An Adaptive Cartesian Grid Embedded Boundary Method for the Incompressible Navier Stokes Equations in Complex Geometry

B. van Straalen, D. Trebotich, T. Ligocki, D.T. Graves, P. Colella, M.F. Barad

Lawrence Berkeley National Laboratory, Berkeley, CA

Abstract

We present a second-order accurate projection method to solve the incompressible Navier-Stokes equations on irregular domains in two and three dimensions. We use a finite-volume discretization obtained from intersecting the irregular domain boundary with a Cartesian grid. We address the small-cell stability problem associated with such methods by hybridizing a conservative discretization of the advective terms with a stable, nonconservative discretization at irregular control volumes, and redistributing the difference to nearby cells. Our projection is based upon a finite-volume discretization of Poisson's equation. We use a second-order, L^∞-stable algorithm to advance in time. Block structured local refinement is applied in space. The resulting method is second-order accurate in L^1 for smooth problems. We demonstrate the method on benchmark problems for flow past a cylinder in 2D and a sphere in 3D as well as flows in 3D geometries obtained from image data.

Key words: incompressible flow, embedded boundary methods, complex geometry, projection method

[1] Research supported at the Lawrence Berkeley National Laboratory by the U.S. Department of Energy: Director, Office of Science, Office of Advanced Scientific Computing, Mathematical, Information, and Computing Sciences Division under Contract DE-AC02-05CH11231.

1 Introduction

1.1 Governing Equations

We are solving the constant-density, incompressible Navier-Stokes equations for velocity $\vec{u}$, normalized pressure p with a kinematic viscosity of ν over a domain Ω. These equations are given by

$$\begin{aligned} \tfrac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} &= -\nabla p + \nu \Delta \vec{u} \\ \nabla \cdot \vec{u} &= 0 \\ \vec{u} \cdot \hat{n} &= f(\vec{x}) \text{ on } \partial\Omega \end{aligned} \tag{1}$$

We transform this constrained system of equations into an initial value problem by using the Hodge decomposition. Any vector field $\vec{w}$ on a bounded domain Λ where

$$\int_{\partial\Lambda} \vec{w} \cdot \hat{n} \; dA = 0 \tag{2}$$

can be uniquely decomposed into two orthogonal vector fields, one divergence-free and the other the gradient of a scalar.

$$\begin{aligned} \vec{w} &= \vec{w}_d + \nabla\phi \\ \nabla \cdot w_d &= 0 \quad \text{on} \quad \Lambda, \\ \tfrac{\partial \phi}{\partial n} &= \vec{w} \cdot \hat{n} \quad \text{on} \quad \partial\Lambda \end{aligned} \tag{3}$$

It is convenient to express this process in operator form. The projection operator $\mathbb{P}$ is defined as that operator that extracts the divergence-free part of a vector field, while $\mathbb{Q}$ extracts the gradient:

$$\begin{aligned} \mathbb{P}\vec{w} &= \vec{w}_d \\ \mathbb{P} &\equiv (I - \nabla(\Delta^{-1})\nabla\cdot) \\ \mathbb{Q} &\equiv (\nabla(\Delta^{-1})\nabla\cdot) \end{aligned} \tag{4}$$

To represent complex geometries, we use a finite volume discretization defined by the intersection of the irregular domain with a rectangular grid. Primary dependent variables are descretized at Cartesian cell centers. There are several advantages to this approach. Three-dimensional grid generation is tractable with embedded boundary methods. Away from the irregular boundary, the discretization reduces to well-understood regular-grid methods. Elliptic solvers are tractable with this discretization since geometric multigrid works well in this setting.

The algorithm used in the present work is a second-order predictor corrector scheme in time, following [12,11,3]. This predictor-corrector formulation reduces the Navier Stokes calculation to separate evaluation of classical PDE discretizations. The parabolic term is advanced and then the elliptic constraint is enforced. We use an approximate projection to enforce the divergence-free constraint following [22,1,4].

1.2 Prior Work

There have been many efforts in two dimensions to solve the incompressible flow equations with embedded boundaries. Almgren, et. al [2] present a second-order algorithm to solve the two-dimensional incompressible Euler equations using a finite volume discretization. Popinet [28] solves the two-dimensional incompressible Euler equations and uses a second, order, finite difference, tree-based adaptive strategy. Calhoun [8] solves vorticity-stream function representation of the Navier Stokes equations in two dimensions and also shows second order convergence. There have been many others.

Recently there have been several algorithms published that use finite differences to solve the incompressible Navier Stokes equations in three dimensions. Marella, et al. [30] solve the three-dimensional Navier Stokes equations in the context of moving boundaries. They use level sets for the description of the geometry and use a finite difference discretization of the equations (they present no formal convergence tests). Gilmanov, et al. [17] solve the 3D Navier Stokes equations in the context of moving boundaries using a hybrid embedded boundary-immersed boundary method. This finite difference method uses a staggered grid and shows second order convergence.

The present work follows from finite volume methods in other fields. Pember, et. al [27] solve the compressible Euler equations using an early embedded boundary method. Modiano, et. al [26] made that method consistent by improving the discretization at the embedded boundary. Colella, et. al [14] extended that method to three dimensions and improved its stability. Johansen, et. al [19] solve elliptic equations using finite volume discretizations at the embedded boundary. Schwartz, et. al [31] and McCorquodale, et. al [24] solve parabolic equations using a similar discretization. It is in these works that it was found that a Crank Nicholson time discretization is only marginally stable with finite volume embedded boundary discretizations. They use an L^∞-stable time discretization developed by [33].

2 Temporal discretization

This starting point for the semi-implicit approach is the discretization in time of

$$\frac{dq}{dt} = L(q) + f \qquad q, f = q(t), f(t) \in \mathbb{R}^m \tag{5}$$

where L is (a discretization of) a second-order elliptic operator discribed in section 3.4. In [3], Crank-Nicholson was used as a basis for discretizing (5) in time. However, it was observed in [20,25], that the neutral stability of Crank-Nicholson interacts with the embedded boundary discretization of the Laplacian to give an unstable method for (5). Following [25,33], we use instead a second-order accurate, L^∞-stable implicit Runga-Kutta method that is [2]

$$q^{n+1} = (I - \mu_1 L)^{-1}(I - \mu_2 L)^{-1}((I + \mu_3 L)q^n + (I + \mu_4 L)f^{n+\frac{1}{2}})$$

$$f^{n+\frac{1}{2}} = f((n + \tfrac{1}{2})\Delta t), \quad q^n \approx q(n\Delta t)$$

$$L^{TGA}(q^n, f^{n+\frac{1}{2}}) = \frac{q^{n+1} - q^n}{\Delta t} - f^{n+\frac{1}{2}}$$

If $q(t)$ is a solution of (5), then

$$L^{TGA}(q^n, f^{n+\frac{1}{2}}) = (Lq)((n + \tfrac{1}{2})\Delta t) + O(\Delta t^2)$$

We use this method to discretize the projection form of the Navier-Stokes equations. If $\vec{u}^n \approx \vec{u}(n\Delta t), \quad p^{n-\frac{1}{2}} \approx p((n - \frac{1}{2})\Delta t)$, where $\vec{u}(t), p(t)$ are spatial discretizations of

[2] For a second-order L^∞-stable method, following [33], we pick $a > 1/2$ and

$$\mu_1 = \tfrac{a - \sqrt{a^2 - 4a + 2}}{2}\Delta t$$
$$\mu_2 = \tfrac{a + \sqrt{a^2 - 4a + 2}}{2}\Delta t$$
$$\mu_3 = (1 - a)\Delta t$$
$$\mu_4 = (\tfrac{1}{2} - a)\Delta t$$

For a method that uses real arithmetic only, the truncation error is minimized by taking $a = 2 - \sqrt{2} - \epsilon$, where ϵ is machine precision.

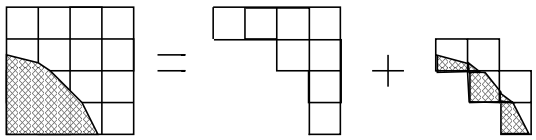

Fig. 1. Decomposition of the grid into regular, irregular, and covered cells. The gray regions are outside the solution domain.

the solution to (**??**), then

$$\begin{aligned} \vec{u}^* &= \vec{u}^n - \Delta t(\vec{u}\cdot\nabla\vec{u}^{n+\frac{1}{2}}) + L^{TGA}(\vec{u}^n, -\vec{u}\cdot\nabla\vec{u}^{n+\frac{1}{2}} - \nabla p^{n-\frac{1}{2}}) \\ \vec{u}^{n+1} &= \mathbb{P}\vec{u}^* \\ \nabla p^{n+\frac{1}{2}} &= \tfrac{-1}{\Delta t}\mathbb{Q}\vec{u}^* \end{aligned} \tag{6}$$

where $\vec{u}\cdot\nabla\vec{u}^{n+\frac{1}{2}}$ approximates the advective derivative at time $(n+\frac{1}{2})\Delta t$ and $\mathbb{P}$ and $\mathbb{Q}$ are spatial approximations to the projection operator defined in (4). In the next section, we define in detail the spatial discretizations.

