Title
A fourth-order Cartesian grid embedded boundary method for Poisson's equation

Journal
Communications in Applied Mathematics and Computational Science, 12(1)

ISSN
1559-3940

Authors
Devendran, Dharshi
Graves, Daniel
Johansen, Hans
et al.

Publication Date
2017

DOI
10.2140/camcos.2017.12.51

Peer reviewed

A Fourth Order Cartesian Grid Embedded Boundary Method for Poisson's Equation

Dharshi Devendran † Daniel T. Graves † Hans Johansen †
Terry Ligocki †

March 17, 2016

Abstract

In this paper, we present an fourth order algorithm to solve Poisson's equation in two and three dimensions. We use a Cartesian grid, embedded boundary method to resolve complex boundaries. We use a weighted least squares algorithm to solve for our stencils. We use convergence tests to demonstrate accuracy and we show the eigenvalues of the operator to demonstrate stability. We compare accuracy and performance with an established second order algorithm. We also discuss in depth strategies for retaining higher order accuracy in the presence of non-smooth geometries.

1 Prior Art

There are many numerical approaches to solve Poisson's equation in complex geometries. Green function approaches [25, 16, 8], such as the fast multipole method, are fast and near-optimal in complexity, but they are not conservative. Also, they cannot be easily extended to variable and tensor coefficient Poisson operators, which are important in the earth sciences and multi-material problems.

Another popular approach is to use the finite element method, which has a number of advantages. These advantages include negative-definite discrete operators, higher-order accuracy, and ease of extension to variable coefficients. The conditioning and accuracy of the discrete finite element operator can be strongly mesh-dependent, however [6]. Unfortunately, generating meshes with higher-order conforming elements for complex 3D domains is still an expensive, globally-coupled computation, and an open area of research [29].

This motivates the need for simpler grid generation. Cut cells are a simple way of addressing this. In a cut cell (or embedded boundary) method,

†Lawrence Berkeley National Laboratory, Berkeley, CA. Research at LBNL was supported financially by the Office of Advanced Scientific Computing Research of the US Department of Energy under contract number DE-AC02-05CH11231.

the discrete domain is the intersection of the complex geometry with a regular Cartesian grid. Such intersections are local, and can be calculated very efficiently in parallel, enabling fast computation of solution-dependent moving boundaries [1, 32, 26]. Cut cells have been used successfully to solve Poisson's equation in finite volume [19, 31] and finite difference [14, 23] discretizations.

For many problems, such as heat and mass transfer, discrete conservation is important. Finite volume methods are discretely conservative by construction because they are in discrete flux-divergence form [22]. Previous finite volume methods for Poisson's equation are first order in truncation error near the embedded boundary and second order in solution error [19, 31]. Our finite volume discretization of Poisson's equation is third order in truncation error and fourth order in solution error. The discretization is in flux-divergence form and therefore strongly conservative.

The second order, finite volume, strongly conservative Schwartz, et al. algorithm [31] has been used in many larger applications, including incompressible Navier Stokes with moving boundaries [26], compressible Navier Stokes [15] and a DNA-transport application [37]. We compare our algorithm to the Schwartz, et al. algorithm by comparing both eigenvalue spectrums and the how many degrees of freedom are required to achieve a given degree of accuracy. We also provide a strategy for maintaining higher-order accuracy in the presence of geometric discontinuities.

2 Underlying Analysis

Given a charge density ρ, Poisson's equation can be written as

$$\nabla \cdot (\nabla \phi) = \rho \tag{1}$$

for the potential ϕ. If we integrate this equation over a control volume V and apply the divergence theorem, this becomes

$$\int_{\partial V} \nabla \phi \cdot \hat{n} \;\; dA = \int_V \rho \;\; dV \tag{2}$$

where $\hat{n}$ is the outward-facing unit normal to the surface. Our volumes are Cartesian cells cut by an embedded boundary.

Formally, the underlying description of space is given by rectangular control volumes on a Cartesian mesh $\Upsilon_{\mathbf{i}} = [(\mathbf{i} - \frac{1}{2}\mathbf{u})h, (\mathbf{i} + \frac{1}{2}\mathbf{u})h], \mathbf{i} \in \mathbf{Z}^{\mathcal{D}}$, where $\mathcal{D}$ is the dimensionality of the problem, h is the mesh spacing, and $\mathbf{u}$ is the vector whose entries are all one (note we use bold font $\mathbf{u} = (u_1, \ldots, u_d, \ldots, u_D)$ to indicate a vector quantity). Given an irregular domain Ω, we obtain control volumes $V_{\mathbf{i}} = \Upsilon_{\mathbf{i}} \bigcap \Omega$ and faces $A_{\mathbf{i} \pm \frac{1}{2}\mathbf{e}_d}$ which are the intersection of the boundary of $\partial V_{\mathbf{i}}$ with the coordinate planes $\{\mathbf{x} : x_d = (i_d \pm \frac{1}{2})h\}$ ($\mathbf{e}_d$ is the unit vector in the d direction). We also define $A_{B,\mathbf{i}}$ to be the intersection of the boundary of the irregular domain with the Cartesian control volume: $A_{B,\mathbf{i}} = \partial\Omega \bigcap \Upsilon_{\mathbf{i}}$. Figure 1 illustrates a volume cut by an embedded boundary.

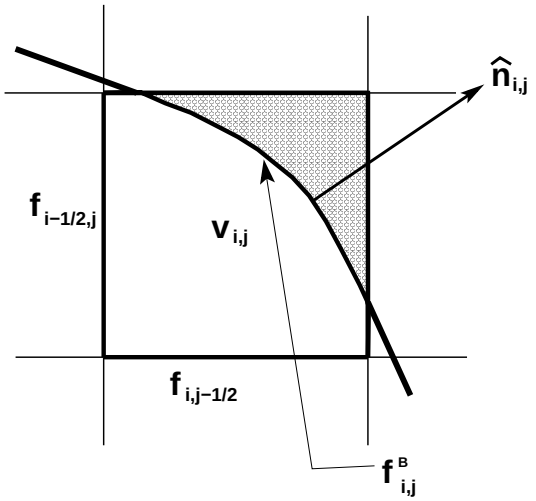

Figure 1: Illustration of cut cell notation. The shaded region is outside the solution domain. The volume $V_{\mathbf{i}}$ is connected to other volumes via the faces aligned with the coordinate planes. The EB face is formed by the intersection of the embedded boundary and the cell.

Throughout this paper, we use the following compact "multi-index" notation:

$$(\mathbf{x}-\bar{\mathbf{x}})^{\mathbf{p}} = \prod_{d=1}^{\mathcal{D}} (x_d - \bar{x}_d)^{p_d}$$

$$\mathbf{p}! = \prod_{d=1}^{\mathcal{D}} p_d!$$

Given a point in space $\bar{\mathbf{x}}$, and a $\mathcal{D}$-dimensional integer vector $\mathbf{p}$, we define $m_{\mathbf{i}}^{\mathbf{p}}(\bar{\mathbf{x}})$ to be the $\mathbf{p}^{th}$ moment of the volume $V_{\mathbf{i}}$ relative to the point $\bar{\mathbf{x}}$.

$$m_{\mathbf{i}}^{\mathbf{p}}(\bar{\mathbf{x}}) = \int_{V_{\mathbf{i}}} (\mathbf{x}-\bar{\mathbf{x}})^{\mathbf{p}} dV \tag{3}$$

The volume of the cut cell $|V_{\mathbf{i}}| = m_{\mathbf{i}}^{\mathbf{z}}$, where $\mathbf{z}$ is the zero vector. We define the face moments $m^{\mathbf{p}}_{\mathbf{i}\pm\frac{1}{2}\mathbf{e}_d}(\bar{\mathbf{x}})$ to be the $\mathbf{p}^{th}$ moments (relative to the point $\bar{\mathbf{x}}$) of the faces $A_{\mathbf{i}\pm\frac{1}{2}\mathbf{e}_d}$.

$$m^{\mathbf{p}}_{\mathbf{i}\pm\frac{1}{2}\mathbf{e}_d}(\bar{\mathbf{x}}) = \int_{A_{\mathbf{i}\pm\frac{1}{2}\mathbf{e}_d}} (\mathbf{x}-\bar{\mathbf{x}})^{\mathbf{p}} dA \tag{4}$$

We define two moments corresponding to the embedded boundary face $A_{B,\mathbf{i}}$: $m^{\mathbf{p}}_{B,\mathbf{i}}$ and $m^{\mathbf{p}}_{B,\mathbf{i},d}$:

$$m^{\mathbf{p}}_{B,\mathbf{i}}(\bar{\mathbf{x}}) = \int_{A_{B,\mathbf{i}}} (\mathbf{x}-\bar{\mathbf{x}})^{\mathbf{p}}\, dA \tag{5}$$

and

$$m^{\mathbf{p}}_{B,\mathbf{i},d}(\bar{\mathbf{x}}) = \int_{A_{B,\mathbf{i}}} (\mathbf{x}-\bar{\mathbf{x}})^{\mathbf{p}}\, \hat{n}_d(\mathbf{x})\, dA, \tag{6}$$

where $\hat{n}_d$ is the d^{th} component of the outward-facing unit normal to the EB face (note that $\hat{n}_d = \hat{n}_d(\mathbf{x})$).

