighbor volume and boundary moments, so it may be initialized once per geometry and stored as a sparse matrix operator over the (much smaller) subset of irregular cells.

3.2.2. Approximate Projection. The higher-order projection operator starts with cell-average velocities, $\langle \mathbf{u} \rangle_{\boldsymbol{i}}$, and modifies them to be *approximately* divergence-free (Eq. (2.14)). To accomplish this, we require discretizations and boundary conditions for each operator in Eq. (2.13). First, the Laplacian operator $\mathbf{L}$ is essentially the same as the viscous flux in Section 3.2.1, as a divergence of face-averaged quantities, but with different boundary conditions based on Eq. (2.16). For the cell-average gradient operator, $\mathbf{G}$, the WLS stencil is similar to Eq. (3.5). However, in this case we use cell moments instead of face moments to calculate a cell-average gradient:

$$G_{d,\boldsymbol{i}}^{\mathbf{q}} = \int_{\mathcal{V}_{\boldsymbol{i}}} \frac{\partial}{\partial x_d} (\mathbf{x} - \bar{\mathbf{x}}_{\boldsymbol{i}})^{\mathbf{q}} \, \mathrm{d}\mathbf{x} = \mathrm{q}_d m_{\boldsymbol{i}}^{\mathbf{q}-\boldsymbol{e}^d} . \tag{3.6}$$

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

Finally, we define a cell-average divergence operator $\mathbf{D}$ 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

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

The divergence flux $\langle \mathbf{u} \cdot \hat{\mathbf{n}} \rangle_{\boldsymbol{f}}$ can be constructed similarly to the Laplacian, with an operator that computes an average velocity flux U_f from each velocity component coefficient

$$U_{\boldsymbol{f}}^{\mathbf{q}} = \sum_{d=1}^{\mathrm{D}} \int_{\mathcal{A}_{\boldsymbol{f}}} (\mathbf{x} - \bar{\mathbf{x}}_{\boldsymbol{i}})^{\mathbf{q}} \hat{\mathbf{n}}_{\boldsymbol{f},d} \, \mathrm{d}\mathbf{x} = \sum_{d=1}^{\mathrm{D}} m_{\boldsymbol{f},d}^{\mathbf{q}} . \tag{3.7}$$

Again, the terms on regular faces have single component normal vectors $\hat{\mathbf{n}}_{\boldsymbol{f}} = \boldsymbol{e}^d$, and as a result $\langle \mathbf{u} \cdot \hat{\mathbf{n}} \rangle_{\boldsymbol{f}} = \langle \mathrm{u}_d \rangle_{\boldsymbol{f}}$. The boundary condition for the divergence is $\mathbf{u} \cdot \hat{\mathbf{n}} = 0$ at any solid wall, including EB boundaries. The velocity is specified at inflow boundaries, while outflow has no boundary conditions applied [10].

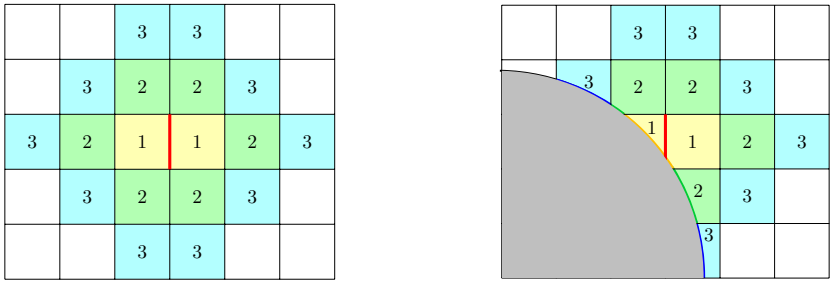
3.2.3. Physical Boundary Conditions. For Dirichlet boundary conditions, each face with a prescribed function uses a boundary average value $\langle \mathbf{u} \rangle_{\boldsymbol{f}} = \langle \mathbf{u}_{\mathrm{bc}}(\boldsymbol{x}) \rangle_{\boldsymbol{f}}$. A polynomial fitting the function about a face can be reconstructed using:

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

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.). This does require that the boundary conditions and derivatives are known at least to order Q.

3.2.4. Interpolation Neighborhoods. An important aspect of the WLS reconstruction is neighborhood selection. Conceptually, we require a large enough interpolation neighborhood to attain the desired accuracy and stability properties, while not making it so far-reaching so as to create large (expensive), ill-conditioned interpolation matrices. In this paper, the focus is on fourth-order accurate stencils, which in the general 2D case, requires a minimum of 10 neighbors (with at least Q in each direction) for all of the polynomial coefficients for $|\mathbf{q}| < Q = 4$. Higher-order derivative operations will require more neighbors to maintain the same accuracy, and values near an embedded boundary will require the boundary condition and further cells because of the lower accuracy of one-sided differences. For example, for a fourth-order WLS viscous operator (Laplacian), flux stencils of radius 3 cells from the reconstructed face can be used for a third-order accurate gradient. This is illustrated for a radius of 3 in Fig. 3.1(a) for regular regions, and Fig. 3.1(b) in the presence of an EB region with the inclusion of boundary conditions. However, for stencils that do not specify a boundary condition, such as the divergence on outflow boundaries, this radius of cells may need to be expanded to make the system sufficiently over-determined. Similarly, for stencils that evaluate cell-averaged values, such as the gradient for the projection operator, cell-average reconstructions are used. An example of a stencil of radius 3 centered about a cell is shown in Fig. 3.2(a) for regular cells and Fig. 3.2(b) for a cut-cell, following a similar pattern to reconstruction centered on faces. Cell-average reconstructions may include the cell where the reconstruction takes place, which is indicated as a radius of zero.