3 Spatial Discretization

3.1 Embedded boundary description

Cartesian grids with embedded boundaries are useful to describe finite-volume representations of solutions to PDE in the presence of irregular boundaries. In Figure 1, the grey area represents the region excluded from the solution domain. The underlying description of space is given by rectangular control volumes on a Cartesian grid $\Upsilon_{\boldsymbol{i}} = [(\boldsymbol{i}-\frac{1}{2}\boldsymbol{V})h, (\boldsymbol{i}+\frac{1}{2}\boldsymbol{V})h], \boldsymbol{i}\in\mathbb{Z}^D$, where D is the dimensionality of the problem, h is the mesh spacing, and $\boldsymbol{V}$ is the vector whose entries are all one. Given an irregular domain Ω, we obtain control volumes $V_{\boldsymbol{i}} = \Upsilon_{\boldsymbol{i}} \bigcap \Omega$ and faces $A_{\boldsymbol{i}\pm\frac{1}{2}\hat{x}^d}$ which are the intersection of the boundary of $\partial V_{\boldsymbol{i}}$ with the coordinate planes $\{\vec{x} : x_d = (i_d \pm \frac{1}{2})h\}$. We also define $A^B_{\boldsymbol{i}}$ to be the intersection of the boundary of the irregular domain with the Cartesian control volume: $A^B_{\boldsymbol{i}} = \partial\Omega \bigcap \Upsilon_{\boldsymbol{i}}$. For ease of exposition, we will assume here that there is only one control volume per Cartesian cell. The algorithm described here has been generalized to allow for boundaries whose width is less than the mesh spacing.

To construct finite-difference methods using this description, we will need several

quantities derived from these geometric objects.

- Volume fractions κ and area fraction α:

$$\kappa_{\boldsymbol{i}} = \frac{|V_{\boldsymbol{i}}|}{h^{\boldsymbol{D}}} \quad \alpha_{\boldsymbol{i}+\frac{1}{2}\hat{x}_s} = \frac{|A_{\boldsymbol{i}+\frac{1}{2}\hat{x}_s^d}|}{h^{(\boldsymbol{D}-1)}}, \quad \alpha_{\boldsymbol{i}}^B = \frac{|A_{\boldsymbol{i}}^B|}{h^{\boldsymbol{D}-1}}$$

- The centroids of the faces and of $A_{\boldsymbol{i}}^B$; and $\mathbf{n}$, the average of outward normal of $\partial\Omega$ over $A_{\boldsymbol{i}}^B$.

$$\vec{x}_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d} = \left[\frac{1}{|A_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d}|}\int\limits_{A_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d}} \vec{x}dA\right]$$

$$\vec{x}_{\boldsymbol{i}}^B = \left[\frac{1}{|A_{\boldsymbol{i}}^B|}\int\limits_{A_{\boldsymbol{i}}^B} \vec{x}dA\right]$$

$$\hat{n}_{\boldsymbol{i}} = \frac{1}{|A_{\boldsymbol{i}}^B|}\int\limits_{A_{\boldsymbol{i}}^B} \hat{n}dA$$

where $\boldsymbol{D}$ is the dimension of space and $1 <= d <= \boldsymbol{D}$. We assume we can compute all derived quantities to $O(h^2)$. Colella, et. al [15] contains our algorithm for calculating these geometric quantities at sufficient accuracy from a surface description.

We then define the conservative divergence approximation of a flux $\vec{F}$

$$D(\vec{F})_{\boldsymbol{v}} = \frac{1}{h\kappa_{\boldsymbol{v}}}((\sum_{d=1}^{\boldsymbol{D}}(\alpha_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d}\widetilde{F^d}_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d} - \alpha_{\boldsymbol{i}-\frac{1}{2}\hat{x}^d}\widetilde{F^d}_{\boldsymbol{i}-\frac{1}{2}\hat{x}^d}) + \alpha_{\boldsymbol{v}}^B F_{\boldsymbol{v}}^B) \tag{7}$$

where $\widetilde{F}_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d}$ indicates that the flux has been interpolated to the face centroid using linear (2D) or bilinear (3D) interpolation of face-centered fluxes, as in [31]. For example, given the face with outward normal $\hat{x}^1$, with centroid $\vec{x}$, define the linearly interpolated flux in the d $(d \neq 1)$ direction by

$$\widetilde{F^d}_{\boldsymbol{i}+\frac{1}{2}\hat{x}^1} = \eta F_{\boldsymbol{i}+\frac{1}{2}\hat{x}^1} + (1-\eta)F_{\boldsymbol{i}+\frac{1}{2}\hat{x}^1\pm\hat{x}^d}$$

$$\eta = 1 - \frac{|\vec{x}\cdot\hat{x}^d|}{h_d} \tag{8}$$

$$\pm = \begin{cases} + & \vec{x}\cdot\hat{x}^d > 0 \\ - & \vec{x}\cdot\hat{x}^d \leq 0. \end{cases}$$

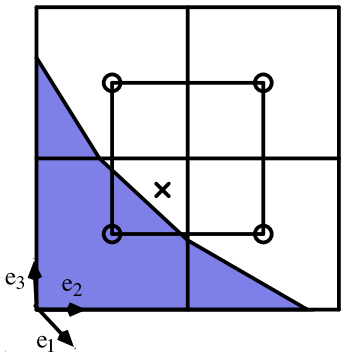

Fig. 2. Three-dimensional bilinear flux

The bilinear interpolation of the flux for the face with normal $\hat{x}^1$ can be written

$$\widetilde{F}_{\boldsymbol{i}+\frac{1}{2}\hat{x}^1} = \omega F^d_{\boldsymbol{i}+\frac{1}{2}\hat{x}^1} + (1-\omega) F^d_{\boldsymbol{i}\pm\hat{x}^{d'}+\frac{1}{2}\hat{x}^1}$$

$$\omega = 1 - \frac{|\vec{x}\cdot\hat{x}^{d'}|}{h_{d'}} \tag{9}$$

$$\pm = \begin{cases} + & \vec{x}\cdot\hat{x}^{d'} > 0 \\ - & \vec{x}\cdot\hat{x}^{d'} \leq 0 \end{cases}$$

where $d' \neq d$, $d' \neq 1$, as in figure 2.

3.2 Projection discretization

We need to define two types of projection. The first, a so called "MAC" projection is used to extract the divergence-free component of a velocity field. The second, referred to here as a "cell- centered" projection operator, is used to extract the divergence-free component of a cell-centered velocity field.

We define the face-centered gradient of a cell-centered scalar ϕ to be the ordinary second-order accurate finite-difference approximation in the normal direction and the average of neighboring normal gradients in other directions,

$$G^{mac}(\phi)^d_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d} = \frac{1}{h}(\phi_{\boldsymbol{v}^+(\boldsymbol{i}+\frac{1}{2}\hat{x}^d)} - \phi_{\boldsymbol{v}^-(\boldsymbol{i}+\frac{1}{2}\hat{x}^d)}),$$

and for $d' \neq d$

$$G^{mac}(\phi)^{d'}_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d} = \frac{1}{N_{\mathcal{G}}} \sum_{\boldsymbol{i}+\frac{1}{2}\hat{x}^{d'} \in \mathcal{G}^{d',d}} (G^{mac}(\phi)^{d'}_{\boldsymbol{i}+\frac{1}{2}\hat{x}^{d'}}).$$

Where $\mathcal{G}^{d',d}$ is the set of faces in the d' direction who contain one control volume in $\boldsymbol{i} + \frac{1}{2}\hat{x}^d$ and $N_{\mathcal{G}}$ is the number of faces in this set. On a regular grid, $\mathcal{G}$ is the set of four neighboring faces in the d' direction. We define our discrete Laplacian operator to be the conservative divergence of the face-centered gradient,

$$L \equiv DG^{mac}$$

Our MAC projections are defined to be

$$\mathbb{P}^{mac} \equiv (I - G^{mac}(L^{-1})D)$$

$$\mathbb{Q}^{mac} \equiv (I - \mathbb{P}^{mac})$$

Operationally, we first solve the discrete Poisson's equation

$$\kappa_{\boldsymbol{i}} L\phi_{\boldsymbol{i}} = \kappa_{\boldsymbol{i}} D^{mac}(\vec{u}_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d}) \tag{10}$$

for $\phi_{\boldsymbol{i}}$ and subsequently subtract the face-centered gradient from the velocity to obtain the divergence-free field

$$\mathrm{P}^{mac}(\vec{u})_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d} = \vec{u}_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d} - G^{mac}\phi_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d}$$

The boundary conditions for the solve (10) are $\nabla\phi \cdot \hat{n} = 0$ at inflow or no-flow boundaries, $\phi = 0$ at outflow boundaries. Since the embedded boundary is a no-flow boundary, $F^B = 0$ (see equation 7) in this context. Trebotich, et al. [32] explain how these boundary conditions are correct even at inflow faces as the divergence of the velocity field holds the inhomogeneity.