We define flux function to be the gradient of the potential ($\mathbf{F} \equiv \nabla\phi$). Given a volume $V_\mathbf{i}$ we can rewrite the integral form of Poisson's equation (2) as a sum of integrals over each face in the volume.

$$\int_{V_\mathbf{i}} \nabla\cdot(\nabla\phi) \ dV = \sum_{d=1}^{\mathcal{D}} \left(\int_{A_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}} F_d \ dA - \int_{A_{\mathbf{i}-\frac{1}{2}\mathbf{e}_d}} F_d \ dA + \int_{A_{B,\mathbf{i}}} F_d \ \hat{n}_d \ dA \right), \quad (7)$$

We use the following notation to denote the averages of ϕ over a computational volume:

$$\langle\phi\rangle_\mathbf{i} = \frac{1}{|V_\mathbf{i}|} \int_{V_\mathbf{i}} \phi \ dV \quad (8)$$

The average flux over a coordinate face is defined as

$$\langle F_d\rangle_{\mathbf{i}\pm\frac{1}{2}\mathbf{e}_d} = \frac{1}{|A_{\mathbf{i}\pm\frac{1}{2}\mathbf{e}_d}|} \int_{A_{\mathbf{i}\pm\frac{1}{2}\mathbf{e}_d}} F_d \ dA,$$

and the average flux at the irregular face is given by

$$\langle F_d \ \hat{n}_d\rangle_{B,\mathbf{i}} = \frac{1}{|A_{B,\mathbf{i}}|} \int_{A_{B,\mathbf{i}}} F_d\hat{n}_d \ dA.$$

To create a conservative divergence operator, we discretize our divergence operator as a sum of average fluxes. We define the volume fraction κ to be the fraction of the volume of the cell inside the solution domain, so that

$$\kappa = h^{-\mathcal{D}}|V_\mathbf{i}| = h^{-\mathcal{D}} m_\mathbf{i}^\mathbf{z} \quad . \quad (9)$$

Given a flux function $\mathbf{F}$, the κ-weighted divergence of the flux is defined to be the volume average of the divergence multiplied by κ:

$$\begin{aligned} \kappa L(\phi)_\mathbf{i} = \kappa \langle\nabla\cdot\mathbf{F}\rangle_\mathbf{i} &= \frac{1}{h^\mathcal{D}} \int_{V_\mathbf{i}} \nabla\cdot\mathbf{F} dV \\ &= \frac{1}{h^\mathcal{D}} \sum_{d=1}^{\mathcal{D}} \left(|A_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}| \langle F_d\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d} - |A_{\mathbf{i}-\frac{1}{2}\mathbf{e}_d}| \langle F_d\rangle_{\mathbf{i}-\frac{1}{2}\mathbf{e}_d} + |A_{B,\mathbf{i}}| \langle F_d \ \hat{n}_d\rangle_{B,\mathbf{i}} \right) \end{aligned} \quad (10)$$

We weigh the conservative divergence this way to avoid small-κ numerical instabilities. Implicit algorithms for Poisson's equation (1) solve the discrete system

$$\kappa \langle\nabla\cdot\nabla\phi\rangle_\mathbf{i} = \kappa \langle\rho\rangle_\mathbf{i} \quad (11)$$

for ϕ [19, 31], which avoids very large negative eigenvalues from terms with κ^{-1}. Up to this point, no approximations have been made.

The accuracy of the method is dependent only upon the accuracy of the discretization of the average fluxes. Previous conservative algorithms for embedded boundaries compute fluxes that are second order [28, 26, 31, 13, 15, 27, 9]. In those algorithms, the cell-averages of ϕ are approximated to second order by pointwise values at the centroids of faces, and fluxes are constructed by differencing those pointwise values. For fourth-order accurate fluxes, we need to use the cell-averages of ϕ directly in the local polynomial expansion of ϕ.

For some integer Q, suppose we want an $O(h^Q)$ approxmation to the flux $\mathbf{F} = \nabla\phi$. Given a sufficiently smooth function ϕ, we can approximate ϕ in the neighborhood of $\bar{\mathbf{x}}$ using a multi-dimensional Taylor expansion:

$$\phi(\mathbf{x}) = \sum_{|\mathbf{q}|<=Q} \frac{1}{\mathbf{q}!}\phi^{(\mathbf{q})}(\bar{\mathbf{x}})\,(\mathbf{x}-\bar{\mathbf{x}})^{\mathbf{q}} + O(h^{Q+1}), \tag{12}$$

Here we use the the multi-index partial derivative notation

$$\phi^{(\mathbf{q})} = \partial^{\mathbf{q}}\phi = \frac{\partial^{q_1}}{\partial x_1^{q_1}} \cdots \frac{\partial^{q_\mathcal{D}}}{\partial x_\mathcal{D}^{q_\mathcal{D}}}\phi\,. \tag{13}$$

If we put the expansion (12) into one of the integrals in (7) over a coordinate-aligned face, we get

$$\int\limits_{A_{\mathbf{i}\pm\frac{1}{2}\mathbf{e}_d}} \frac{\partial\phi}{\partial x_d}\; dA = \sum_{|\mathbf{q}|<=Q} q_d c_{\mathbf{q}} \int_{A_{\mathbf{i}\pm\frac{1}{2}\mathbf{e}_d}} (\mathbf{x}-\bar{\mathbf{x}})^{\mathbf{q}-\mathbf{e}_d}\; dA + O(h^Q) \tag{14}$$

$$= \sum_{|\mathbf{q}|<=Q} q_d c_{\mathbf{q}} m^{\mathbf{q}-\mathbf{e}_d}_{\mathbf{i}\pm\frac{1}{2}\mathbf{e}_d}(\bar{\mathbf{x}}) + O(h^Q) \tag{15}$$

where $c_q = \frac{1}{\mathbf{q}!}\phi^{(\mathbf{q})}(\bar{\mathbf{x}})$. The flux equation at the irregular boundary becomes

$$\int\limits_{A_{B,\mathbf{i}}} \frac{\partial\phi}{\partial x_d}\hat{n}_d\; dA = \sum_{|\mathbf{q}|<=Q} q_d c_{\mathbf{q}} \int_{A_{B,\mathbf{i}}} (\mathbf{x}-\bar{\mathbf{x}})^{\mathbf{q}-\mathbf{e}_d}\hat{n}_d\; dA + O(h^Q) \tag{16}$$

$$= \sum_{|\mathbf{q}|<=Q} q_d c_{\mathbf{q}} m^{\mathbf{q}-\mathbf{e}_d}_{B,\mathbf{i},d}(\bar{\mathbf{x}}) + O(h^Q) \tag{17}$$

All the moments can be generated to any order as shown in Schwartz, et al. [32]. Therefore, generating an $O(h^Q)$ algorithm for Poisson's equation reduces to finding the coefficients $c_{\mathbf{q}}$.

3 Algorithm

3.1 Basic Methodology

We define $\mathcal{N}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}$ to be the set of volumes in the neighborhood of face $\mathbf{i}+\frac{1}{2}\mathbf{e}_d$ (our neighborhood algorithm is described in §3.4). We put the expansion (12)

into (8) for every volume $V_{\mathbf{j}} \in \mathcal{N}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}$:

$$\langle \phi \rangle_{\mathbf{j}} = \frac{1}{|V_{\mathbf{j}}|} \sum_{|\mathbf{q}|<=Q} c_{\mathbf{q}} \int_{V_{\mathbf{j}}} (\mathbf{x} - \bar{\mathbf{x}}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d})^{\mathbf{q}} \; dV \tag{18}$$

$$= \frac{1}{|V_{\mathbf{j}}|} \sum_{|\mathbf{q}|<=Q} c_{\mathbf{q}} m_{\mathbf{j}}^{\mathbf{q}} (\bar{\mathbf{x}}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}), \tag{19}$$

where $\bar{\mathbf{x}}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d} = h(\mathbf{i} + \frac{1}{2}\mathbf{e}_d)$ is the center of the target face. This forms a system of equations for the coefficients $c_{\mathbf{q}}$.

Define C to be a column vector composed of the Taylor coefficients $c_{\mathbf{q}}$. In C, the powers of $\mathbf{q}$ are listed in lexicographical order. For example, in 2D, for $Q = 2$

$$C = \begin{bmatrix} c^{(0,0)} \\ c^{(1,0)} \\ c^{(2,0)} \\ c^{(0,1)} \\ c^{(1,1)} \\ c^{(0,2)} \end{bmatrix}. \tag{20}$$

Define Φ to be the column vector of all $\langle \phi \rangle_{\mathbf{j}}$ such that $V_{\mathbf{j}} \in \mathcal{N}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}$. Define M to be the matrix of the neighbors' volume moments normalized by their volumes. Each row of M corresponds to a particular neighbor. In particular, suppose $\mathcal{N}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d} = \{V_{j_1}, \ldots, V_{j_N}\}$. Then the l^{th} row of M are the moments for neighbor V_{j_l}. Each column of M corresponds to a particular power $\mathbf{q}$. Because the volume of $V_{\mathbf{j}}$ is $m_{\mathbf{j}}^{\mathbf{z}}$, the first column is made of ones.