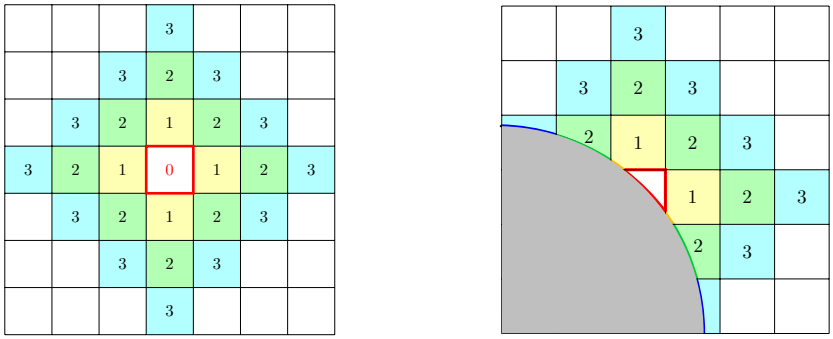
3.3. Weighting and Stability. To reconstruct the stencils required for the spatial discretization in the presence of EB geometries, the WLS method Eq. (3.4) is used. First, we select a neighborhood of cells within a given radius (including any cut-cells and their contained subset of the boundary). Rather than exhaustively searching for an interpolation neighborhood that can determine all the required coefficients with finite volume stencils, we use a larger number of neighbors and a weighting scheme that assigns relative importance to each cell's entry in the WLS system. Smaller relative weights mean that cells have less importance and a smaller coefficient in the resulting stencils. As in Devendran et al. [11], an effective weighting for a fourth-order



(a) An example regular flux stencil with 18 cells and no boundary conditions.

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

Fig. 3.1: Stencil neighborhoods for flux construction in 2D on the red highlighted faces, with cells numbered according to the Manhattan distance from the indicated red face. Boundary subregions included in the stencil are colored in (b).



(a) An example regular cell stencil with 25 cells and no boundary conditions.

(b) An example irregular cell stencil, using 16 cells and 7 boundary conditions.

Fig. 3.2: Stencil neighborhoods for cell value construction around the red highlighted cells, with cells numbered according to the Manhattan distance from the red cell. Boundaries included in the stencil are colored in (b).

interpolation is:

$$W_{\boldsymbol{i},\boldsymbol{f}} = \max(D_{\boldsymbol{i},\boldsymbol{f}}, 1)^{-5}, \tag{3.8}$$

where $D_{\boldsymbol{i},\boldsymbol{f}}$ is the Euclidean distance between cell $\boldsymbol{i}$ center and face $\boldsymbol{f}$ center. This weighting was shown to improve solution stability and spectral properties, especially when interpolation neighborhoods are large and there are many possible consistent stencils.

4. Implicit-Explicit Time Marching Method. When solving the Stokes equations, the maximum stable time step for the source and viscous terms can be significantly different. The source terms may generally be non-linear but less stiff, making them well suited for explicit time marching methods. In contrast, the viscous terms for the Stokes equations are stiff, but linear in each velocity variable, making

them well suited for implicit time integration using fast linear solvers. In the context of EB methods, where cut-cells can become arbitrarily small, the difference of time step stability constraints between terms can be orders of magnitude different. A hybrid implicit-explicit (ImEx) Runge-Kutta (RK) method [4, 33] is chosen for the EB algorithm, which allows for the source terms to be updated using an explicit method, and the viscous terms with an implicit method. This method has successfully been demonstrated for stiff, higher-order finite volume methods for advection-diffusion problems [33]. To achieve fourth-order accuracy, the six-stage, L-stable ImEx scheme ARK4(3)6L[2]SA [17] is used. At each stage, the implicit time integrator is applied to the viscous term discretization for each velocity component, requiring a globally-coupled sparse matrix system to be solved. To do this efficiently, we use an algebraic multigrid (AMG) solver from PETSC [5, 6]. After the final stage update from the time integrator, we apply the approximate projection operator (2.13) to the predicted velocity to enforce (to the target order of accuracy) the divergence-free constraint (2.4).

5. Algorithm Verification. We use grid convergence to demonstrate the order of accuracy of the algorithm. The solution error is computed from cell-averages as

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

and convergence rates are evaluated using L_∞, L_2, and L_1 norms computed as

$$L_\infty = \max_{\boldsymbol{i} \in \Omega} |E_{\boldsymbol{i}}| \,, \qquad L_2 = \left(\frac{1}{N_c} \sum_{\boldsymbol{i} \in \Omega} E_{\boldsymbol{i}}^2 \right)^{\frac{1}{2}} , \qquad L_1 = \frac{1}{N_c} \sum_{\boldsymbol{i} \in \Omega} |E_{\boldsymbol{i}}| \,,$$

where N_c is the number of cells in the domain Ω. These solution errors are notably *not* weighted by the cell volumes, in order to properly represent the errors in small cells. When exact solutions are known, convergence rates are calculated as

$$\mathrm{Q}_{\text{obsv}} = \log_r \left(\frac{\| \phi_{h_i} - \phi_{\text{exact}} \|}{\| \phi_{h_{i+1}} - \phi_{\text{exact}} \|} \right) . \tag{5.1}$$

where ϕ_{h_i} represents the solution at refinement level h_i, and r is a constant refinement ratio so that $r h_{i+1} = h_i$.

The method of manufactured solutions can be used to generate artificial analytic solutions for verification [29]. For this method, a manufactured solution of sufficient complexity is chosen, then a source term and boundary conditions are derived which satisfy the governing equations. However, for the Stokes equations manufactured solutions should still obey the divergence free constraint, limiting the availability of analytical solutions without implementing discrete boundary integral methods.

Algorithm verification of cases without analytic solutions use a standard Richardson extrapolation method to evaluate the convergence by including an additionally refined solution. Convergence rates with this method [33] are measured by

$$\mathrm{Q}_{\text{obsv}} \approx \log_r \left(\frac{\| \phi_{h_i} - \phi_{h_{i+1}} \|}{\| \phi_{h_{i+1}} - \phi_{h_{i+2}} \|} \right) . \tag{5.2}$$

We verify the projection and viscous operators separately on simple embedded boundary geometries. The projection operator was verified for correctness and stability using the Taylor Green vortex. The viscous operator with boundary conditions was verified using manufactured solutions in space and time. The order of accuracy for solving the Stokes equations was verified on a Couette flow between concentric circles.