Our cell-centered projection is the composition of our MAC projection with two averaging operators. We define an operator to average cell-centered velocites to face centers: For a face with a normal direction d we have

$$A^{C\to F}(u^d)_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d} = \tfrac{1}{2}(u^d_{\boldsymbol{v}+\hat{x}^d} + u^d_{\boldsymbol{v}})$$

We also define an averaging operator to average gradients from face centers to cell centers.

$$A^{F\to C}(G^{mac}(\phi)^d)_{\boldsymbol{i}} = \tfrac{1}{2}(\bar{G}^{mac,d}_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d}(\phi) + \bar{G}^{mac,d}_{\boldsymbol{i}-\frac{1}{2}\hat{x}^d}(\phi))$$

where $\bar{G}^{mac,d}_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d}(\phi) = G^{mac,d}_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d}(\phi)$ if $(\alpha_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d} > 0)$. Otherwise, we must extrapolate to a covered face, as described in Section 3.6. The cell-centered projection operators are defined to be

$$\mathbb{P}^{cc}(\vec{u}) = \vec{u} - A^{F\to C}(\mathbb{Q}^{mac}(A^{C\to F}(\vec{u})).$$
$$\mathbb{Q}^{cc} \equiv (I - \mathbb{P}^{cc}).$$

3.3 Stability of the approximate projection

One can separate projections into two categories: discrete and approximate. To define a projection operator, one defines a discrete divergence D, a gradient discretization G and a Laplacian discretization L. A discrete projection is a projection whose divergence operator is the discrete adjoint of its gradient operator with respect to some appropriate discrete scalar inner product and vector inner product $<G^h\phi, \vec{u}> = <D\vec{u}, \phi>$ and whose Laplacian operator is given by $L \equiv DG$. An approximate projection is one which does not meet one of these constraints. Discrete projections are idempotent $\mathbb{P}(\mathbb{P}(\vec{u})) = \mathbb{P}(\vec{u})$. This is an attractive property that can simplify algorithm design. The MAC projection described above is a discrete projection.

There are some disadvantages to discrete projections, however. For colocated velocities and symmetric discretizations of divergence and gradient, $L \equiv DG$ typically produces discretizations of the Laplacian that are badly behaved (the stencils become decoupled [18]). Approximate projections simplify numerical linear algebra by sacrificing some of the design advantages associated with discrete projections. The cell-centered projection described in the previous section is an approximate projection because the Laplacian is not the composition of the divergence and gradient operators ($L \neq DG$) and is therefore not idempotent ($\mathbb{P}(\mathbb{P}(\vec{u})) \neq \mathbb{P}(\vec{u})$). Following Martin, et. al [23], we have designed our update equation 6 around having all of the velocity field projected at every time step, as opposed to just projecting an update (the methods are eqivalent if one uses a discrete projection). In this section, we show that our approximate projection is a stable operator in that the divergence of a velocity field diminishes with repeated application of the projection. We start with an initial velocity field of potential flow over a sphere (or cylinder in 2d) with radius = 0.1. The sphere is in the center of a unit square domain. We iteratively project the velocity field $\vec{u}$ and evaluate the norm of the diverence $\kappa D\vec{u}$ the norm of $\nabla\phi$ after each projection. Figures 3, 4, 5 and 6 show that all norms of both quantities monotonically decrease with number of projection iterations.

3.4 Viscous operator discretization

In section 2 we describe or temporal discretization. In that section, we make reference to a Helmholtz operator. In this section we define that operator. We are solving

$$(I + \mu L)\phi = rho \tag{11}$$

where μ is a constant. Just as we did for the MAC projection, we discretize $L \equiv DG^{mac}$ (see equation 7). In this context, however, since the irregular boundary is a no-slip

boundary, we must solve (11) with Dirichelet boundary conditions $\phi = 0$ on the irregular boundary. To do this, we must compute

$$F^B = \frac{\partial \phi}{\partial \hat{n}}$$

at the embedded boundary. We follow Schwartz, et. al [31] and compute this gradient by casting a ray into space, interpolating ϕ to points along the ray, and computing the normal gradient of phi by differencing the result. In three dimensions, refer to figure 7. We cast a ray along the normal of the VoF from the centroid of area of the irregular face C. We find the closest points B and C where the ray intersects the planes formed by cell centered points. The axes of these planes d_1, d_2 will be the directions not equal to the largest direction of the normal. We use biquadratic interpolation to interpolate data from the nearest cell centers to the intersection points B and C. In two dimesions, we find the nearest lines of cell centers (instead of planes) and the interpolation is quadratic. We then use this interpolated data to compute a $O(h^2$ approximation of $\frac{\partial \phi}{\partial \hat{n}}$. In the case where there are not enough cells to cast this ray, we use a least-squares approximation to $\frac{\partial \phi}{\partial \hat{n}}$ which is $O(h)$. As shown in [19], the modified equation analysis shows that, for Dirichlet boundary conditions, it is sufficient to have $O(1)$ boundary condtions to achieve second order solution error convergence for elliptic equations.

3.5 Advective derivative discretization

Since the flow is incompressible, we can discretize the solution $\vec{u} \cdot \nabla \vec{u} = \nabla \cdot (\vec{u}\vec{u})$. We use (7) with $\mathbf{F} = \vec{u}\vec{u}$ to generate a conservative discretization of $\nabla \cdot (\vec{u}\vec{u})$. Ideally we would like to use the this approximation to $\vec{u} \cdot \nabla \vec{u}$ in our solution update. The difficulty with this approach is that the CFL stability constraint on the time step is at best $\Delta t = O(\frac{h}{v_{\boldsymbol{i}}^{max}}(\kappa_{\boldsymbol{i}})^{\frac{1}{D}})$, where $v_{\boldsymbol{i}}^{max}$ is the magnitude of the maximum wave speed for the $\boldsymbol{i}^{th}$ control volume. This is the well known small-cell problem for embedded boundary methods. There have been a number of proposals to deal with this problem, including merging the small control volumes with nearby larger ones [29,13], and the development of specialized stencils that guarantee the required cancellations [7,6,9,21]. The approach we have taken to this problem has been to expand the range of influence of the small control volumes algebraically to obtain a stable method [10,5,27]. The starting point for this approach is to compute a stable but non-conservative approximation to $\vec{u} \cdot \nabla \vec{u}$ that does not include the effect of the embedded boundary,

$$\vec{u} \cdot \nabla \vec{u}_{\boldsymbol{i}}^{NC} = \frac{1}{2h} \sum_{d=1}^{D} (u_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d}^{n+\frac{1}{2},d} + u_{\boldsymbol{i}-\frac{1}{2}\hat{x}^d}^{n+\frac{1}{2},d})(\vec{u}_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d}^{n+\frac{1}{2}} - \vec{u}_{\boldsymbol{i}-\frac{1}{2}\hat{x}^d}^{n+\frac{1}{2}}). \tag{12}$$

where the fluxes in this expression are centered at $(\boldsymbol{i} \pm \frac{1}{2}\hat{x}^d)h$. We use a linear hybridization of the two estimates of $\vec{u} \cdot \nabla \vec{u}$,

$$\vec{u} \cdot \nabla \vec{u}^{n+\frac{1}{2}} = \kappa_{\boldsymbol{i}} (\nabla \cdot (\vec{u}\vec{u}))_{\boldsymbol{i}} + (1 - \kappa_{\boldsymbol{i}})(\vec{u} \cdot \nabla \vec{u})^{NC}_{\boldsymbol{i}}). \tag{13}$$

The small denominator in $\nabla \cdot (\vec{u}\vec{u})$ is canceled and we obtain a stable method. However, the method fails to conserve. This lack of conservation is measured by the difference between the hybrid discretization and the conservative,

$$\delta M_{\boldsymbol{i}} = \kappa_{\boldsymbol{i}} (\nabla \cdot (\vec{u}\vec{u})^{n+\frac{1}{2}} - \vec{u} \cdot \nabla \vec{u}^{n+\frac{1}{2}}) = \kappa_{\boldsymbol{i}} (1 - \kappa_{\boldsymbol{i}})(\nabla \cdot (\vec{u}\vec{u})_{\boldsymbol{i}} - \vec{u} \cdot \nabla \vec{u}^{NC}_{\boldsymbol{i}}).$$

To maintain overall conservation, we redistribute $\delta M_{\boldsymbol{i}}$ into nearby cells,

$$\vec{u} \cdot \nabla \vec{u}^{n+\frac{1}{2}}_{\boldsymbol{i}'} := \vec{u} \cdot \nabla \vec{u}^{n+\frac{1}{2}}_{\boldsymbol{i}'} + w_{\boldsymbol{i},\boldsymbol{i}'} \delta M_{\boldsymbol{i}} \ , \ \boldsymbol{i}' \in N(\boldsymbol{i}), \tag{14}$$

$$w_{\boldsymbol{i},\boldsymbol{i}'} \geq 0, \quad \sum_{\boldsymbol{i}' \in N(\boldsymbol{i})} w_{\boldsymbol{i},\boldsymbol{i}'} \kappa_{\boldsymbol{i}'} = 1 \tag{15}$$

where $N(\boldsymbol{i})$ is some set of indices in the neighborhood of $\boldsymbol{i}$. The sum condition (15) makes the redistribution step conservative. The weights $w_{\boldsymbol{i},\boldsymbol{i}'}$ must be bounded independent of $(\kappa_{\boldsymbol{i}'})^{-1}$. We use volume weighted redistribution,

$$w_{\boldsymbol{i},\boldsymbol{i}'} = (\sum_{\boldsymbol{i}' \in N(\boldsymbol{i})} \kappa_{\boldsymbol{i}'})^{-1}$$

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

Our procedure for calculating the velocities used to compute $\nabla \cdot (\vec{u}\vec{u})$ assumes that we have a second-order accurate method for computing velocities at the centers of cell faces.

$$\vec{u}^{n+\frac{1}{2}}_{\boldsymbol{i}+\frac{1}{2}\hat{x}^d} = \vec{u}((\boldsymbol{i} + \tfrac{1}{2}\hat{x}^d)h, t^n + \frac{\Delta t}{2}) + O(h^2) \tag{16}$$

We use these velocities in (12) to compute $\vec{u} \cdot \nabla \vec{u}^{NC}$. To compute $\nabla \cdot (\vec{u}\vec{u})$, we interpolate $\vec{u}\vec{u}$ to face centroids as in (8)-(9). Critical to the success of this approach is the calculation of $\vec{u} \cdot \nabla \vec{u}^{NC}$. In control volumes with $\kappa_{\boldsymbol{i}} << 1$, $(\vec{u} \cdot \nabla \vec{u})^{NC}$ is almost entirely responsible for the update of $\vec{u}_{\boldsymbol{i}}$. For that reason, $\vec{u} \cdot \nabla \vec{u}^{NC}$ must be designed carefully so that, for example, the solution in small control volumes comes into equilibrium with the larger control volumes around it.