For example, in 2D with $Q = 2$, the moment matrix M takes the form

$$M = \begin{bmatrix} 1 & \frac{m_{j_1}^{(1,0)}}{m_{j_1}^{(0,0)}} & \frac{m_{j_1}^{(2,0)}}{m_{j_1}^{(0,0)}} & \frac{m_{j_1}^{(0,1)}}{m_{j_1}^{(0,0)}} & \frac{m_{j_1}^{(1,1)}}{m_{j_1}^{(0,0)}} & \frac{m_{j_1}^{(0,2)}}{m_{j_1}^{(0,0)}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \frac{m_{j_N}^{(1,0)}}{m_{j_N}^{(0,0)}} & \frac{m_{j_N}^{(2,0)}}{m_{j_N}^{(0,0)}} & \frac{m_{j_N}^{(0,1)}}{m_{j_N}^{(0,0)}} & \frac{m_{j_N}^{(1,1)}}{m_{j_N}^{(0,0)}} & \frac{m_{j_N}^{(0,2)}}{m_{j_N}^{(0,0)}} \end{bmatrix}. \tag{21}$$

Extending both C and M to $Q = 4$ simply requires adding the extra moments in Pascal's triangle in lexicographical order. All moments in the system of equations are cented around the target face at $\bar{\mathbf{x}}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d} = h(\mathbf{i} + \frac{1}{2}\mathbf{e}_d)$. The system of equations formed by (8) over the neighborhood $\mathcal{N}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}$ takes the form

$$\Phi = MC. \tag{22}$$

Say there are P coeffcients we need and N neighbors in $\mathcal{N}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}$. If $N > P$, we have an overdetermined system that we can solve by weighted least squares. We define a weighting matrix W and use it to multiply both sides of our system

$$W\Phi = WMC$$

The choice of weighting matrix is discussed in §3.3. Taking the Moore-Penrose pseudoinverse, we solve for the Taylor coeffcients

$$C = (WM)^{\dagger} W\Phi$$

and use these coefficients to compute the flux at the face. Recall from (15) that we need to shift and transform the coefficients to compute the average gradient at the face. Define G to be the row vector $G = [\ldots q_d m^{\mathbf{q}-\mathbf{e}^d}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d} \ldots]$ where $|\mathbf{q}| <= Q$. Equation 15 becomes

$$|A_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}|\langle F_d\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d} = GC$$

We express this flux calculation as a stencil. Because these are all linear operators, we know we can express the flux as a column vector $S_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}$ acting on the solution, $|A_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}|\langle F_d\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d} = S^T_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}\Phi$ where

$$S_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d} = G(WM)^{\dagger}W \tag{23}$$

At every face in the domain, we solve for the stencil $S_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}$. For faces near the domain boundaries and the embedded boundary we add boundary equations to the system (22). This is discussed in §3.2. We solve for our stencils using the singular value decompostion framework from LAPACK [2].

Putting the flux stencils from (23) into (10), we get the stencil for our operator κL:

$$\kappa L_{\mathbf{i}} = \frac{1}{h^{\mathcal{D}}} \sum_{d=1}^{\mathcal{D}} \left(S_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d} - S_{\mathbf{i}-\frac{1}{2}\mathbf{e}_d} + S^{EB}_{\mathbf{i}} \right), \tag{24}$$

where $S^{EB}_{\mathbf{i}}$, the stencil for the embedded boundary flux, is discussed in §3.2.

Once our stencils are calculated, we use the Chombo infrastructure [10, 11], which uses the Martin and Cartwright multigrid algorithm [24], to solve the system. The bottom solver for our multigrid algorithm is the PETSc algerbraic multigrid solver [5, 3, 4]. For eigenvalue calculations, we use the SLEPc infrastructure [18, 17, 7]. For details of our adaptive multigrid algorithm, see Devendran, et al. [12].

3.2 Boundary Conditions

Boundary conditions for this algorithm are used in two ways. First, we need to calculate the fluxes at the boundary to complete our finite volume discretization (see (24)). Second, to improve the stability of the operator, we include boundary condition equations in the system (22) that we solve to obtain the polynomial expansion.

To calculate boundary fluxes, we need different procedures for different types of boundary conditions. For Neumann boundary conditions, the flux is the specified boundary condition. For Dirichlet boundary conditions, on the other hand, we need to compute a stencil to calculate the flux. For Dirichlet domain

boundary faces, we solve for the flux stencil as we would for any other face. For Dirichlet boundary conditions on embedded boundary faces, we follow the same procedure except that we use the polynomial expansion from (17) where the derivatives of the normal to the boundary are included.

To improve the stability of the operator, we add equations that contain boundary condition information to the system (19) used to compute polynomial coefficients. Suppose a volume $V_{\mathbf{j}}$ in the neighbor set $\mathcal{N}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}$ contains a domain face $\mathbf{j}+\frac{1}{2}\mathbf{e}_d$ which has a Dirichlet boundary condition $\langle\phi\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}_d} = \phi_{DB}$. We add the equation

$$\phi_{DB} = \frac{1}{|A_{\mathbf{j}+\frac{1}{2}\mathbf{e}_d}|}\int_{A_{\mathbf{j}+\frac{1}{2}\mathbf{e}_d}} \phi \; dA \tag{25}$$

$$= \frac{1}{|A_{\mathbf{j}+\frac{1}{2}\mathbf{e}_d}|}\sum_{|\mathbf{q}|<=Q} c_{\mathbf{q}} \int_{A_{\mathbf{j}+\frac{1}{2}\mathbf{e}_d}} (\mathbf{x}-\bar{\mathbf{x}}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d})^{\mathbf{q}} \; dA \tag{26}$$

$$= \frac{1}{|A_{\mathbf{j}+\frac{1}{2}\mathbf{e}_d}|}\sum_{|\mathbf{q}|<=Q} c_{\mathbf{q}} m^{\mathbf{q}}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}(\bar{\mathbf{x}}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d}) \tag{27}$$

to the system (19). If $V_{\mathbf{j}}$ contains an embedded boundary face with a Diriechlet boundary condtion $\langle\phi\rangle^{EB}_{\mathbf{i}} = \phi_{EB}$ we add the appropriate form of (17):

$$\phi_{EB} = \frac{1}{|A_{B_{\mathbf{i}}}|}\sum_{|\mathbf{q}|<=Q} c_{\mathbf{q}} \int_{A_{B_{\mathbf{i}}}} (\mathbf{x}-\bar{\mathbf{x}})^{\mathbf{q}} n_d \; dA \tag{28}$$

$$= \frac{1}{|A_{B_{\mathbf{i}}}|}\sum_{|\mathbf{q}|<=Q} c_{\mathbf{q}} m^{\mathbf{q}}_{B_{\mathbf{i}},d}(\bar{\mathbf{x}}) \tag{29}$$

to our equation set (19). The extension of this process to Neumann boundary conditions is straightforward.

3.3 Weighting Matrix

Using weighted least squares adds a great deal of flexibility to a least-squares system solver. We use a diagonal weighting matrix W in (23). Using a diagonal weighting matrix amounts to assigning relative importance to the various equations in the system; larger weights mean that equation will more heavily influence the solution to the system [35]. We have found that the choice of weighting function strongly influences the eigenvalues of the resulting operator, and thus its stability.

Potential theory tells us that the effect of a charge in a Poisson system should diminish quickly with distance [34]. To mimic this in our stencils, we want volumes closer to the target face to have a much larger weight than those further away. If the volume being weighted is $\mathbf{j}$ and the target face is $\mathbf{i}+\frac{1}{2}\mathbf{e}_d$, the weight value $W_{\mathbf{j},\mathbf{i}+\frac{1}{2}\mathbf{e}_d}$ is given by

$$W_{\mathbf{j},\mathbf{i}+\frac{1}{2}\mathbf{e}_d} = (D_{\mathbf{j},\mathbf{i}+\frac{1}{2}\mathbf{e}_d})^{-5}$$

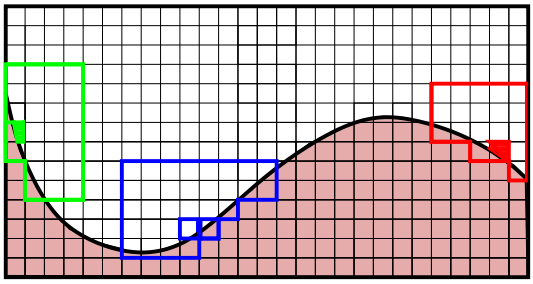

Figure 2: Neighbors of faces cut by the embedded boundary. Geometric constraints can greatly alter the number of neighbors available within a given radius

where $D_{\mathbf{j},\mathbf{i}+\frac{1}{2}\mathbf{e}_d}$ is the distance between the volume and the target face. Using this weighting, we find that our stencil values in the interior appear to be a perturbation off of a standard second-order stencil and our eigenvalues are stable.