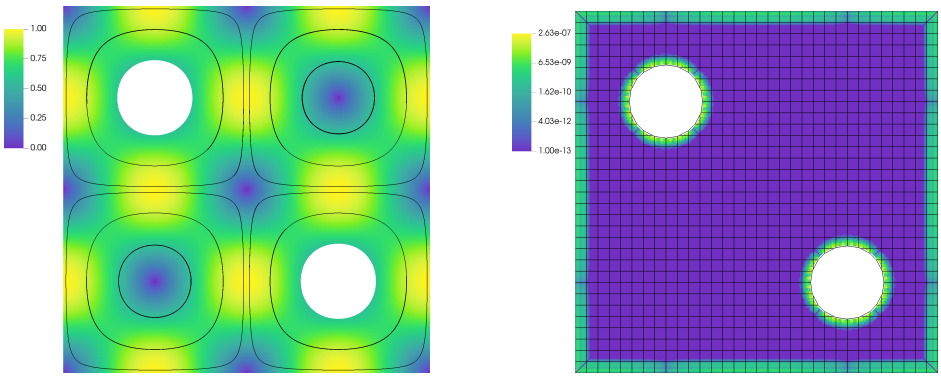
5.1. Projection of the Taylor-Green Vortex. The approximate projection operator $\mathbf{P}(\mathbf{u})$ is demonstrated to be both fourth-order accurate and stable by solving the classic Taylor-Green vortex [9] on a unit domain. The stream function is defined by

$$\psi = \sin(n\pi x)\sin(n\pi y)\,. \tag{5.3}$$

and the corresponding velocity field in Cartesian coordinates is,

$$u = \sin(n\pi x)\cos(n\pi y)\,, \quad v = \cos(n\pi x)\sin(n\pi y)\,, \tag{5.4}$$

which is analytically divergence free. For this case, we choose the period $n = 2$, and as a result there is no flow through the domain boundary. In addition, EB boundaries are cut along the contours of the stream function at $\psi = -0.8$. Initial conditions use the exact velocity profiles, integrated to fourth-order cell-averages, while projection boundary conditions are specified (no-flow, but not viscous, walls). The geometry and velocity field are shown in Fig. 5.1(a).



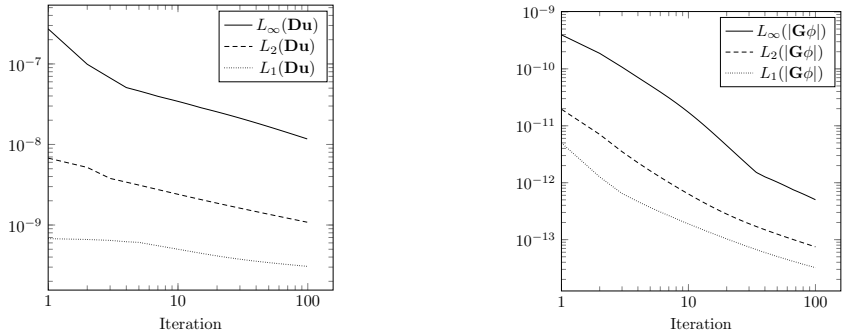
(a) The Taylor-Green velocity magnitude $|\mathbf{u}|$ and streamlines.

(b) Log plot of the divergence magnitude $\mathbf{Du}$ and a coarsened grid representation.

Fig. 5.1: The Taylor-Green vortex at $h = 1/256$.

Stability of the approximate projection operator is demonstrated by showing that repeated applications of the projection on a velocity field reduce the discrete divergence monotonically towards zero. The velocity divergence $\mathbf{Du}$ and pressure gradient $\mathbf{G}\phi$ are computed on a grid with cell size $h = 1/256$ for 100 projection applications and plotted in Fig. 5.2. These quantities strictly decrease, showing stability of the fourth-order approximate projection.

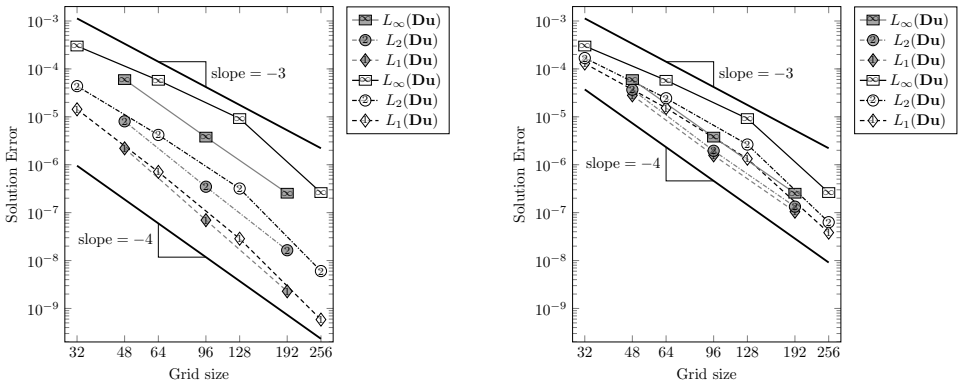
Convergence tests are performed to ensure the targeted fourth-order accuracy is achieved. The projection is applied once for grids of decreasing refinement, and the divergence field $\mathbf{Du}$ is used to evaluate the solution error. The finest levels are chosen with cell sizes $h = 1/256$ and $h = 1/192$, and subsequent coarser levels each double h. The L_1, L_2, and L_∞ errors are calculated and plotted in Fig. 5.3(a), where convergence rates reach and exceed expected fourth-order accuracy. Divergence errors are the largest in cut-cells at the embedded boundaries, as seen in Fig. 5.1(b), and dominate the L_∞ errors. Convergence rates of only the cut-cells are shown in Fig. 5.3(b) and observed somewhere between third- and fourth-order accurate. This is expected at



(a) Error norms of the velocity divergence. (b) Error norms of the pressure gradient.

Fig. 5.2: Solution error norms from repeated projections of the Taylor-Green vortex on grid size $h = 1/256$.

boundaries, where one-sided differences and geometry approximations can both affect accuracy. Nevertheless, the solution still retains fourth-order accuracy overall because the embedded boundary is codimension one smaller, and because of the smoothing effect of the elliptic operator, derivatives in cut-cells show "super-convergence" at similar rate [15].