A covered face is a face whose aperture vanishes. To compute $\vec{u} \cdot \nabla \vec{u}^N C$ (see (12)), we need a second-order solution at covered faces. The flux is obtained by choosing the upwind value at the face. For the side of the face next to the control volume, we use the extrapolated state from the control volume. For the other side of the covered face, we must extrapolate from values at neighboring faces.

Specifically, assume that all $\boldsymbol{i}$ is not covered, but $\boldsymbol{i} \mp \hat{x}^d$ is covered, so that the face connecting the two is covered. Then we want to compute $\vec{u}_{\boldsymbol{i} \mp \hat{x}^d, \pm, d}$, given a collection of values $\{W_{\boldsymbol{i}', \pm, d}\}$ that are assumed to be defined if $\alpha_{\boldsymbol{i}' \pm \frac{1}{2}\hat{x}^d} \neq 0$.

3.6.1 Two-dimensional Extrapolation

In two dimensions, extrapolation to covered faces is done as illustrated in Figure 8. First we define the control volumes involved.

$$\begin{aligned} \boldsymbol{i}^u &= \boldsymbol{i} + s^{d'} \hat{x}^{d'} - s^d \hat{x}^d \\ \boldsymbol{i}^s &= \boldsymbol{i} + s^d \hat{x}^d \\ \boldsymbol{i}^c &= \boldsymbol{i} + s^{d'} \hat{x}^{d'} \end{aligned}$$

where $d' \neq d$ and $s^d = \text{sign}(n^d)$.

Define $\vec{u}^{u,s,c}$, extrapolations to the edges near the control volumes near $\boldsymbol{i}$.

$$\begin{aligned} \vec{u}^u &= \vec{u}_{\boldsymbol{i}^u, \mp, d} \\ \vec{u}^s &= \vec{u}_{\boldsymbol{i}^s, \mp, d} - s^d \Delta^d \vec{u} \\ \vec{u}^c &= \vec{u}_{\boldsymbol{i}^c, \mp, d} \end{aligned}$$

To extrapolate to the covered faces, we use a linear combination of the values defined above to compute the value along a ray normal to the boundary and passing through the center of the covered face. We then extrapolate that value to the covered face using the second-order slopes combined with characteristic limiting described in Section 3.7. In the case where one of the values being used to interpolate corresponds to a value on the cell adjacent to the covered face in question, (the case illustrated in Figure 8) we use a value extrapolated from $\boldsymbol{i}^s$ (the cell adjacent in the d direction) rather than $\boldsymbol{i}$. This choice satisfies the design criterion that the action of the nonconservative evolution should, over time, tend to make the solution at $\boldsymbol{i}$ tend toward the value of a locally constant solution in the surrounding cells. This was the design criterion

for computing covered faces in [27]; the procedure given here has the same goal, but using an approach that produces second-order accurate fluxes. For example, in the case of a linear equation and the normal pointing in the $\hat{x}^1$ direction, extrapolation from a locally constant state to the right of $\boldsymbol{i}$ in Figure 8 leads to the solution in $\boldsymbol{i}$ to eventually take on that constant value. If one used the value at the face extrapolated from $\boldsymbol{i}$, the solution would tend to the locally constant value be true only if the advection velocity were negative; otherwise, the value at $\boldsymbol{i}$ would remain unchanged.

If $|n_d| < |n_{d'}|$:

$$\vec{u}_{\boldsymbol{i}\mp\hat{x}^d,\pm,d} = \frac{|n_d|}{|n_{d'}|}\vec{u}^c + (1 - \frac{|n_d|}{|n_{d'}|})\vec{u}^u - (\frac{|n^d|}{|n^{d'}|}s^d\Delta^d\vec{u} + s^{d'}\Delta^{d'}\vec{u})$$

$$\Delta^{d''}\vec{u} = \frac{|n_d|}{|n_{d'}|}\Delta_2^{d''}\vec{u}^n_{\boldsymbol{i}^c} + (1 - \frac{|n_d|}{|n_{d'}|})\Delta_2^{d''}\vec{u}^n_{\boldsymbol{i}^u},\ d'' = 1, 2 \tag{17}$$

If $|n_d| \geq |n_{d'}|$:

$$\vec{u}_{\boldsymbol{i}\mp\hat{x}^d,\pm,d} = \frac{|n_{d'}|}{|n_d|}\vec{u}^c + (1 - \frac{|n_{d'}|}{|n_d|})\vec{u}^s - (\frac{|n^{d'}|}{|n^d|}s^{d'}\Delta^{d'}\vec{u} + s^d\Delta^d\vec{u})$$

$$\Delta^{d''}\vec{u} = \frac{|n_{d'}|}{|n_d|}\Delta_2^{d''}\vec{u}^n_{\boldsymbol{i}^c} + (1 - \frac{|n_{d'}|}{|n_d|})\Delta_2^{d''}\vec{u}^n_{\boldsymbol{i}^s},\ d'' = 1, 2 \tag{18}$$

We found that the use of the linear interpolation algorithms (17), (18) to compute the slopes used in extrapolating to the covered faces led to a more robust and accurate algorithm than other simpler choices that we considered. The intent is to use slopes computed at the same cell centers as the values used in the original linear interpolation in Figure 8, and in the same proportions. By using that choice, it appears that no further limiting of those slopes is required.

If one or both of the faces from which we are extrapolating are covered we drop order. If only one of the faces is covered we set the extrapolated value to be the value on the other face. If both faces are covered, we set the extrapolated value to $\vec{u}^n_{\boldsymbol{i}}$.

3.6.2 Extrapolation to Covered Face in Three Dimensions

We define the direction of the face normal to be d and d_1, d_2 to be the directions tangential to the face. The procedure extrapolation procedure is given as follows.

- Define the associated control volumes.

- Form a 2×2 grid of values along a plane h away from the covered face and bilinearly interpolate to the point where the normal intersects the plane.
- Use the slopes of the solution to extrapolate along the normal to obtain a second-order approximation of the solution at the covered face.

Which plane is selected is determined by the direction of the normal. See Figure 9 for an illustration.

If $|n^d| \geq |n^{d_1}|, |n^{d_2}|$, we define a bilinear function $\mathcal{B}$ that interpolates the 2×2 grid of values.

$$\mathcal{B}(Q, \Delta) = A + B\xi + C\eta + D\xi\eta - \zeta\Delta_{\boldsymbol{i}^{00}} \tag{19}$$

$$\begin{aligned}
A &= Q_{\boldsymbol{i}^{00}} \\
B &= s^{d_1}(Q_{\boldsymbol{i}^{01}} - Q_{\boldsymbol{i}^{00}}) \\
C &= s^{d_2}(Q_{\boldsymbol{i}^{10}} - Q_{\boldsymbol{i}^{00}}) \\
D &= s^{d_1}s^{d_2}(Q_{\boldsymbol{i}^{11}} - Q_{\boldsymbol{i}^{00}}) - (Q_{\boldsymbol{i}^{10}} - Q_{\boldsymbol{i}^{00}}) - (Q_{\boldsymbol{i}^{01}} - Q_{\boldsymbol{i}^{00}})
\end{aligned}$$

$$\xi = \frac{|n_{d_1}|}{|n_d|},\ \eta = \frac{|n_{d_2}|}{|n_d|},\ \zeta = -1 + s^{d_1}\xi + s^{d_2}\eta + (s^{d_1}s^{d_2} - 2s^d)\xi\eta$$

$$\begin{aligned}
s^{d_i} &= sign(n_{d_i}) \\
\boldsymbol{i}^{00} &= \boldsymbol{i} + s^d\hat{x}^d \\
\boldsymbol{i}^{10} &= \boldsymbol{i} + s^{d_1}\hat{x}^{d_1} \\
\boldsymbol{i}^{01} &= \boldsymbol{i} + s^{d_2}\hat{x}^{d_2} \\
\boldsymbol{i}^{11} &= \boldsymbol{i} + s^{d_1}\hat{x}^{d_1} + s^{d_2}\hat{x}^{d_2}
\end{aligned}$$

$\mathcal{B}$ interpolates the values in the (d_1, d_2) plane in Figure 9, with $Q_{\boldsymbol{i}^{00}}$ the value at A, $Q_{\boldsymbol{i}^{00}} - s^d\Delta_{\boldsymbol{i}^{00}}$ the value at point B, and the remaining values filling in the bilinear stencil. Using this function, we can define the extrapolated value on the covered face.

$$\begin{aligned}
\vec{u}_{\boldsymbol{i}\mp\hat{x}^d,\pm,d} =& \mathcal{B}(\vec{u}_{\cdot,\pm,d}, \Delta_2^d\vec{u}^n) - \mathcal{B}(\Delta_2^d\vec{u}^n, \Delta \equiv 0) \\
&- s^{d_1}\frac{|n^{d_1}|}{|n^d|}\mathcal{B}(\Delta_2^{d_1}\vec{u}^n, \Delta \equiv 0) - s^{d_2}\frac{|n^{d_2}|}{|n^d|}\mathcal{B}(\Delta^{d_2}\vec{u}, \Delta \equiv 0) \\
& d_1 \neq d_2 \neq d
\end{aligned} \tag{20}$$

We use the bilinear stencil to interpolate values of both the solution and of the slopes, except that we use piecewise-constant extrapolation to extrapolate the value of the slopes from A to B.