3.4 Neighborhood Algorithm

We define the neighborhood of the face to be the the set of of valid cells within a discrete radius $R = 3$ cells of either cell of the face:

$$\mathcal{N}_{\mathbf{i}+\frac{1}{2}\mathbf{e}_d} = \{\mathbf{j} : i_d - j_d < R \text{ or } j_d - (i_d + 1) < R \text{ for any } 1 \leq d \leq \mathcal{D}\}. \tag{30}$$

We use this many cells because we need enough cells in the system (22) so that the system will be overdetermined even in the case where the embedded boundary cuts out half of the cells in the neighborhood. Figure 2 illustrates how the number of neighbor volumes can vary due to geometric constraints. We detect if there are are not enough cells for any given face and, for that face, we use a larger R.

4 Convergence Tests

To validate our algorithm, we present convergence tests to show that our algorithm is converging at expected rates. We test both truncation error T and solution error ϵ. We evaluate convergence using L_1, L_2, and L_∞ norms. Given an error field E defined on volumes $V_\mathbf{i}$ in Ω, and a norm operator $||E||$, the rate of convergence ϖ is defined as

$$\varpi = \log_2 \left(\frac{||E^{2h}||}{||E^h||} \right).$$

Given a computational domain Ω, we define the L_∞ norm of a field to be maximum value over that field while the L_1 and L_2 norms are integral norms.

These take the form

$$||E||_\infty = \max_{\mathbf{i}\in\Omega} |E_\mathbf{i}|$$
$$||E||_1 = \frac{1}{V_\omega}\int_\Omega |E_\mathbf{i}|dV = \frac{1}{V_\Omega}\sum_{\mathbf{i}\in\Omega} |E_\mathbf{i}|V_\mathbf{i}$$
$$||E||_2 = \left(\frac{1}{V_\omega}\int_\Omega |E_\mathbf{i}|^2 dV\right)^{\frac{1}{2}} = \left(\frac{1}{V_\Omega}\sum_{\mathbf{i}\in\Omega} |E_\mathbf{i}|^2 V_\mathbf{i}\right)^{\frac{1}{2}}$$

where V_Ω is the volume of the whole domain.

Given a smooth input potential ϕ^e, we compute the truncation error T by comparing the discrete operator L with the exact average Poisson operator L^e:

$$T = \kappa(L(\phi^e) - L^e(\phi^e)) \tag{31}$$

where $\kappa L(\phi)$ is given in (10) and

$$L^e(\phi^e)_\mathbf{i} = \int_\mathbf{i} \nabla\cdot(\nabla\phi^e)\ dV. \tag{32}$$

We weight the operator this way because the volume fraction κ can be arbitrarily small and because this is the form of the operator that is used in the solution process (see (11)). The solution error ϵ is given by comparing the computed solution ϕ to the exact solution ϕ^e:

$$\epsilon = \phi - \phi^e. \tag{33}$$

We expect the truncation error to be larger at the embedded boundary since the operator is formally third order in the cut cells. Potential theory tells us that these truncation errors at the boundary should be smoothed out in solution error. We therefore expect solution error to be uniformly fourth order in all norms.

For these tests, we need a smooth geometry and preferably one whose curvature varies. Our computational domain is the unit cube. Given a center point $\mathbf{x}_0$, we use the exterior of an ellipse of the form

$$\sum_{d=1}^{\mathcal{D}} \frac{x_d^2 - x_{0,d}^2}{r_d^2} = 0. \tag{34}$$

where $\mathbf{r} = (0.25, 0.5, 0.75)$ and $\mathbf{x}_0 = (0.5, 0.5, 0.5)$. A picture of this ellipse is given in figure 7. We generate our geometric moments to $O(h^6)$ so that our results would only reflect the accuracy of our Poisson discretization. Our finest grid spacing in these tests is $128^\mathcal{D}$ ($h = 1/128$). The exact potential field for these tests is given by

$$\phi^e = \prod_{d=1}^{\mathcal{D}} cos(\pi\mathbf{x}_d). \tag{35}$$

4.1 Truncation Error

In Table 1, we present truncation error rates for the case where the domain has Neumann boundary condtions and the irregular boundary has Dirichlet boundary condtitions ($\phi|_{\partial\Omega} = \phi^e$). In Table 2, we present truncation error rates for the case where the domain has Dirichlet boundary conditions and the irregular boundary has Neumann boundary conditions ($\nabla\phi\cdot\hat{n} = \nabla\phi^e\cdot\hat{n}$). For the two examples, we present convergence rates for both two and three dimensions. The third-order truncation error at the embedded boundary dominates the error on the domain, and the L_∞ norm reflects this. The truncation error in the L_1 norm, on the other hand, is fourth-order because the embedded boundary only has codimension one.

$\mathcal{D}$	Norm	$\|\|\epsilon^{2h}\|\|$	ϖ	$\|\|\epsilon^{h}\|\|$
2	L_∞	1.290e-04	2.60	2.130e-05
2	L_1	5.336e-06	3.99	3.358e-07
2	L_2	1.200e-05	3.55	1.022e-06
3	L_∞	9.222e-04	3.86	6.334e-05
3	L_1	1.071e-05	4.00	6.687e-07
3	L_2	2.507e-05	3.66	1.984e-06

Table 1: Truncation error convergence rates with Dirichlet boundary conditions on the embedded boundary and Neumann boundary conditions on the domain. The geometry is the exterior of the ellipse shown in Figure 7 and $h = 1/128$.

$\mathcal{D}$	Norm	$\|\|\epsilon^{2h}\|\|$	ϖ	$\|\|\epsilon^{h}\|\|$
2	L_∞	1.979e-04	2.99	2.485e-05
2	L_1	1.423e-05	3.95	9.184e-07
2	L_2	3.897e-05	3.46	3.530e-06
3	L_∞	4.490e-04	2.22	9.645e-05
3	L_1	2.698e-05	3.95	1.747e-06
3	L_2	6.697e-05	3.44	6.161e-06

Table 2: Truncation error convergence rates with Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain. The geometry is the exterior of an ellipse shown in Figure 7 and $h = 1/128$.

4.2 Solution Error

Johansen [19, 20] shows that a method can have a lower-order truncation error on the embedded boundary (which is a codimension one smaller set) than in the interior and still maintain the proper order for solution error. We solve $\kappa L\phi = \kappa L(\phi^e)$ and compute the solution error. For this test ϕ^e is given by (35). We present solution error rates for the case where both the domain and the

irregular boundary have Dirichlet boundary condtitions ($\phi|_{\partial\Omega} = \phi_e$) in Table 3. We present solution error rates for the case where both the domain and the irregular boundary have Neumann boundary conditions ($\nabla\phi \cdot \hat{n} = \nabla\phi^e \cdot \hat{n}$) in Table 4. In both cases, we show uniform fourth order convergence rates in all norms. We also run this test at much higher resolutions in §6.1.

$\mathcal{D}$	Norm	$\|\|\epsilon^{2h}\|\|$	ϖ	$\|\|\epsilon^{h}\|\|$
2	L_∞	1.626e-07	3.94	1.060e-08
2	L_1	8.934e-08	3.91	5.952e-09
2	L_2	1.032e-07	3.93	6.783e-09
3	L_∞	3.060e-07	3.97	1.954e-08
3	L_1	1.955e-07	3.95	1.265e-08
3	L_2	2.154e-07	3.96	1.386e-08

Table 3: Solution error convergence rates with Dirichlet boundary conditions on the embedded boundary and Neumann boundary conditions on the domain. The geometry is the exterior of an ellipse and $h = 1/128$.

$\mathcal{D}$	Norm	$\|\|\epsilon^{2h}\|\|$	ϖ	$\|\|\epsilon^{h}\|\|$
2	L_∞	1.835e-07	3.96	1.176e-08
2	L_1	6.904e-08	3.95	4.459e-09
2	L_2	8.678e-08	3.96	5.558e-09
3	L_∞	3.879e-07	3.86	2.669e-08
3	L_1	9.325e-08	3.92	6.175e-09
3	L_2	1.315e-07	3.94	8.559e-09

Table 4: Solution error convergence rates with Neumann boundary conditions the embedded boundary and Dirichlet boundary conditions on the domain. The geometry is the exterior of an ellipse and $h = 1/128$.

5 Operator Eigenvalues

In this section, we compare the spectrum of our algorithm to the widely used, second-order algorithm presented by Schwartz, et al. [31].

The eigenvalues of the Poisson operator will depend upon the geometry and resolution of the problem as well as the operator boundry conditions. Due to limitations in computational resources, we are only able to show the spectrum for coarse two-dimensional problems (our resolution is 32^2). We use the Krylov-Schur module in SLEPc [18] to compute the eigenvalues.

We present the spectrum for our fourth-order operator with Dirichlet boundary conditions on the embedded boundary and Neumann boundary condtions on the domain in Figure 3. We present the spectrum for the second-order operator with identical conditions in Figure 5. We also present the fourth-order

spectrum for Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain in Figure 4 and the second order spectrum in Figure 6. In both cases, Dirichlet boundary conditions on the embedded boundary introduce more complex eigenvalues. In both cases, all the eigenvalues have negative real components and are therefore stable.