(a) Error measures of the entire domain. (b) Error measures of only the cut-cells.

Fig. 5.3: Convergence rates of the divergence for the Taylor-Green vortex in 2D. The gray series of points are coarsened from a fine level of $h = 1/192$, and the white series coarsened from $h = 1/256$.

5.2. Manufactured Solution for Diffusion Inside a Circle. To demonstrate the accuracy of the viscous operator in the Stokes equations, we use a manufactured solution and solve without the projection operator. This completely decouples the velocity equations. The algorithm targets fourth-order in space and time using the high-order stencils described in Section 4 and the ImEx scheme in Section 3. A circular domain is created of radius 0.3 centered about the point $(0.5, 0.5)$. The manufactured

solution is the same for each component, defined by

$$\mathrm{u}_d(\mathbf{x},t) = \sin(2\pi t)\sin(R^2 - (\mathbf{x}-\mathbf{x}_0)^2) \tag{5.5}$$

where $R = 0.3$ matches the domain radius, and $\mathbf{x}_0 = (0.5, 0.5)$ gives 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 where

$$\begin{aligned}\mathrm{s}_d(\mathbf{x},t) =& 2\pi\cos(2\pi t)\sin(R^2-(\mathbf{x}-\mathbf{x}_0)^2)\\ &-\sin(2\pi t)\sum_d^{\mathrm{D}}\left(-4\mathrm{x}_d\sin(R^2-(\mathbf{x}-\mathbf{x}_0)^2)+2\cos(R^2-(\mathbf{x}-\mathbf{x}_0)^2)\right).\end{aligned}$$

The source term is evaluated explicitly in time, while the Laplacian term is evaluated implicitly. The solution is initialized using fourth-order cell-averages at time 0.125, and a viscosity of $\nu = 1$. On the finest level, with 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 number of time steps to reach the same end time. Using the chosen exact solution in Eq. (5.5), the L_1, L_2, and L_∞ errors and convergence rates are calculated and compiled in Table 5.1. Third-order truncation error is anticipated at the embedded boundaries, and will dominate the L_∞ norm. However, the L_1 and L_2 norms still attain fourth-order accuracy because the embedded boundary is only codimension one [15]. Fourth-order accuracy is demonstrated in these error norms, verifying the algorithm.

Table 5.1: Viscous operator convergence errors and rates for diffusion inside a circle.

N	L_1	Rate(L_1)	L_2	Rate(L_2)	L_∞	Rate(L_∞)
16	3.676e-07		5.323e-07		1.271e-06	
32	1.421e-08	4.693	2.111e-08	4.656	6.810e-08	4.223
64	6.449e-10	4.463	9.529e-10	4.470	3.186e-09	4.418
128	3.688e-11	4.128	5.467e-11	4.123	2.195e-10	3.860

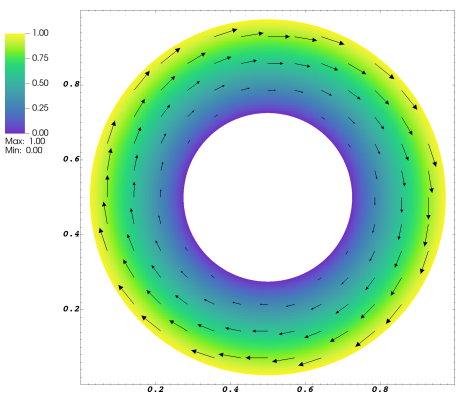
5.3. Spherical Couette Flow. A pair of concentric spheres are created with radii $r_{\text{inner}} = 0.225$ and $r_{\text{outer}} = 0.475$. The inner sphere is held stationary, while the outer sphere is rotated about the z-axis with constant angular velocity $\omega_{\text{outer}} = \frac{1}{0.475}$ so that the outer sphere has a peak tangential velocity of 1. Velocity of the rotating sphere is specified purely tangential to the surface by

$$u_{\tan} = \sin(\varphi)R\omega$$

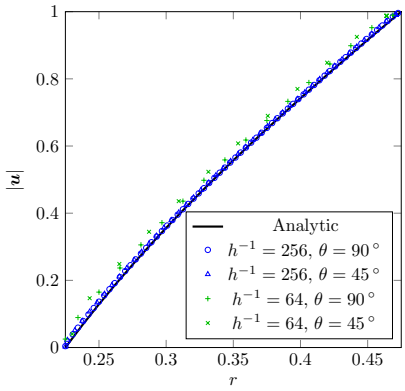
where R is the sphere radius, ω the angular velocity, and φ the polar angle from the z-axis. Converting this to Cartesian coordinates prescribes a velocity field

$$u = \omega y\,, \qquad v = -\omega x\,, \qquad w = 0\,,$$

where x and y are the Cartesian coordinates on the sphere centered at the origin. The solution is initialized to zero velocity uniformly with a viscosity of $\nu = 1$. Using the developed method in this work, the Stokes equations are solved to steady state.

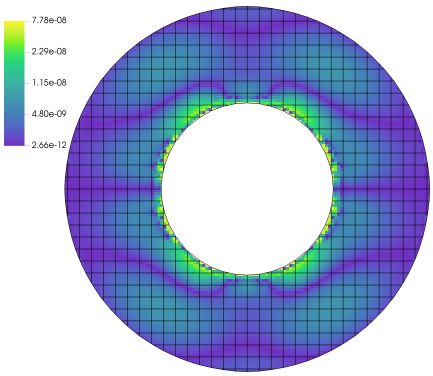


(a) Velocity magnitude and vectors for the $h = 1/256$ solution.

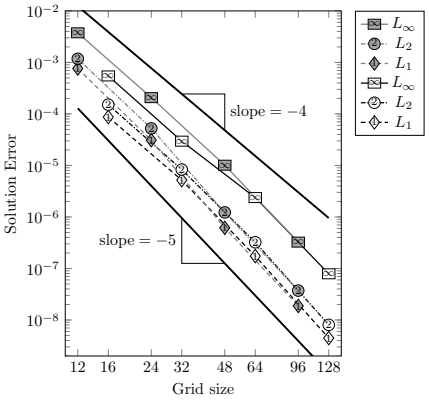


(b) Velocity magnitude profiles compared to the analytic solution.