The case where one of the tangential directions corresponds to the largest component of the normal is similar. Assuming $|n^{d_1}| > |n^d|, |n^{d_2}|$, we define

$$\mathcal{B}(Q) = A + B\zeta + C\eta + D\xi\eta \tag{21}$$

with

$$A = Q_{\boldsymbol{i}^{00}} \tag{22}$$
$$B = s^d(Q_{\boldsymbol{i}^{01}} - Q_{\boldsymbol{i}^{00}}) \tag{23}$$
$$C = s^{d_2}(Q_{\boldsymbol{i}^{10}} - Q_{\boldsymbol{i}^{00}}) \tag{24}$$
$$D = s^d s^{d_2}(Q_{\boldsymbol{i}^{11}} - Q_{\boldsymbol{i}^{00}}) - (Q_{\boldsymbol{i}^{01}} - Q_{\boldsymbol{i}^{00}}) - (Q_{\boldsymbol{i}^{10}} - Q_{\boldsymbol{i}^{00}}) \tag{25}$$

$$\zeta = \frac{|n^d|}{|n^{d_1}|},\ \eta = \frac{|n^{d_2}|}{|n^{d_1}|} \tag{26}$$

$$\begin{aligned}
\boldsymbol{i}^{00} &= \boldsymbol{i} + s^{d_1}\hat{x}^{d_1} - s^d\hat{x}^d \\
\boldsymbol{i}^{10} &= \boldsymbol{i} + s^{d_1}\hat{x}^{d_1} - s^d\hat{x}^d + s^{d_2}\hat{x}^{d_2} \\
\boldsymbol{i}^{01} &= \boldsymbol{i} + s^{d_1}\hat{x}^{d_1} \\
\boldsymbol{i}^{11} &= \boldsymbol{i} + s^{d_1}\hat{x}^{d_1} - s^{d_2}\hat{x}^{d_2}
\end{aligned}$$

Then

$$\begin{aligned}
\vec{u}_{\boldsymbol{i}\mp\hat{x}^d,\pm,d} =& \mathcal{B}(\vec{u}_{\cdot,\pm,d}) - \mathcal{B}(\Delta_2^{d_1}\vec{u}) \\
&- s^d\frac{|n^d|}{|n^{d_1}|}\mathcal{B}(\Delta_2^d\vec{u}) - s^{d_2}\frac{|n^{d_2}|}{|n^{d_1}|}\mathcal{B}(\Delta_2^{d_2}W)
\end{aligned} \tag{27}$$

If any of the values required to perform the interpolation are unavailable, e.g. because the cells are covered, we drop order by using a weighted sum of the available values:

$$\vec{u}_{\boldsymbol{i}\mp\hat{x}^d,\pm,d} = \frac{\sum_{\boldsymbol{i}'}\vec{u}_{\boldsymbol{i}',\pm,d}\kappa_{\boldsymbol{i}'}}{\sum_{\boldsymbol{i}'}\kappa_{\boldsymbol{i}'}} \tag{28}$$

where the sums are over $\boldsymbol{i}' \in \{\boldsymbol{i}^{00}, \boldsymbol{i}^{01}, \boldsymbol{i}^{10}, \boldsymbol{i}^{11}\}$, provided that at least one of the $\boldsymbol{i}'$ is not covered. If all of the faces used for interpolation are covered, we set the extrapolated value to be $\vec{u}_{\boldsymbol{i}}^n$.

3.7 Slope Calculation

The notation

$$CC = A \mid B \mid C$$

means that the 3-point formula A is used for CC if all cell-centered values it uses are available, the 2-point formula B is used if the cell to the right (i.e. the high side) of the current cell is covered, and the 2-point formula C is used if the cell to the left (i.e. the low side) current cell is covered.

To compute the limited differences in the first step on the algorithm, we use the second-order slope calculation [3] with van Leer limiting.

$$\begin{aligned}
\Delta_2^d W_{\boldsymbol{i}} &= \Delta^{vL}(\Delta^C W_{\boldsymbol{i}}, \Delta^L W_{\boldsymbol{i}}, \Delta^R W_{\boldsymbol{i}}) \mid \Delta^{VLL} W_{\boldsymbol{i}} \mid \Delta^{VLR} W_{\boldsymbol{i}} \\
\Delta^B W_{\boldsymbol{i}} &= \tfrac{2}{3}((W - \tfrac{1}{4}\Delta_2^d W)_{\boldsymbol{i}+\hat{x}^d} - (W + \tfrac{1}{4}\Delta_2^d W)_{\boldsymbol{i}-\hat{x}^d}) \\
\Delta^C W_{\boldsymbol{i}} &= \tfrac{1}{2}(W^n_{\boldsymbol{i}+\hat{x}^d} - W^n_{\boldsymbol{i}-\hat{x}^d}) \\
\Delta^L W_{\boldsymbol{i}} &= W^n_{\boldsymbol{i}} - W^n_{\boldsymbol{i}-\hat{x}^d} \\
\Delta^R W_{\boldsymbol{i}} &= W^n_{\boldsymbol{i}+\hat{x}^d} - W^n_{\boldsymbol{i}} \\
\Delta^{3L} W_{\boldsymbol{i}} &= \tfrac{1}{2}(3W^n_{\boldsymbol{i}} - 4W^n_{\boldsymbol{i}-\hat{x}^d} + W^n_{\boldsymbol{i}-2\hat{x}^d}) \\
\Delta^{3R} W_{\boldsymbol{i}} &= \tfrac{1}{2}(-3W^n_{\boldsymbol{i}} + 4W^n_{\boldsymbol{i}+\hat{x}^d} - W^n_{\boldsymbol{i}+2\hat{x}^d}) \\
\Delta^{VLL} W_{\boldsymbol{i}} &= \min(\Delta^{3L} W_{\boldsymbol{i}}, \Delta^L W_{\boldsymbol{i}}) \quad \text{if } \Delta^{3L} W_{\boldsymbol{i}} \cdot \Delta^L W_{\boldsymbol{i}} > 0 \\
\Delta^{VLL} W_{\boldsymbol{i}} &= 0 \quad \text{otherwise} \\
\Delta^{VLR} W_{\boldsymbol{i}} &= \min(\Delta^{3R} W_{\boldsymbol{i}}, \Delta^R W_{\boldsymbol{i}}) \quad \text{if } \Delta^{3R} W_{\boldsymbol{i}} \cdot \Delta^R W_{\boldsymbol{i}} > 0 \\
\Delta^{VLR} W_{\boldsymbol{i}} &= 0 \quad \text{otherwise}
\end{aligned}$$

We apply the van Leer limiter component-wise to the differences.

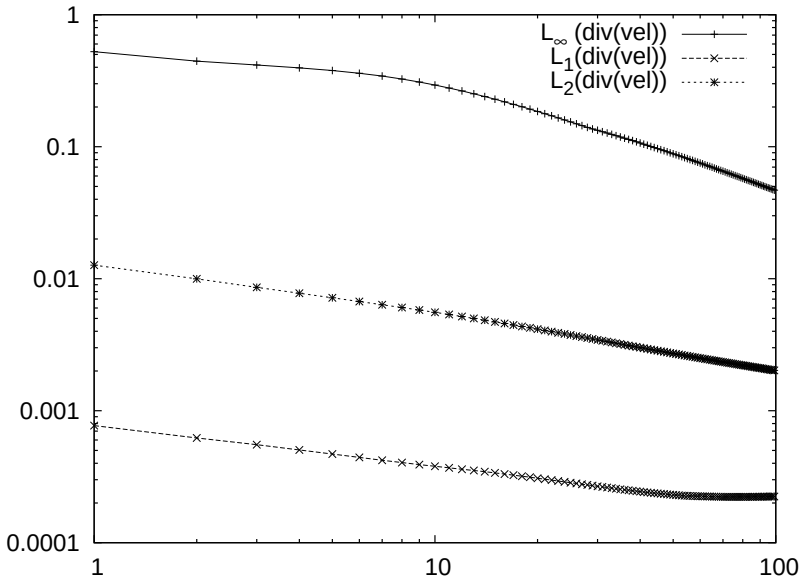

Fig. 3. Norms (L_1, L_2, L_∞) of $\kappa D\vec{u}$ versus number of cell-centered projection iterations. A 2D test where $h = 1/256$.