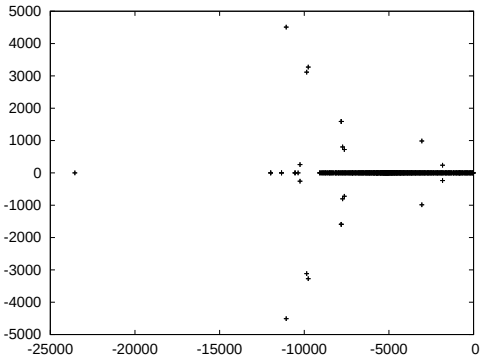

Figure 3: Eigenvalues for the fourth order cell-averaged two-dimensional Poisson operator with Dirichlet boundary conditions on the embedded boundary and Neumann boundary conditions on the domain boundary. The geometry implicit function is described by (34) and the resolution is 32^2.

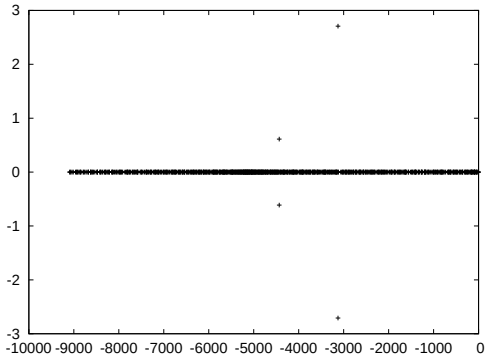

Figure 4: Eigenvalues for the fourth order cell-averaged two-dimensional Poisson operator with Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain boundary. The geometry implicit function is described by (34) and the resolution is 32^2.

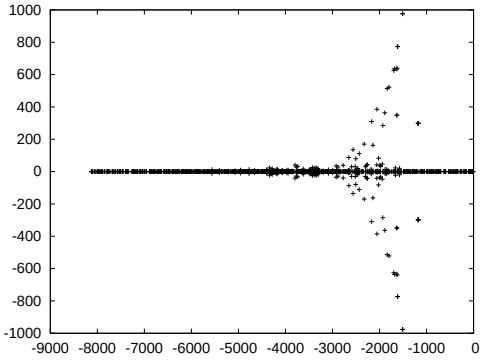

Figure 5: Eigenvalues for the two-dimensional, second-order Poisson operator (described in [31]) with Dirichlet boundary conditions on the embedded boundary and Neumann boundary conditions on the domain boundary. The geometry implicit function is described by (34) and the resolution is 32^2.

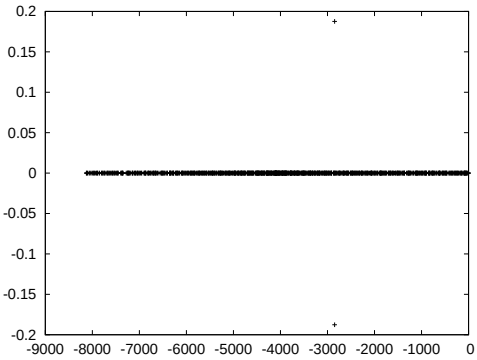

Figure 6: Eigenvalues for the two-dimensional, second-order Poisson operator (described in [31]) with Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain boundary. The geometry implicit function is described by (34) and the resolution is 32^2.

6 Effect on Accuracy of Geometric Differentiability

Fundamentally, the appeal of a higher-order method is that one can achieve higher accuracy with fewer degrees of freedom. To reliably achieve this rate of convergence, however, one needs a sufficiently smooth description of the geometry. To achieve $O(h^P)$ accurate fluxes, Schwartz, Percelay, et al. [32] show that all geometric moments in the calculation must also converge to $O(h^P)$.

Unfortunately, geometric descriptions are not always sufficiently smooth. This is not necessarily catastrophic. Johansen [20] shows that large truncation errors can be ameliorated under certain circumstances. Specifically, Johansen shows that $O(1)$ truncation errors at a Dirichlet boundary condition will not prevent second order solution error convergence. Similarly, $O(h)$ truncation errors at a Neumann boundary will not prevent second order solution error convergence.

These competing effects present a bit of a complex picture. To see how our algorithm fits into this picture, we compare our algorithm to the widely-used Schwartz, et al. [31] algorithm for Poisson's equation. We compare the two algorithms using both a smooth and a non-smooth geometric description. These comparisons are done for both Dirichlet and Neumann boundary conditions at the embedded boundary. Because some of the techniques used in this section are resource-intensive, we restrict our comparisons to two-dimensions so we can

achieve much higher resolutions.

All of these tests are done with an exact potential ϕ_e

$$\phi_e = \prod_{d=1}^{\mathcal{D}} sin(\pi x_d)$$

and an charge distribution $\rho = \nabla \cdot \nabla \phi_e$. The calculation domain is the unit square and there are Dirichlet boundary conditions on the domain boundary. In all of these results we present both the resolution and the number points that are in play in the calculation (those not completely covered by the embedded boundary). This number of points represents the number of degrees of freedom in the calculation.

6.1 Accuracy vs. Resolution for a Smooth Geometry

First we compare our algorithm to the Schwartz, et al. algorithm with a smooth geometry. Here, our geometry is the exterior of the ellipse whose implicit function is described by (34). Figure 7 shows a solution error plot with Neumann boundary conditions. Figure 8 shows a solution error plot with Dirichlet boundary conditions. Both cases show that the solution error that is distributed throughout the domain. Tables 5 and 6 show norms of our solution error at many resolutions for Neumann and Dirichlet boundary conditions, respectively, for both algorithms. For both Neumann and Dirichlet boundary conditions, we get much smaller errors even with greatly reduced resolution. For example, in the Neumann case, we get an order of magnitude smaller errors at 64^2 (less than one thousand degrees of freedom) than Schwartz, et al. get at 1024^2 (almost one million degrees of freedom).

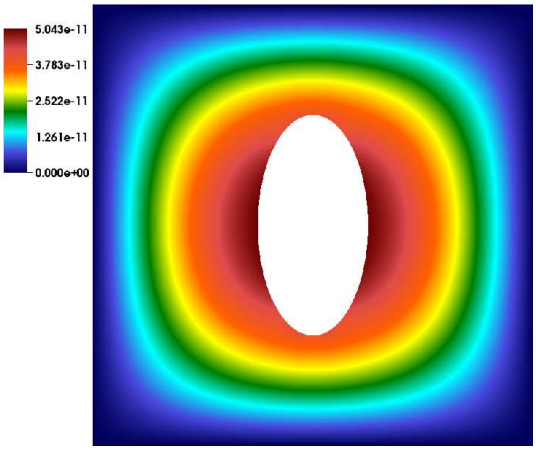

Figure 7: Solution error for the ellipse geometry in two dimensions using Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain boundary. This is using the current fourth order algorithm. Resolution is 512^2.

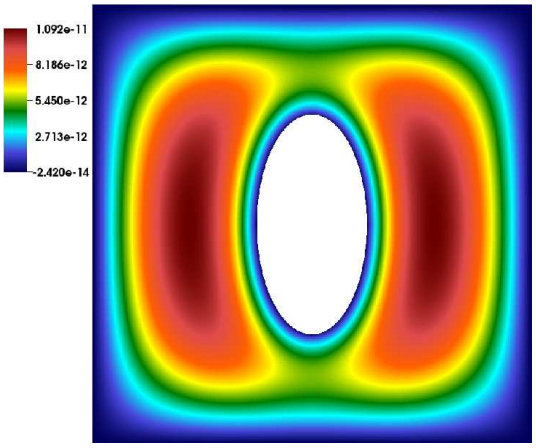

Figure 8: Solution error for the ellipse geometry in two dimensions using Dirichlet boundary conditions both on the embedded and on the domain boundary. This is using the current fourth order algorithm. Resolution is 512^2.

Algorithm	Resolution	Num. Points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$
Schwartz	32^2	952	1.419e-03	3.354e-04	4.144e-04
Schwartz	64^2	3752	3.511e-04	8.645e-05	1.070e-04
Schwartz	128^2	14884	8.639e-05	2.186e-05	2.709e-05
Schwartz	256^2	59312	2.083e-05	5.443e-06	6.741e-06
Schwartz	512^2	236832	5.167e-06	1.354e-06	1.677e-06
Schwartz	1024^2	946432	1.272e-06	3.380e-07	4.185e-07
Current	32^2	952	2.786e-06	1.041e-06	1.316e-06
Current	64^2	3752	1.833e-07	6.897e-08	8.670e-08
Current	128^2	14884	1.176e-08	4.459e-09	5.557e-09
Current	256^2	59312	7.431e-10	2.833e-10	3.512e-10
Current	512^2	236832	5.045e-11	1.941e-11	2.396e-11
Current	1024^2	946432	1.809e-11	7.489e-12	9.105e-12

Table 5: Error vs. refinement comparison with the second order Schwartz, et al. algorithm with the elliptical geometry. This uses Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain boundary.

Algorithm	Resolution	Num. Points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$
Schwartz	32^2	952	5.415e-03	1.322e-03	1.715e-03
Schwartz	64^2	3752	1.590e-03	3.592e-04	4.665e-04
Schwartz	128^2	14884	4.581e-04	9.569e-05	1.243e-04
Schwartz	256^2	59312	1.259e-04	2.498e-05	3.247e-05
Schwartz	512^2	236832	3.430e-05	6.449e-06	8.381e-06
Schwartz	1024^2	946432	9.289e-06	1.647e-06	2.142e-06
Current	32^2	952	7.190e-07	3.169e-07	3.841e-07
Current	64^2	3752	4.146e-08	1.907e-08	2.300e-08
Current	128^2	14884	2.527e-09	1.173e-09	1.400e-09
Current	256^2	59312	1.564e-10	7.370e-11	8.717e-11
Current	512^2	236832	1.093e-11	5.195e-12	6.109e-12
Current	1024^2	946432	5.159e-11	2.583e-12	3.024e-12

Table 6: Error vs. refinement comparison with the the second order Schwartz, et al. algorithm with the elliptical geometry. This uses Dirichlet boundary conditions everywhere.