Fig. 5.4: The two-dimensional circular Couette flow at steady state.



(a) Solution errors of the u velocity on the $h = 1/128$ grid, with a coarsened grid representation.



(b) Convergence rates of the u velocity. The gray series of points are coarsened from a fine level of $h = 1/192$, and the white series from $h = 1/256$.

Fig. 5.5: The two-dimensional circular Couette flow solution errors.

A two-dimensional case normal to the z-axis is shown in Fig. 5.4(a), and the three-dimensional version in Fig. 5.8.

On the finest level, of grid size $h = 1/256$, a time step of $\Delta t = 1 \times 10^{-3}$ is taken to advance the solution forward until the divergence norm converges in the two-dimensional case. To validate the solutions, radial profiles of the solution in two dimensions are compared to the analytic solution [22]

$$u_\theta(r) = \omega_{\text{outer}} r_{\text{outer}} \frac{r/r_{\text{inner}} - r_{\text{inner}}/r}{r_{\text{outer}}/r_{\text{inner}} - r_{\text{inner}}/r_{\text{outer}}},$$

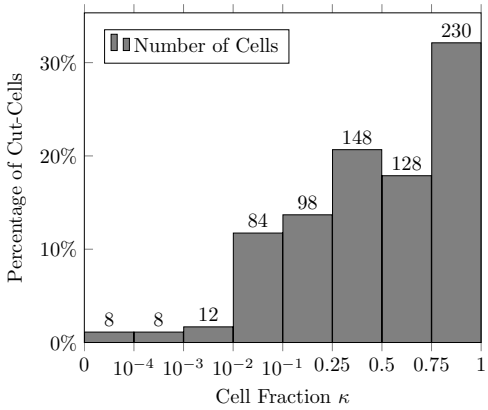


Fig. 5.6: Distribution of cut-cell sizes for the two-dimensional Couette flow at grid size $h = 1/256$.

in Fig. 5.4(b), where good agreement is observed. Convergence rates are measured and shown in Fig. 5.5(b) using the Richardson extrapolation method for a series of grids coarsened by a factor of 2 from refinements of both $h = 1/256$ and $h = 1/192$. Solution norms for the u and v velocity components only have differences on the order of machine precision, since the flow is symmetric, and so only one set of errors are shown. Results show that fourth-order accuracy is achieved or exceeded for all solution norms. The L_∞ norm in particular is dominated by cut-cell values, as shown in Fig. 5.5(a). On the finest grid there are 39 364 valid cells, of which 716 are cut-cells with volume fractions as small as $\kappa = 1.317 \times 10^{-5}$, demonstrating the robustness of the method. The distribution of cut-cell sizes is shown in Fig. 5.6.

A more rigorous analysis of stability is shown with the spectrum of the fourth-order Poisson operator $\mathbf{L}\phi$ in 2D using Neumann boundary conditions, in Fig. 5.7. Eigenvalues calculations use SLEPc [14], which are practical to compute for only modest 2D and very small 3D problems. Notably all eigenvalues have very small imaginary parts, and strictly negative real parts, indicating the operator is stable. In addition, there are no ill-conditioning effects of large eigenvalues from small cell volumes. For comparison, we have plotted the maximum eigenvalue for the standard second-order (five-point, [1 -2 1] in each direction) and fourth-order (nine-point star-shaped, with [-1 16 -30 16 -1]/12 in each direction), which bound the observed spectrum well.

For higher-order methods, one of the important considerations is their computational performance, especially in comparison to lower-order schemes. For the Couette flow case, we compare the run time and solution errors of the presented algorithm at second and fourth order. This is possible because the developed algorithm has the order-of-accuracy as an adjustable parameter, similar to finite-element methods. Run times of the algorithm on grids of size $h = 1/32$, $h = 1/64$, and $h = 1/128$ are listed in Table 5.2. The run times were split between one-time setup time and the time taken to evaluate each time step. The setup time includes the generation of the cut-cell grid, evaluation of all associated geometric moments, and creation of stencils. The step time is an averaged evaluation time of each time step, which includes multiple implicit solves and a projection. All timings are produced on an Intel® Core™ i7-12700H processor utilizing the 6 performance cores.

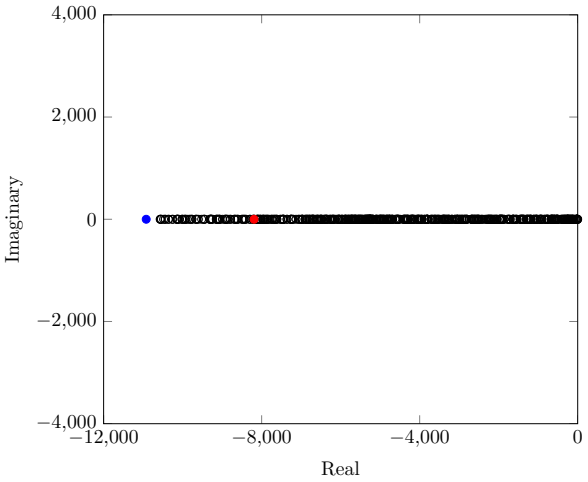


Fig. 5.7: Plot of the eigenvalues for the Poisson operator $\mathbf{L}\phi$ with Neumann boundary conditions on the embedded boundary domain used in the 2D Couette flow case at grid size $h = 1/32$. For comparison, the largest magnitude eigenvalue of standard regular stencils are shown for a second-order (red dot) and fourth-order (blue dot). Note that there are no ill-conditioning effects of large eigenvalues from small cells.

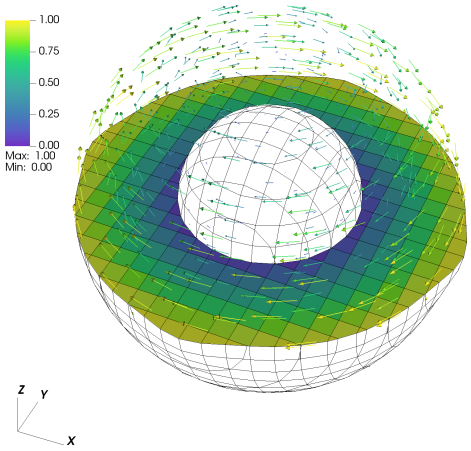


Fig. 5.8: The three-dimensional spherical Couette flow at steady state. The velocity vector field is shown, and a slice along the x-y plane shows the grid and velocity magnitude.