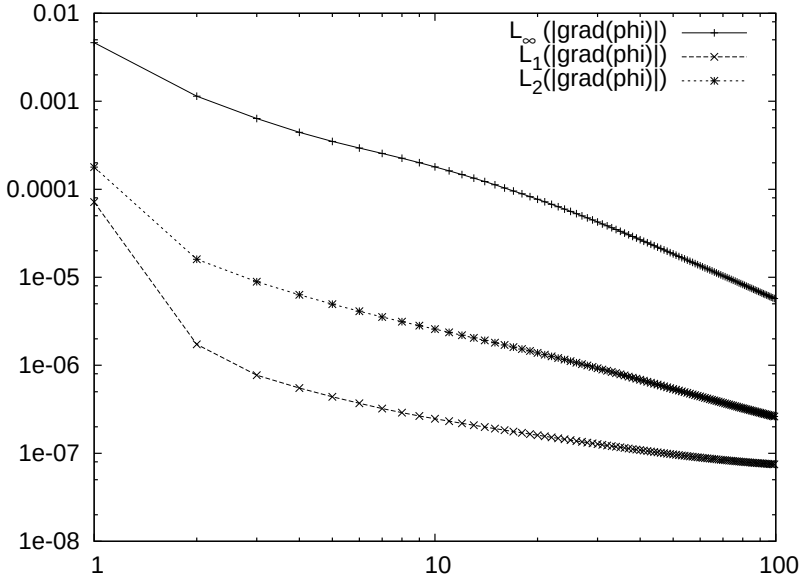

Fig. 4. Norms (L_1, L_2, L_∞) of $\nabla\phi$ versus number of cell-centered projection iterations. A 2D test where $h = 1/256$.

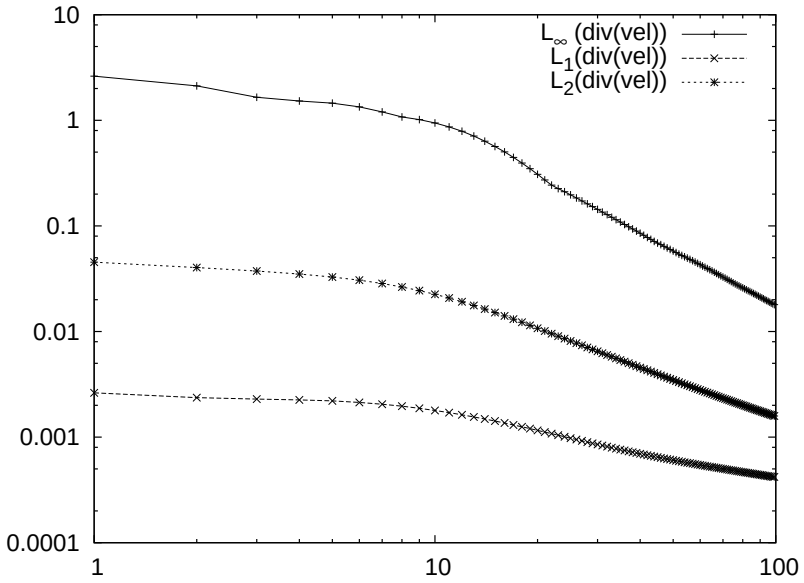

Fig. 5. Norms (L_1, L_2, L_∞) of $\kappa D\vec{u}$ versus number of cell-centered projection iterations. A 3D test where $h = 1/64$.

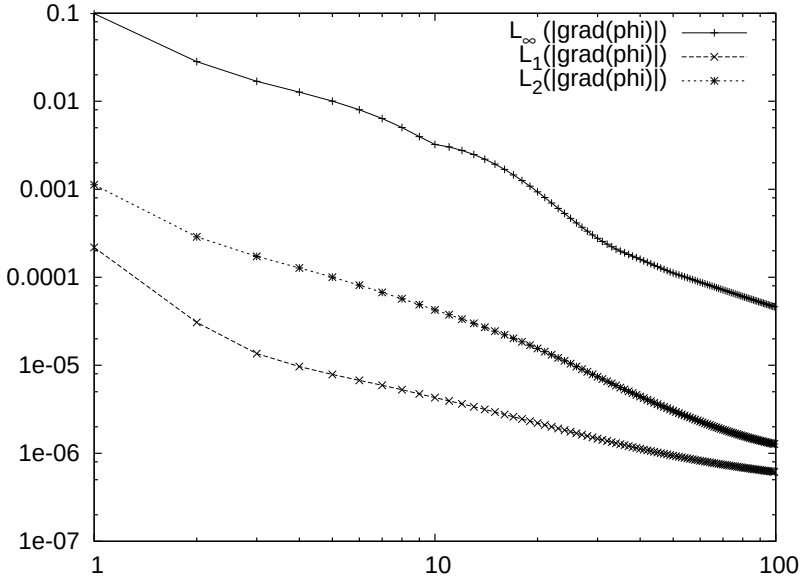

Fig. 6. Norms (L_1, L_2, L_∞) of $\nabla\phi$ versus number of cell-centered projection iterations. A 3D test where $h = 1/64$.

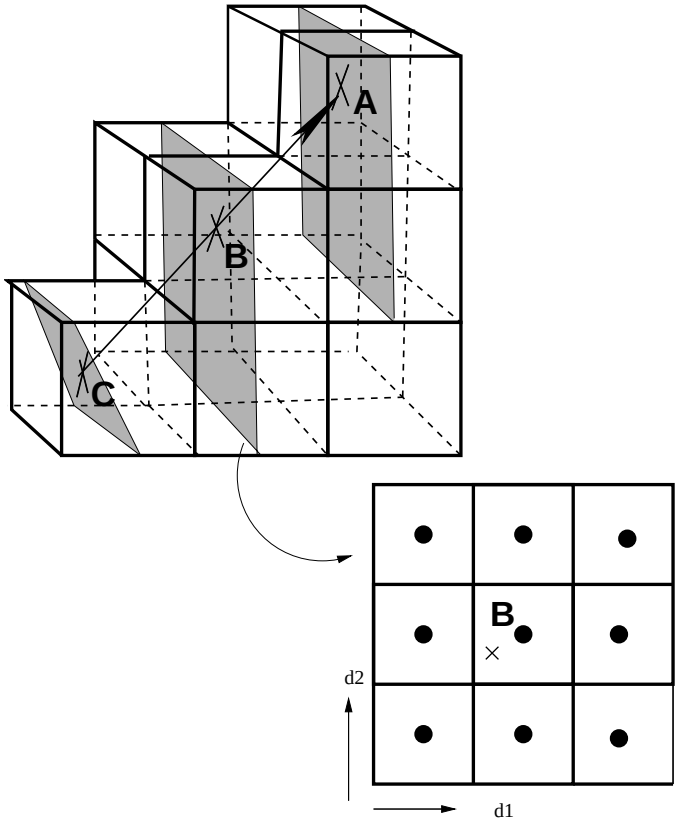

Fig. 7. Ray casting to get fluxes for Dirichlet boundary conditions at the irregular boundary. A ray is cast along the normal from the centroid of the irregular area C and the points A and B are the places where this ray intersects the planes formed by cell centers. Data is interpolated in these planes to get values at the intersection points. That data is used to compute a normal gradient of the solution.

.

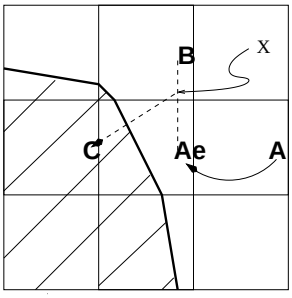

Fig. 8. Illustration of extrapolation to covered faces in two dimensions The covered face is at C. We extrapolate from A to Ae and interpolate between Ae and B to the point X where the boundary normal intersects the line. We then extrapolate back along the normal to get to the covered face.

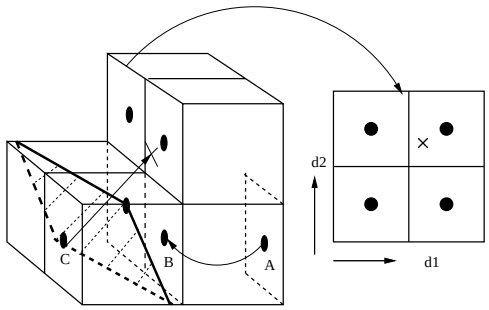

Fig. 9. Illustration of extrapolation to covered faces in three dimensions. The covered face is at C. We extrapolate from A to B to form a plane of values in $d1 - d2$. We interpolate within that plane to the point X where the boundary normal intersects the plane. We then extrapolate back along the normal to get to the covered face.

4 Results

4.1 Convergence Tests

We define a volume-weighted averaging operator A

$$A(\vec{u}^h)_{\boldsymbol{i}_c} = \frac{\sum\limits_{\boldsymbol{i}_f \in R(\boldsymbol{i}_c)} \vec{u}_{\boldsymbol{i}_f} \kappa_{\boldsymbol{i}_f}}{\sum\limits_{\boldsymbol{i}_f \in R(\boldsymbol{i}_c)} \kappa_{\boldsymbol{i}_f}},$$

where $R(\boldsymbol{i}_c)$ is the set of control volumes formed by a graph refinement of $\boldsymbol{i}_c$. The solution error at a given grid resolution, ϵ, is defined by

$$\epsilon^e \equiv \vec{u}^h(t) - \vec{u}^e(t)$$

where t is some fixed time interval independent of the mesh spacing. We approximate the exact solution by using the average of a finer solution at the same time

$$\epsilon^{2h} = \vec{u}^{2h}(t) - A(\vec{u}^h(t))$$

The L^1 norm of the error is calculated as follows:

$$L^1(E) = \frac{1}{V} \int\limits_{\Omega} |E| dV = \frac{1}{\sum\limits_{\Omega} \kappa_{\boldsymbol{i}}} \sum_{\Omega} \kappa_{\boldsymbol{i}} |E_{\boldsymbol{i}}|$$

The L^2 norm of the error is calculated as follows:

$$L^2(E) = (\frac{1}{V} \int\limits_{\Omega} E^2 dV)^{\frac{1}{2}} = (\frac{1}{\sum\limits_{\Omega} \kappa_{\boldsymbol{i}}} \sum_{\Omega} \kappa_{\boldsymbol{i}} E_{\boldsymbol{i}}^2)^{\frac{1}{2}}$$

The order of convergence p is estimated by

$$p = \frac{\log(\frac{|\epsilon^{2h}|}{|\epsilon^h|})}{\log(2)}$$

Finally, for the purpose of the convergence study, we have turned off the van Leer limiters, using instead the linear difference formulas for computing slopes. This allows us to determine the extent to which the modified equation analysis is valid, without the contaminating effects of limiters acting at extrema in the interior of the domain.