6.2 Accuracy vs. Resolution for a Non-smooth Geometry

Now we compare our compare our algorithm to the Schwartz, et al. algorithm with a geometry that is only piecewise smooth. The geometry is given by the exterior of four or circles as shown in Figure 9. The implicit function is C_1 discontinuous. Figure 10 shows a solution error plot with Neumann boundary

conditions. Figure 11 shows a solution error plot with Dirichlet boundary conditions. In both cases, the solution error is concentrated near the discontinuities in the geometry; in the Dirichlet case, it is concentrated in a very small area.

For Neumann boundary conditions at the embedded boundary, Tables 7 and 8 compare our solution errors with the the Schwartz, et al. algorithm for Dirichlet boundary conditions. The errors for the two algorithms are comparable for Neumann boundary conditions though the higher order algorithm does show better results with Dirichlet boundary conditions.

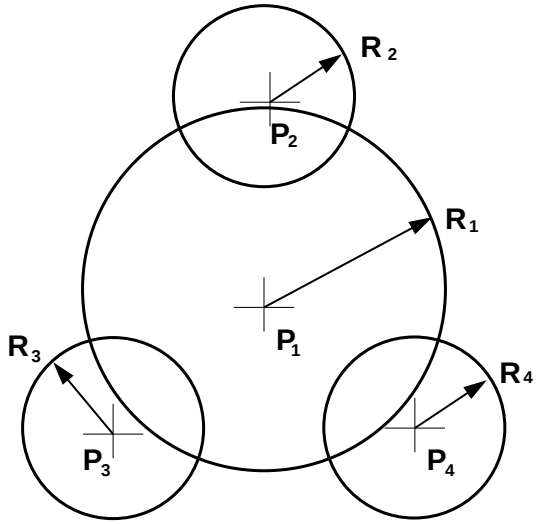

Figure 9: Diagram for our four circle geometry. The computational geometry is the region not covered by these four circles in the unit square. $R_1 = 0.2$. $R_2 = R_3 = R_4 = 0.1$. $P_1 = (0.5, 0.5, 0.5)$. $P_2 = (0.50.7350.5)$. $P_3 = 0.29650.38250.5$. $P_4 = 0.70350.38250.5$.

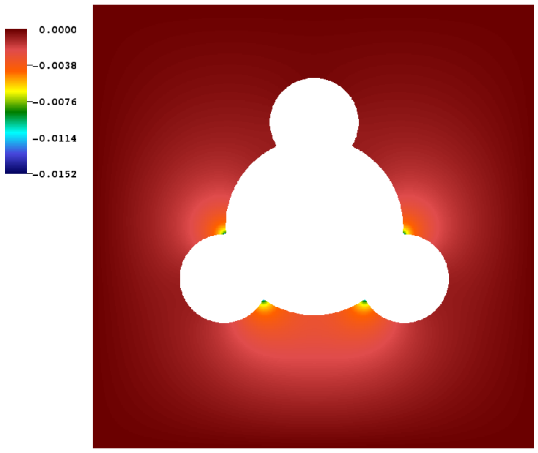

Figure 10: Solution error for the four circle geometry oin two dimensions using Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain boundary. This is using the current fourth order algorithm. Resolution is 512^2.

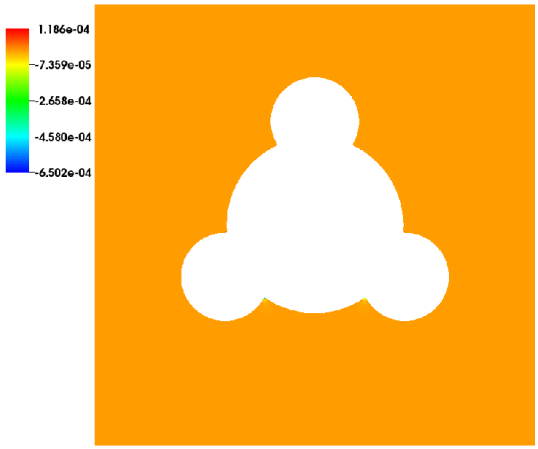

Figure 11: Solution error for the four circle geometry in two dimensions using Dirichlet boundary conditions both on the embedded and on the domain boundary. This is using the current fourth order algorithm. Resolution is 512^2. The error is concentrated very near the cusps in the geometry.

Algorithm	Resolution	Num. Points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$
Schwartz	32^2	862	3.525e-02	3.006e-03	4.625e-03
Schwartz	64^2	3370	2.231e-02	9.703e-04	1.585e-03
Schwartz	128^2	13318	1.291e-02	5.649e-04	1.134e-03
Schwartz	256^2	52930	1.776e-03	8.885e-05	1.325e-04
Schwartz	512^2	211136	3.576e-03	1.403e-04	2.192e-04
Schwartz	1024^2	843316	3.033e-03	1.417e-04	2.089e-04
Current	32^2	862	5.751e-02	4.869e-03	7.208e-03
Current	64^2	3370	3.096e-02	1.420e-03	2.721e-03
Current	128^2	13318	2.817e-02	2.132e-03	3.204e-03
Current	256^2	52930	1.969e-02	1.383e-03	2.020e-03
Current	512^2	211136	1.517e-02	6.754e-04	1.042e-03
Current	1024^2	843316	7.495e-03	3.419e-04	4.934e-04

Table 7: Error vs. refinement comparison with the second order Schwartz, et al. algorithm for the four circle geometry. This uses Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain boundary.

Algorithm	Resolution	Num. Points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$
Schwartz	32^2	862	2.191e-02	1.333e-03	1.750e-03
Schwartz	64^2	3370	1.026e-02	3.717e-04	4.908e-04
Schwartz	128^2	13318	2.850e-03	9.703e-05	1.300e-04
Schwartz	256^2	52930	5.719e-04	2.654e-05	3.491e-05
Schwartz	512^2	211136	8.178e-04	6.686e-06	9.117e-06
Schwartz	1024^2	843316	8.979e-04	1.620e-06	2.330e-06
Current	32^2	862	1.818e-02	1.604e-04	5.435e-04
Current	64^2	3370	2.797e-03	3.228e-05	1.089e-04
Current	128^2	13318	2.317e-02	4.557e-06	7.391e-05
Current	256^2	52930	2.705e-03	2.014e-07	4.966e-06
Current	512^2	211136	6.502e-04	5.940e-08	2.263e-06
Current	1024^2	843316	3.345e-04	8.430e-09	4.955e-07

Table 8: Error vs. refinement comparison with the second order Schwartz, et al. algorithm with the four circle geometry. This uses Dirichlet boundary conditions everywhere.

6.3 Singular Solutions and Error Characteristics

One might be tempted to ascribe this loss in accuracy to a poor approximation of geometric moments. After all, the implicit function from which the moments are generated is not smooth near the corner. To test this theory, we use the refinement algorithm described in [32] to refine the cells near circle intersections

by a factor of 1024^2 in each direction. Since we know the geometric moments of the uncut subcells exactly and only one subcell contains the discontinuity, this increases the accuracy of the geometric moments dramatically. When we run this test, the solution errors do not change. The reason that our accuracy degrades for the four circle geometry is that the solution to the error equation is singular at these points. With homogeneous boundary conditions, the solution of the Poisson equation is singular near corners whose angle is greater than $\pi/2$ [21].

Given a truncaiton error T (defined in (31)) and the solution error ϵ (defined in (33)), the error equation can be written

$$L(\epsilon) = T. \tag{36}$$

The boundary conditions for ϵ are homogeneous analogs of the boundary conditions for ϕ (if ϕ's boundary conditions are inhomogeneous Dirichlet, ϵ's boundary conditions are homogeneous Dirichlet). Because our equation is linear, we can separate the truncation error into two parts. We define T^s to be the truncation error in cells within the stencil width w of the singular points. We define the non-singular component of the truncaiton error to be $T^n = T - T^s$. We then compute the convergence rate of the solution error ϵ^n in the absence of the singular points of the truncation error by solving

$$L\epsilon^n = T^n \tag{37}$$

with the appropriate homogeneous boundary conditions.

We are given a set of M circles $\{C_1...C_M\}$, which intersect at the set of volumes $\mathcal{P}^s = \{P_1...P_M\}$. The singular part of the truncation error T_s is given by

$$T^s_{\mathbf{v}} = \begin{pmatrix} T & \text{if } \mathbf{v} \in \mathcal{P}^s \\ 0 & \text{otherwise} \end{pmatrix}. \tag{38}$$

We solve (37) using both the current fourth-order algorithm and the second-order Schwartz, et al. algorithm. The comparisons with the for the Schwartz, et al. algorithm are given in tables 9 and 10. Again, we show that the current algorithm has comparable errors at 32^2 resolution to the Schwartz, et al. algorithm at 1024^2 resolution. So if one is able to remove the singular part of the solution, she can achieve high accuracy with this algorithm even if the implicit function which generates the geometry is not smooth.