As shown in Table 5.2, we observe the fourth-order step time requires approximately 50% more time that the second-order method does for a given grid size. The setup time of the fourth-order method is approaching 2.5× as expensive as the second-order scheme, but is a one-time cost for stationary boundaries. We also observe that the setup time roughly grows with N, not N^2, because it is primarily a function of the number of cut cells, which is codimension one smaller. The more significant comparison is the time spent to reach a target level of error. If we examine the cost required

to reach approximately the same L_2 error, we see that the second-order scheme with a grid size of $h = 1/128$ and the fourth-order method with $h = 1/32$ are approximately the same. Comparing step times, 0.444 seconds vs. 0.145 seconds, the fourth-order method is only 33% the cost of the second-order method. This suggests that the higher-order algorithm is more efficient for smooth problems, such as the Couette flow case examined here. More performance and optimization analysis could be done, involving the geometric algorithms, matrix solve times, preconditioners, etc., but our focus here is on the development of this new algorithm.

Table 5.2: Run times of the 2D Couette flow case at both second and fourth order. Run times are reported in second.

	Second-Order			Fourth-Order		
N	setup time	step time	L_2 error	setup time	step time	L_2 error
32	1.029	0.099	9.866e-5	4.385	0.145	8.332e-6
64	3.078	0.227	1.909e-5	8.051	0.342	3.114e-7
128	5.102	0.444	5.093e-6	13.603	0.645	7.729e-9

6. Results. Using the verified and validated fourth-order EB algorithm, Stokes flow over a circle and sphere in a channel are tested, and convergence order is verified. The Stokes equations are also used to solve for flow through a complex bio-inspired material of engineering interest.

6.1. Steady Stokes Flow Over a Sphere in a Channel. A square channel is generated with a channel length of 2 in the x-direction, and an inlet length of 1 in the y and z directions. A sphere of radius $r = 0.15$ is centered in the channel at $x = 1$. A developed inflow of peak value 1 is specified at the inflow on the left most boundary, while an outflow is specified for the right most boundary. All other boundary conditions are specified as walls. The Stokes equations are solved to steady state in both two-dimensions, in Fig. 6.1, and three-dimensions, in Fig. 6.4. Although no analytic solution for this flow can be used for comparison, qualitatively the flow is observed to be highly symmetric and respect the imposed boundary conditions. Notably, the inflow boundary and outflow nearly match, and have streamlines normal to the boundaries as expected. At wall boundaries, solution velocities approach zero, although particularly along the sphere there is some discrepancy due to the projection and viscous operators smearing the third-order boundary solution with the fourth-order interior.

Convergence rates for the u and v velocity from Richardson extrapolation are plotted in Fig. 6.2. L_1 and L_2 norms indicate fourth-order accuracy, but the L_∞ norm lags slightly. The plot of solution error (Fig. 6.3) indicates lower accuracy in cut-cells and parts of the domain boundary.

6.2. Stokes Flow for Bone Scaffolding. Cellular structures have been a topic of interest for a range of bio-inspired materials because their geometries can be quickly adapted to target desired mechanical properties. In the work by Asbai-Ghoudan el at.[3] a structure for bone replacement is analyzed, and a key challenge identified when modeling such structures is mesh generation. We construct models of similar geometry using the EB method to significantly simplify the meshing process. The geometry of interest is defined by the gyroid function

$$f(\boldsymbol{x}) = \cos(n\pi x)\sin(n\pi y) + \cos(n\pi y)\sin(n\pi z) + \cos(n\pi z)\sin(n\pi x)\,, \tag{6.1}$$

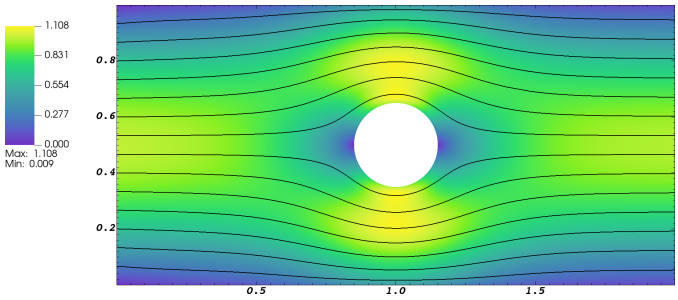


Fig. 6.1: Stokes flow for a circle in a channel, where the left boundary is an inlet, the right an outlet, and all other boundaries walls. Streamlines are shown in black, and the contours plot the velocity magnitude.

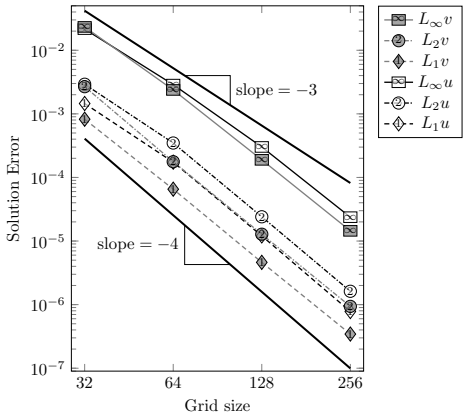


Fig. 6.2: Convergence rates of the 2D circle in a channel.

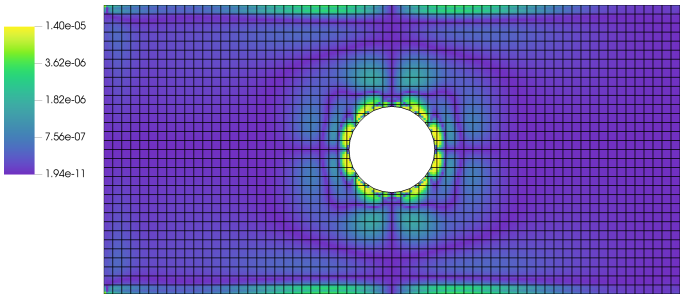


Fig. 6.3: Solution errors for the u velocity of the 2D Stokes flow over a circle in a channel on the $h = 1/265$ grid, with a coarsened grid representation.

evaluated in two dimensions along the $z = 0.1$ plane where $f(\boldsymbol{x}) = 0$ with a period of $n = 2$. The surface is approximately thickened to width of $1/6$. Flow is solved through a pipe of radius $r = 0.95$ and length 8, with a cut of the gyroid centered in the pipe at $x = 4$. A cut is made through the centerline with radius $r = 0.175$.