The geometry is set to be a sphere (diameter = 0.1) in the center of a x-axis aligned cylinder (diameter = 0.98). See 10 for an illustration. The inflow velocity condition is Poiseuille flow with unit maximum velocity. We set the viscosity $\nu = 0.01$. The

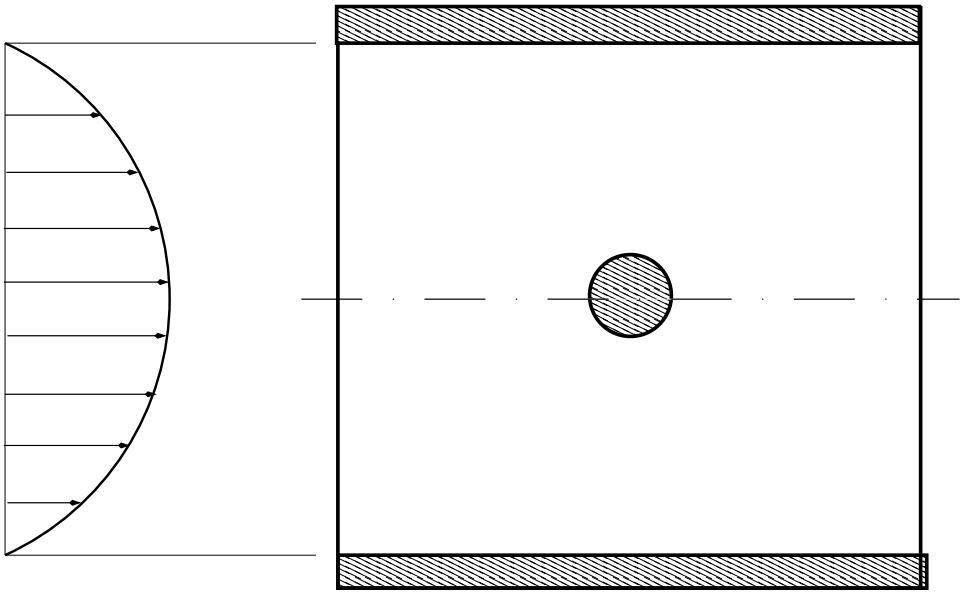

Fig. 10. Initial and boundary conditions for the convergence tests. The geometry is set to be a sphere (diameter = 0.1) in the center of a x-axis aligned cylinder (diameter = 0.98). The inflow velocity condition is Poiseuille flow with maximum velocity = 1.

N	Variable	$E_c = A(u_{2h}) - u_{4h}$	$E_f = A(u_h) - u_{2h}$	p
∞	v_x	2.625923e-01	1.220771e-01	1.10
∞	v_y	2.152841e-01	9.728703e-02	1.14
∞	p	5.282043e-02	5.484670e-02	-0.05
1	v_x	8.017870e-04	1.249725e-04	2.68
1	v_y	5.110770e-04	8.041446e-05	2.66
1	p	3.726635e-03	5.778275e-04	2.68
2	v_x	2.822005e-03	1.004912e-03	1.48
2	v_y	1.787750e-03	5.285102e-04	1.75
2	p	4.955850e-03	8.213788e-04	2.59

Table 1
Solution error in two dimensions. Convergence rates are calculated using L_N norm. $h = \frac{1}{512}$.

solution time $t_f = 0.0256$, the equivalent of 64 time steps at the finest resolution ($h_f = \frac{1.0}{512}$). The boundary conditions are no slip. The solution error tests are given in tables 1 for two dimensions and 1 for three dimensions. We show second order convergence in L_1.

N	Variable	$E_c = A(u_{2h}) - u_{4h}$	$E_f = A(u_h) - u_{2h}$	p
∞	v_x	3.560051e-01	1.560253e-01	1.19
∞	v_y	2.068663e-01	8.783526e-02	1.23
∞	v_z	2.068911e-01	8.783473e-02	1.23
∞	p	5.603254e-01	1.759296e-01	1.67
1	v_x	2.594371e-04	5.192670e-05	2.32
1	v_y	9.378651e-05	1.686378e-05	2.47
1	v_z	9.375695e-05	1.686981e-05	2.47
1	p	1.359413e-04	2.701042e-05	2.33
2	v_x	1.449043e-03	4.341411e-04	1.73
2	v_y	4.765399e-04	1.257778e-04	1.92
2	v_z	4.761483e-04	1.258223e-04	1.92
2	p	4.539508e-04	1.214530e-04	1.90

Table 2
Solution error in three dimensions. Convergence rates are calculated using L_N norm. $h = \frac{1}{512}$.

4.2 External Flows

To demonstrate that the algorithm is robust for long runs with geometry, we present two external flows.

In 11, we show the velocity field for two-dimensional flow over a cylinder with $Re = 500$. The initial velocity was uniform inflow with a small sinusoidal perturbation.

$$v_x = 1 v_y = 0.1 sin(2\pi x)$$

The boundary conditions at the top and bottom are solid wall. The boundary conditions at the left is uniform inflow and outflow on the right. The results are consistent with a von Karman vortex street.

In 12, we show the velocity field for three-dimensional flow over a sphere with $Re = 100$. The initial velocity was uniform inflow with a small sinusoidal perturbation.

$$v_x = 1 v_z = v_y = 0.1 sin(2\pi x)$$

The boundary conditions at the top and bottom are solid wall. The boundary conditions at the left is uniform inflow and outflow on the right. The flow soon stabilizes

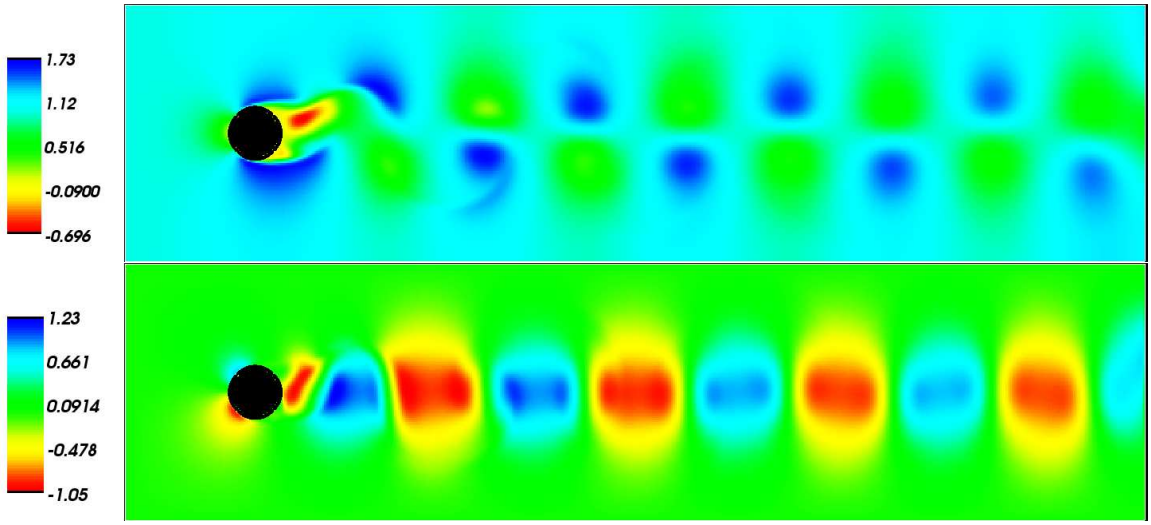

Fig. 11. Embedded boundary calculation of two-dimensional flow over a circular cylinder. The upper figure is the axial velocity, the lower is the transverse velocity. The grid resolution is 512×128. The Reynolds number based upon the diameter is 500. The velocity at inflow is 1.0. There have been 12000 time steps taken and t=25.56.

into the field shown and the results are consitent with those shown in Van Dyke [16].

5 Conclusions

We have developed an algorithm for solving the incompressible Navier-Stokes equations in the presence of embedded boundaries. We have demonstrated that the algorithm is second-order in $L - 1$ with refinement in space and time. We have demonstrated that the algorithm is robust for long runs in complex geometries.