Algorithm	Resolution	Num. Points	$L_\infty(\epsilon^n)$	$L_1(\epsilon^n)$	$L_2(\epsilon^n)$
Schwartz	32^2	862	6.121e-03	1.440e-03	1.875e-03
Schwartz	64^2	3370	1.552e-03	3.911e-04	5.066e-04
Schwartz	128^2	13320	4.119e-04	1.002e-04	1.321e-04
Schwartz	256^2	52930	1.355e-04	2.669e-05	3.515e-05
Schwartz	512^2	211136	3.215e-05	6.797e-06	8.913e-06
Schwartz	1024^2	843316	9.172e-06	1.640e-06	2.152e-06
Current	32^2	862	5.126e-07	1.741e-07	2.236e-07
Current	64^2	3370	2.684e-08	1.080e-08	1.338e-08
Current	128^2	13318	1.586e-09	6.380e-10	7.764e-10
Current	256^2	52930	9.650e-11	3.882e-11	4.683e-11
Current	512^2	211136	6.676e-12	2.693e-12	3.232e-12
Current	1024^2	843316	3.264e-12	1.320e-12	1.577e-12

Table 9: Convergence of the non-singular part of the solution error vs. refinement for both the current algorithm and the Schwartz, et al. algorithm. This uses Dirichlet boundary conditions everywhere.

Algorithm	Resolution	Num. Points	$L_\infty(\epsilon^n)$	$L_1(\epsilon^n)$	$L_2(\epsilon^n)$
Schwartz	32^2	862	1.329e-03	3.557e-04	4.370e-04
Schwartz	64^2	3370	2.881e-04	8.740e-05	1.073e-04
Schwartz	128^2	13320	7.068e-05	2.084e-05	2.552e-05
Schwartz	256^2	52930	1.777e-05	5.171e-06	6.327e-06
Schwartz	512^2	211136	4.382e-06	1.278e-06	1.564e-06
Schwartz	1024^2	843316	1.033e-06	3.222e-07	3.946e-07
Current	32^2	862	2.353e-06	7.242e-07	8.939e-07
Current	64^2	3370	1.864e-07	5.838e-08	7.354e-08
Current	128^2	13318	1.133e-08	3.783e-09	4.735e-09
Current	256^2	52930	7.215e-10	2.405e-10	2.993e-10
Current	512^2	211136	5.392e-11	1.649e-11	2.046e-11
Current	1024^2	843316	1.235e-11	5.044e-12	6.154e-12

Table 10: Convergence of the non-singular part of the solution error vs. refinement for the current algorithm and for the Schwartz, et al. algorithm. This uses Dirichlet boundary conditions on the domain boundary and Neumann boundary conditions on the embedded boundary.

7 Geometric Regularization and Accuracy

We recognize that the technique of removing the singular parts of the truncation error is not generally useful to larger applications. The tests presented in Section 6.3 are predicated upon knowing a priori the singulr points. For high

order methods to be more generally useful, they must produce much better accuracy than lower order methods in the presence of geometric discontinuities without this prior knowledge. In this section, we present a method to smooth the geometric description over a controlled length scale. We then show that, if one is careful about how this length scale converges with grid refinement, she can retain superior accuracy compared to lower order methods even when the input implicit function is only C^0.

7.1 Smoothing the Geometric Description

Recall that, to generate our geometric moments using the algorithm described in [32], we must start with an implicit function $F(\mathbf{x})$ whose zero surface (or contour in 2D) forms the the embedded boundary. Consider the geometry described in Figure 9. The implicit function for each circle C_i, with radius r_i and center $\mathbf{y}_i$ is given by

$$C_i(\mathbf{x}) = r_i^2 - \sum_{d=1}^{\mathcal{D}} (x_d^2 - y_{i,d}^2).$$

The overall implicit function at any point is given by taking the maximum of the four functions.

$$F(\mathbf{x}) = \max_{1<=d<=4} C_i(\mathbf{x}) \tag{39}$$

Since our geometry is smooth away from specific intersection locations, we wish to only smooth within a length scale δ from the intersections of implicit function zero surfaces. To smooth this description we could use a mollifying function and integrate the convolution directly as in [36]. This has the advantage that the length scale over which the smoothing happens is well defined. These functions can be delicate, however, to integrate numerically. Shapiro [33] presents an alternative approach called R-functions (named for V. L. Rvachev, the originator of the concept [30]), in which logical functions such as maxima, minima and absolute values are replaced by differentiable functions with the same zero surfaces. Though this method is far more numerically tractable, the functions for maxima that Shapiro presents do not have a well-defined length scale over which they smooth. The smoothing method described here (which can be properly described as an R-function) provides both a well-defined smoothing length and is numerically tractable.

One way to write the maxima function used in (39) is using an absolute value:

$$\max(a, b) = \frac{1}{2}(a + b + |a - b|).$$

Let us define a function $\max_\delta$ which smooths the function max over a length scale δ

$$\max(a, b) = \frac{1}{2}(a + b + A_\delta(a - b)).$$

where A_δ is the convolution of the absolute function with a sufficiently smooth function $\psi_\delta(x)$ with compact support in contained within $x \in [-\delta, \delta]$:

$$A_\delta(x) = \int_{-\infty}^{\infty} \psi_\delta(x-y)|y|dy = \int_{0}^{\infty} \psi_\delta(x-y)ydy - \int_{-\infty}^{0} \psi_\delta(x-y)ydy.$$

Since our algorithm is fourth order in fluxes, we use geometric our geometric moments to fourth order. The algorithm in [32] requires that the implicit function must have derivatives to fourth order. This implies that the mollifier ψ_δ needs to be C^4 and these derivatives must also have compact support. We also require $\int_{-\infty}^{\infty} \psi(y)dy = 1$. Our choice of ψ_δ

$$\psi_\delta(x) = \left(\begin{array}{ll} \frac{4}{3\delta}\cos^4(\frac{\pi x}{2\delta}) & \text{if } -\delta \leq x \leq \delta \\ 0 & \text{otherwise} \end{array} . \right)$$

fulfills these requirements. We need to integrate only where the mollifier is nonzero. If a and b are signed distance functions, then δ is the length scale over which $A_\delta(a, b)$ represents a smoothing the of the absolute value function.

7.2 Regularization Length Scale and Grid Refinement

Now we investigate the obvious question in all of this, how does one pick the length scale δ? For the piecewise-smooth geometric description presented in Section 6.2, we present three different schemes for δ and see how the accuracy changes with grid refinement. First we use a constant $\delta = 0.01$. Second, we make delta vary linearly with h ($\delta = 4h$). Finally we make $\delta = \sqrt{(R_1 h)}$, where R_1 is described in Figure 9. The convergence rates for the three schemes are quite different.

First, we set our geometric regularization length to a constant $\delta = 0.01$. Tables 11 and 12 show error rates for Dirichlet and Neumann boundary conditions at the cut faces, respectively. With this fixed δ, the current algorithm shows much smaller errors than Schwartz, et al. In the L_1 norm, we get better error rates at 32^2 than Schwartz, et al. gets at 1024^2.

Next, we set our geometric regularization length to $\delta = 4h$. Tables 13 and 14 show the error rates for Dirichlet and Neumann boundary conditions at the cut faces, respectively. With δ converging linearly with grid refinement, the improvement over Schwartz, et al. is far more modest, especially with Neumann boundary conditions at the cut faces.

Finally, we set our geometric regularization length to $\delta = \sqrt{(R_1 h)}$. Tables 15 and 16 show the error rates for Dirichlet and Neumann boundary conditions at the cut faces, respectively. With this formulation of δ, we once again get much better error rates than Schwartz, et al.. Here again, in the L_1 norm, we get better error rates at 32^2 than Schwartz, et al. gets at 1024^2.

Clearly, how the regularization length varies with grid refinement is an important concern. We suspect that the optimal formulation will depend upon the nature of the partial differential equation.