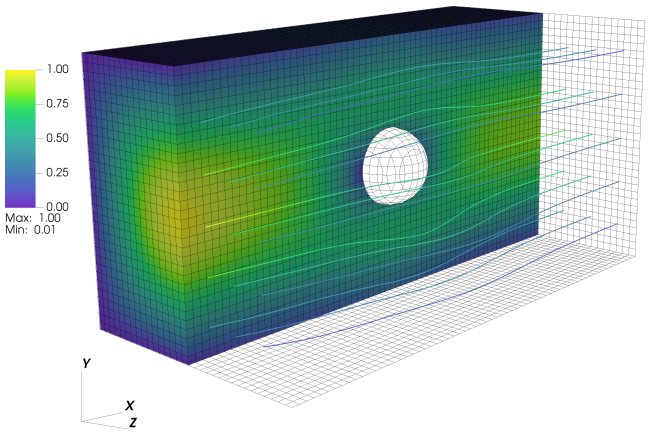


Fig. 6.4: Stokes flow over a sphere in a channel, where the left boundary is an inlet, the right an outlet, and all other boundaries walls. The streamlines and contour plot show the velocity magnitude.

The cut boundaries are smoothed (as in Devendran et al. [11]) to prevent singular solutions around sharp exterior corners. Stokes flow through this structure is shown in Fig. 6.5 for a grid refinement of $h = 1/512$ in the flow direction.

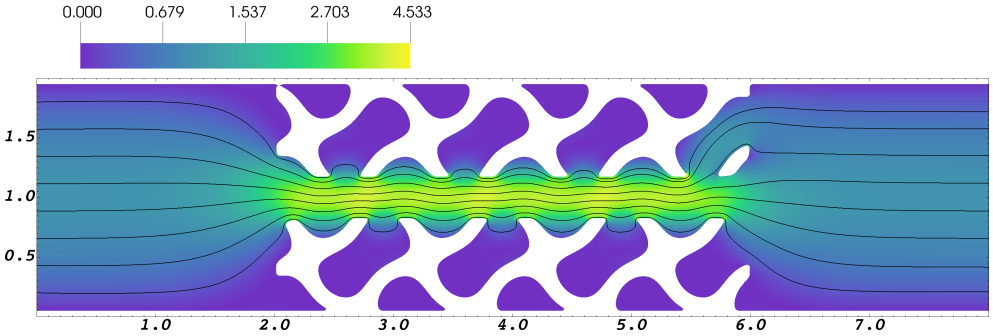
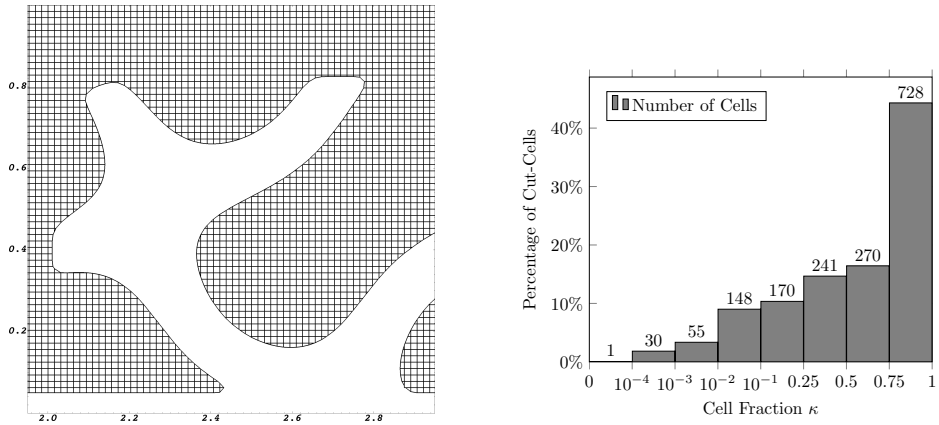


Fig. 6.5: Velocity magnitude and streamlines for a two-dimensional representation of the bone scaffold geometry.

The flow is found to be symmetric, and well-behaved along the cut-cell boundaries. The EB grid for this case contains 26 585 valid cells, of which 1643 are cut-cells with the smallest volume fraction of $\kappa = 5.933 \times 10^{-5}$. A close up view of this grid is shown in Fig. 6.6(a), and the distribution of cut-cell sizes is shown in Fig. 6.6(b). In particular, the mesh generation requires trivial involvement, and because the geometry is specified analytically, adjustments such as gyroid size, position, and thickness are easily made. This shows promise for more detailed analysis using the high-order EB method, where a large design space of geometries could be examined quickly without special considerations for mesh generation or solution stability.

7. Conclusions and Future Work. In this work, we demonstrate that our EB algorithm capable of representing complex geometries is fourth-order accurate for the



(a) A close up of the EB grid.

(b) Distribution of the cut-cell sizes.

Fig. 6.6: Bone scaffold grid.

Stokes equations. 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 EB methods, and provide confidence that the developed algorithm can be extended to solve engineering problems without making case specific considerations for mesh generation.

The natural next step for this work is to extend it to the *incompressible* Navier-Stokes equations. This requires an approach for proper treatment of the non-linear advection term, and the ability to create stable upwind-dissipative stencils for small cells. We will also extend this work to domains with moving boundaries and level set methods, where the automatic mesh generation of EB methods is significantly simpler and more robust. One of the challenges there is to implement boundary conditions that require coupling the velocity components and the boundary motion. Additional, a number of challenges involving conservation in space-time and updating solutions as grids change will need to be addressed. We will also implement a geometric multigrid solver [20, 33] to improve the performance and memory of our current PETSc-based AMG solver, which would be very expensive to set up every time the mesh changes. Additionally, future work with this high-order EB method will include adaptive mesh refinement to improve solution accuracy in regions of interest [33, 11]. Adaptive mesh refinement is relatively easy with EB methods due to the underlying grid being Cartesian, which is one of the motivating factors for using an EB approach rather than unstructured grids where refinement and mesh optimization can become global problems.