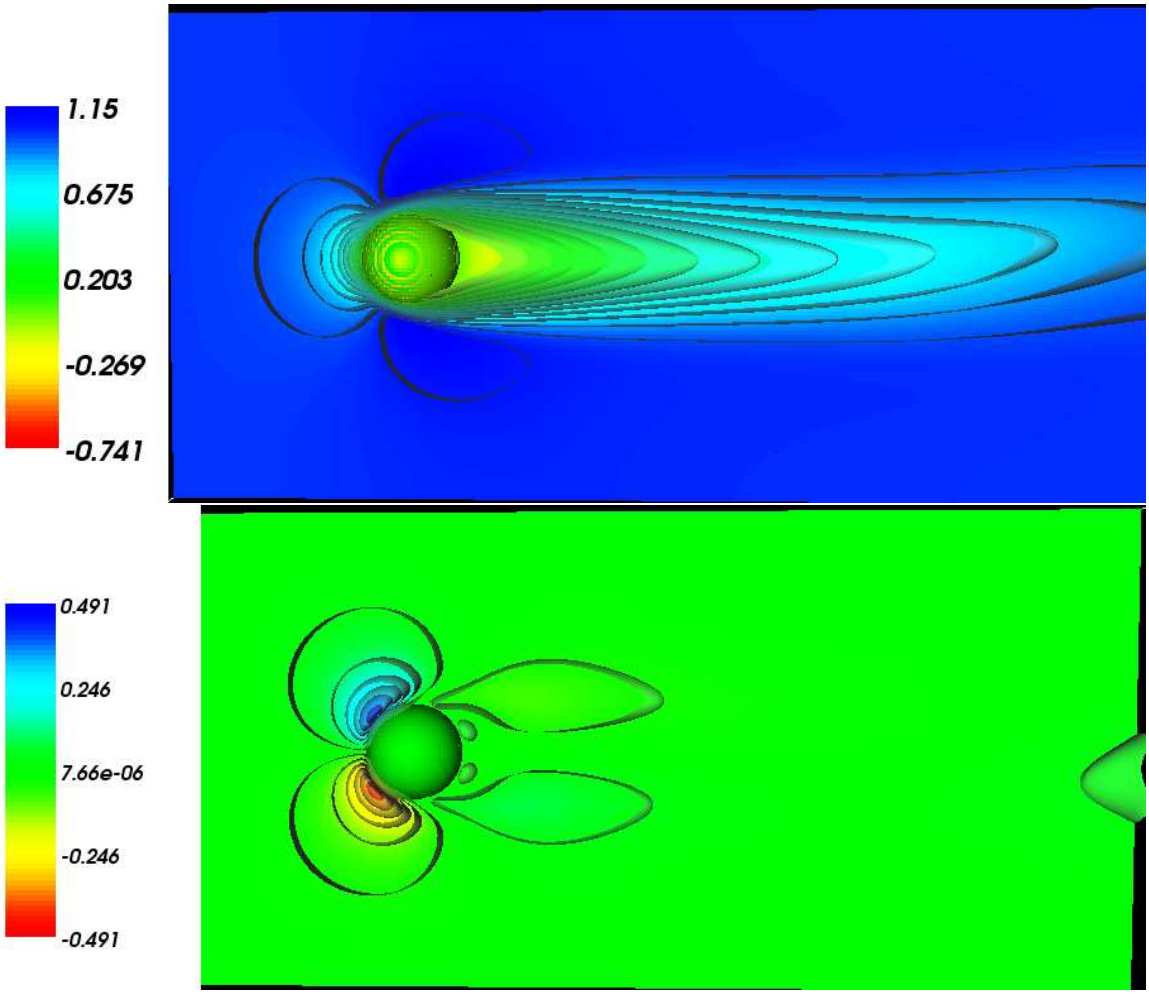

Fig. 12. Embedded boundary calculation of three-dimensional flow over a sphere. The upper figure is the axial velocity, the lower one the transverse velocity. The grid resolution is $256 \times 128 \times 128$. The Reynolds number based upon the diameter is 100. The velocity at inflow is 1.0. There have been 600 time steps taken and t=1.96.

References

[1] A. S. Almgren, J. B. Bell, and W. G. Szymczak. A numerical method for the incompressible Navier-Stokes equations based on an approximate projection. *SIAM J. on Sci. Comp.*, 17:358–369, 1996.

[2] Ann S. Almgren, John B. Bell, Phillip Colella, and Tyler Marthaler. A cell-centered Cartesian grid projection method for the incompressible Euler equations in complex geometries. In *Proceedingsof the AIAA 12th Computational Fluid Dynamics Conference,*, San Diego, California, June 1995.

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

[4] J. B. Bell, P. Colella, and L. H. Howell. An efficient second order projection method for viscous incompressible flow. In *AIAA 10th Comp. Fluid Dynamics Conf.*, pages 360–367, 1991.

[5] J.B. Bell, P. Colella, and M.L. Welcome. Conservative front-tracking for inviscid compressible flow. In *AIAA 10th Computational Fluid Dynamics Conference. Honolulu*, pages 814–822, 1991.

[6] M. Berger, C. Helzel, and R. LeVeque. H-box methods for the approximation of hyperbolic conservation laws on irregular grids. *SIAM Journal of Numerical Analysis*, 41:893–918, 2003.

[7] M. J. Berger and R. J. Leveque. Stable boundary conditions for Cartesian grid calculations. Technical Report 90-37, ICASE, May 1990.

[8] D. Calhoun. A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions. *J. Comput. Phys.*, 176(2):231–275, 2002.

[9] C.Helzel, M.J.Berger, and R.J.LeVeque. A high–resolution rotated grid method for conservation laws with embedded geometries. *SIAM J. Sci. Stat. Comput.*, 2005.

[10] I. L. Chern and P. Colella. A conservative front-tracking method for hyperbolic conservation laws. Technical Report UCRL-97200, Lawrence Livermore National Laboratory, 1987.

[11] A. J. Chorin. Numerical solutions of the Navier-Stokes equations. *Math. Comp.*, 22:745–762, 1968.

[12] A. J. Chorin. On the convergence of discrete approximations to the Navier-Stokes equations. *Math. Comp.*, 23:341–353, 1969.

[13] W. J. Coirier and K. G. Powell. An assessment of Cartesian-mesh approaches for the euler equations. *Journal of Computational Physics*, 117:121–131, 1995.

[14] P. Colella, D. T. Graves, B. Keen, and D. Modiano. A Cartesian grid embedded boundary method for hyperbolic conservation laws. *J. Comput. Phys.*, 211(1):347–366, 2006.

[15] P. Colella, T. Ligocki, and P. Schwartz. Using the divergence theorem for geometry generation. unpublished.

[16] Milton Van Dyke. *An Album of Fluid Motion.* Parabolic Press, Stanford, CA, 1982.

[17] A. Gilmanov and F. Sotiropoulos. A hybrid Cartesian/immersed boundary method for simulating flows with 3d, geometrically complex moving bodies. *J. Comput. Phys.*, 207(2):457–492, 2005.

[18] Louis H. Howell and John B. Bell. An adaptive-mesh projection method for viscous incompressible flow. To appear, SIAM Journal of Scientific Computing.

[19] H. S. Johansen and P. Colella. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *J. Comput. Phys.*, 147(2):60–85, December 1998.

[20] Hans Svend Johansen. *Cartesian Grid Embedded Boundary Methods for Elliptic and Parabolic Partial Differential Equations on Irregular Domains.* PhD thesis, Dept. of Mechanical Engineering, Univ. of California, Berkeley, December 1997.

[21] Benjamin Keen and Smadar Karni. A second order kinetic scheme for gas dynamics on arbitrary grids. *J. Comput. Phys.*, 2005.

[22] M. F. Lai and P. Colella. An approximate projection method for the incompressible Navier-Stokes equations. unpublished.

[23] D Martin and P Colella. A cell-centered adaptive projection method for the incompressible Euler equations. *J. Comput. Phys.*, 2000.

[24] P. McCorquodale, P. Colella, and H. Johansen. A Cartesian grid embedded boundary method for the heat equation on irregular domains. *J. Comput. Phys.*, 173(2):620–635, November 2001.

[25] P. McCorquodale, P. Colella, and H. Johansen. A Cartesian grid embedded boundary method for the heat equation on irregular domains. *J. Comput. Phys.*, 173:620–635, November 2001.

[26] D. Modiano and P. Colella. A higher-order embedded boundary method for time-dependent simulation of hyperbolic conservation laws. In *ASME 2000 Fluids Engineering Division Summer Meeting*, 2000.

[27] R. B. Pember, J. B. Bell, P. Colella, W. Y. Crutchfield, and M. L. Welcome. An adaptive Cartesian grid method for unsteady compressible flow in irregular regions. *J. Comput. Phys.*, 120(2):278–304, September 1995.

[28] S. Popinet. Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries. *J. Comput. Phys.*, 190(2):572–600, 2003.

[29] J. J. Quirk. An alternative to unstructured grids for computing gas dynamics flows around arbitrarily complex two-dimensional bodies. *Computers and Fluids*, 23:125–142, 1994.

[30] H. Liu S. Marella, S. Krishnan and H.S. Udaykumar. Sharp interface Cartesian grid method I: An easily implemented technique for 3d moving boundary computations. *J. Comput. Phys.*, 210(1):1–31, 2005.

[31] P. Schwartz, M. Barad, P. Colella, and T. Ligocki. A Cartesian grid embedded boundary method for the heat equation and poisson's equation in three dimensions. *J. Comput. Phys.*, 211(2):531–550, 2006.

[32] D. Trebotich and P. Colella. A projection method for incompressible viscous flow on moving quadrilateral grids. *J. Comput. Phys.*, (166):191–217, 2001.

[33] E.H. Twizell, A.B. Gumel, and M.A. Arigu. Second-order, l_0-stable methods for the heat equation with time-dependent boundary conditions. *Advances in Computational Mathmatics*, 6:333–352, 1996.