REFERENCES

[1] M. Adams, P. Colella, D. T. Graves, J. N. Johnson, H. S. Johansen, N. D. Keen, T. J. Ligocki, D. F. Martin, P. W. McCorquodale, D. Modiano, P. O. Schwartz, T. D. Sternberg, and B. Van Straalen. Chombo software package for AMR applications—design document. Technical Report LBNL-6616E, Lawrence Berkeley National Laboratory, 2015.

[2] M. Aftosmis, M. Berger, and G. Adomavicius. A parallel multilevel method for adaptively refined Cartesian grids with embedded boundaries. In *38th Aerospace Sciences Meeting*

and Exhibit, 2000.

[3] R. Asbai-Ghoudan, S. Ruiz de Galarreta, and N. Rodriguez-Florez. Analytical model for the prediction of permeability of triply periodic minimal surfaces. *Journal of the Mechanical Behavior of Biomedical Materials*, 124:104804, 2021.

[4] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25(2):151 – 167, 1997. Special Issue on Time Integration.

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

[6] S. Balay, W. D. Gropp, L. Curfman McInnes, and B. F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[7] J. B. Bell, P. Colella, and H. M. Glaz. A second-order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 85(2):257–283, December 1989.

[8] M. J. Berger and M. J. Aftosmis. Progress towards a Cartesian cut-cell method for viscous compressible flow. In *50th AIAA Aerospace Sciences Meeting*, number AIAA 2012-1301. AIAA, 2012.

[9] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comput.*, 22(104):745–745, 1968.

[10] P. Colella and D. P. Trebotich. Numerical simulation of incompressible viscous flow in deforming domains. *Proceedings of the National Academy of Sciences*, 96(10):5378–5381, 1999.

[11] D. Devendran, D. T. Graves, H. Johansen, and T. Ligocki. A fourth-order Cartesian grid embedded boundary method for Poisson's equation. *Comm. App. Math. and Comp. Sci.*, 12(1):51–79, 2017.

[12] D. T. Graves, P. Colella, D. Modiano, J. Johnson, B. Sjogreen, , and X. Gao. A Cartesian grid embedded boundary method for the compressible Navier Stokes equations. *Comm. App. Math. Comp. Sci.*, 8(1):99–122, 2013.

[13] S. M. Guzik, X. Gao, L. D. Owen, P. McCorquodale, and P. Colella. A freestream-preserving fourth-order finite-volume method in mapped coordinates with adaptive-mesh refinement. *Comput. Fluids*, 123:202–217, 2015.

[14] V. Hernandez, J. Roman, and V. Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Trans. Math. Software*, 31(3):351–362, 2005.

[15] H. Johansen and P. Colella. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *J. Comput. Phys.*, 147:60–85, 1998.

[16] S. Kadioglu, R. Klein, and M. Minion. A fourth-order auxiliary variable projection method for zero-mach number gas dynamics. *J. Comput. Phys.*, 227(3):2012–2043, 2008.

[17] C. Kennedy and M. Carpenter. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Appl Numer Math.*, 44:139–181, 2003.

[18] M. Ehsan Khalili, Martin Larsson, and Bernhard Müller. High-order ghost-point immersed boundary method for viscous compressible flows based on summation-by-parts operators. *Int. J. Numer. Meth. Fl.*, 89(7):256–282, 2019.

[19] T. Ligocki, P. Schwartz, J. Percelay, and P. Colella. Embedded boundary grid generation using the divergence theorem, implicit functions, and constructive solid geometry. *J. Phys.: Conf. Ser.*, 125, 2008.

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

[21] D. F. Martin, P. Colella, and D. Graves. A cell-centered adaptive projection method for the incompressible Navier–Stokes equations in three dimensions. *J. Comput. Phys.*, 227(3):1863–1886, 2008.

[22] K. Masatsuka. *I do like CFD, VOL.1, Second Edition*. Number v. 1. K. Masatsuka, 2013.

[23] P. McCorquodale and P. Colella. A high-order finite-volume method for conservation laws on locally refined grids. *Comm. App. Math. Comput. Sci.*, 6(1):1–25, 2011.

[24] P. McCorquodale, M. R. Dorr, J. A. F. Hittinger, and P. Colella. High-order finite-volume methods for hyperbolic conservation laws on mapped multiblock grids. *J. Comput. Phys.*, (288):181–195, May 2015.

[25] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[26] N. Overton-Katz, X. Gao, S. M. Guzik, O. Antepara, D. T. Graves, and H. Johansen. Towards a

high-order embedded boundary finite volume method for the incompressible Navier-Stokes equations with complex geometries. In *AIAA SCITECH 2022 Forum*, 2022.

[27] C. Peskin. Numerical analysis of blood flow in the heart. *J. Comput. Phys.*, 25(3):220–252, 1977.

[28] K.J. Richards, P.K. Senecal, and E. Pomraning. Converge 3.0*. Convergent Science, Madison, WI, 2021.

[29] K. Salari and P. Knupp. Code verification by the method of manufactured solutions. Technical Report SAND2000-1444, Sandia National Laboratories, June 2000.

[30] R. Saye. Implicit mesh discontinuous galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part i. *J. Comput. Phys.*, 344:647–682, 2017.

[31] J. Slotnick, A. Khodadoust, J. Alonso, D Darmofal, W Gropp, E. Lurie, and D Mavriplis. Cfd vision 2030 study: A path to revolutionary computational aerosciences. Technical Report 218178, NASA/CR, 2014.

[32] D. Trebotich and D. T. Graves. An adaptive finite volume method for the incompressible Navier-Stokes equations in complex geometries. *Comm. App. Math. and Comp. Sci.*, 10(1):43–82, 2015.

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