Algorithm	Resolution	Num. Points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$
Schwartz	32^2	862	1.184e-02	1.869e-03	2.522e-03
Schwartz	64^2	3368	1.715e-03	4.142e-04	5.414e-04
Schwartz	128^2	13316	5.922e-04	1.023e-04	1.356e-04
Schwartz	256^2	52916	1.355e-04	2.676e-05	3.527e-05
Schwartz	512^2	211062	3.215e-05	6.784e-06	8.892e-06
Schwartz	1024^2	843004	8.063e-06	1.644e-06	2.159e-06
Current	32^2	862	7.904e-03	4.768e-05	2.992e-04
Current	64^2	3368	9.380e-05	1.418e-06	4.421e-06
Current	128^2	13316	2.921e-06	9.434e-09	5.098e-08
Current	256^2	52916	2.745e-07	2.839e-10	2.146e-09
Current	512^2	211062	3.223e-09	3.365e-12	3.125e-11
Current	1024^2	843004	1.063e-10	2.001e-12	2.434e-12

Table 11: Comparison of error rates with the four-circle geometry for the current algorithm and for the Schwartz, et al. algorithm. The boundary conditions are Dirichlet everywhere. Here we set the geometric regularization length to a constant $\delta = 0.01$

Algorithm	Resolution	Num. Points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$
Schwartz	32^2	862	1.989e-02	2.026e-03	3.003e-03
Schwartz	64^2	3368	2.593e-03	2.686e-04	3.564e-04
Schwartz	128^2	13316	1.171e-03	1.207e-04	1.657e-04
Schwartz	256^2	52916	2.522e-04	3.012e-05	4.093e-05
Schwartz	512^2	211062	6.144e-05	7.472e-06	1.014e-05
Schwartz	1024^2	843004	1.417e-05	1.798e-06	2.436e-06
Current	32^2	862	1.525e-01	1.166e-02	1.905e-02
Current	64^2	3368	1.739e-03	5.812e-05	1.102e-04
Current	128^2	13316	7.054e-05	3.077e-06	5.886e-06
Current	256^2	52916	3.593e-06	4.053e-08	8.884e-08
Current	512^2	211062	1.425e-07	9.227e-09	1.473e-08
Current	1024^2	843004	6.998e-09	4.220e-10	6.809e-10

Table 12: Comparison of error rates with the four-circle geometry for the current algorithm and for the Schwartz, et al. algorithm. The domain boundary conditions are Dirichlet and the embedded boundary boundary conditions are Neumann. Here we set the geometric regularization length to a constant $\delta = 0.01$

Algorithm	Resolution	Num. Points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$
Schwartz	32^2	828	6.864e-03	1.527e-03	1.993e-03
Schwartz	64^2	3238	1.900e-03	3.941e-04	5.172e-04
Schwartz	128^2	12810	4.826e-04	9.172e-05	1.213e-04
Schwartz	256^2	50918	1.237e-04	2.394e-05	3.157e-05
Schwartz	512^2	203052	3.354e-05	6.085e-06	8.008e-06
Schwartz	1024^2	810964	8.330e-06	1.513e-06	1.989e-06
Current	32^2	828	9.448e-05	1.965e-06	5.407e-06
Current	64^2	3326	9.659e-07	1.622e-08	4.080e-08
Current	128^2	13266	2.179e-08	8.458e-10	1.295e-09
Current	256^2	52886	2.110e-08	9.797e-11	2.689e-10
Current	512^2	211088	1.028e-08	2.417e-11	1.331e-10
Current	1024^2	843270	8.976e-08	1.068e-11	2.235e-10

Table 13: Comparison of error rates with the four-circle geometry for the current algorithm and for the Schwartz, et al. algorithm. The boundary conditions are Dirichlet everywhere. Here we set the geometric regularization length to $\delta = 4h$

Algorithm	Resolution	Num. Points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$
Schwartz	32^2	828	2.849e-03	4.876e-04	6.323e-04
Schwartz	64^2	3238	1.128e-03	1.457e-04	1.937e-04
Schwartz	128^2	12810	2.555e-04	3.155e-05	4.091e-05
Schwartz	256^2	50918	1.782e-04	1.154e-05	1.631e-05
Schwartz	512^2	203052	4.259e-06	1.249e-06	1.554e-06
Schwartz	1024^2	810964	5.368e-06	2.148e-07	3.271e-07
Current	32^2	828	1.683e-03	1.122e-04	2.043e-04
Current	64^2	3326	6.971e-06	5.483e-07	9.162e-07
Current	128^2	13266	6.512e-07	6.887e-08	9.896e-08
Current	256^2	52886	8.852e-07	7.875e-08	1.125e-07
Current	512^2	211088	4.898e-07	3.394e-08	5.000e-08
Current	1024^2	843270	3.257e-07	1.630e-08	2.439e-08

Table 14: Comparison of error rates with the four-circle geometry for the current algorithm and for the Schwartz, et al. algorithm. The boundary conditions are Dirichlet on the domain boundary and Neumann on the embedded boundary. Here we set the geometric regularization length to $\delta = 4h$

Algorithm	Resolution	Num. Points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$
Schwartz	32^2	846	6.581e-03	1.499e-03	1.997e-03
Schwartz	64^2	3312	1.717e-03	3.672e-04	4.784e-04
Schwartz	128^2	13088	4.546e-04	9.330e-05	1.220e-04
Schwartz	256^2	52022	1.250e-04	2.548e-05	3.365e-05
Schwartz	512^2	207510	3.837e-05	6.258e-06	8.146e-06
Schwartz	1024^2	828794	1.011e-05	1.600e-06	2.111e-06
Current	32^2	846	5.402e-06	2.143e-07	3.879e-07
Current	64^2	3330	2.792e-07	1.039e-08	1.807e-08
Current	128^2	13252	1.421e-08	5.089e-10	7.016e-10
Current	256^2	52794	2.260e-09	3.414e-11	7.429e-11
Current	512^2	210856	4.074e-10	2.406e-12	5.238e-12
Current	1024^2	842726	3.629e-11	1.996e-12	2.390e-12

Table 15: Comparison of error rates with the four-circle geometry for the current algorithm and for the Schwartz, et al. algorithm. The boundary conditions are Dirichlet everywhere. Here we set the geometric regularization length to $\delta = \sqrt{(}R_1 h)$

Algorithm	Resolution	Num. Points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$
Schwartz	32^2	846	1.579e-03	5.218e-04	6.538e-04
Schwartz	64^2	3312	3.690e-04	1.203e-04	1.507e-04
Schwartz	128^2	13088	9.429e-05	3.133e-05	3.924e-05
Schwartz	256^2	52022	3.297e-05	8.306e-06	1.045e-05
Schwartz	512^2	207510	5.283e-06	1.822e-06	2.270e-06
Schwartz	1024^2	828794	1.309e-06	4.342e-07	5.399e-07
Current	32^2	846	6.139e-05	4.725e-06	8.113e-06
Current	64^2	3330	1.274e-06	8.435e-08	1.691e-07
Current	128^2	13252	4.698e-07	4.297e-08	6.226e-08
Current	256^2	52794	8.422e-08	7.356e-09	1.057e-08
Current	512^2	210856	2.127e-08	1.846e-09	2.645e-09
Current	1024^2	842726	3.693e-09	2.843e-10	4.046e-10

Table 16: Comparison of error rates with the four-circle geometry for the current algorithm and for the Schwartz, et al. algorithm. The boundary conditions are Dirichlet on the domain boundary and Neumann on the embedded boundary. Here we set the geometric regularization length to $\delta = \sqrt{(}R_1 h)$

8 Conclusions

We present a fourth order, conservative discretization of Poisson's equation in the presence of complex geometry. We show that our algorithm converges at the expected rate for smooth solutions and geometries. We show that our algorithm

has a similar eigenvalue spectrum to the a widely-used second order but is much more accurate with a sufficiently smooth geometric description. We show that the effect of geometric discontinuties on error rates can be profound. Even in the presence of these discontinuities, however, higher order convergence can be recovered if one removes the singular parts of the solution or smooths the geometric description. To retain higher order accuracy, how the smoothing length scale varies with grid refinement is an important concern.

References

[1] M. J. Aftosmis, M. J. Berger, and J. E. Melton. Robust and efficient Cartesian mesh generation for component-base geometry. *AIAA Journal*, 36(6):952–960, June 1998.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[3] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.5, Argonne National Laboratory, 2014.

[4] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, and Hong Zhang. PETSc Web page, 2014.

[5] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry 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.

[6] Susanne Brenner and Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Springer, New York, 2007.

[7] C. Campos, J. E. Roman, E. Romero, and A. Tomas. SLEPc users manual. Technical Report DSIC-II/24/02 - Revision 3.3, D. Sistemes Informàtics i Computació, Universitat Politècnica de València, 2012.

[8] H. Cheng, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. *JCP*, 155:468–496, 1999.

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

[10] P. Colella, D. T. Graves, T. J. Ligocki, D. F. Martin, D. Modiano, D. B. Serafini, and B. Van Straalen. Chombo Software Package for AMR Applications - Design Document. Technical Report LBNL-6616E, LBNL, July 2014.

[11] P. Colella, D. T. Graves, T. J. Ligocki, G.H. Miller, D. Modiano, P.O. Schwartz, B. Van Straalen, J. Pillod, D. Trebotich, and M. Barad.

Ebchombo software package for cartesian grid, embedded boundary application. Technical Report LBNL-6615E, LBNL, 2014.

[12] D. Devendran, D. T. Graves, and H. Johansen. A hybrid multigrid algorithm for Poisson's equation using an adaptive, fourth order treatment of cut cells. Technical Report LBNL-1004329, LBNL, 2014.

[13] Z. Dragojlovic, F. Najmabadi, and M. Day. "An embedded boundary method for viscous, conducting compressible flow". *J. Comp. Phys.*, 216(1):37–51, 2006.

[14] F. Gibou and R. Fedkiw. A fourth order accurate discretization for the laplace and heat equations on arbitrary domains, with applications to the stefan problem. *J. Comput. Phys.*, 202:577–601, 2005.

[15] 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. *Communications in Applied Mathematics and Compuational Science*, 8(1):99–122, 2013.

[16] L. Greengard and J-Y Lee. A direct adaptive poisson solver of arbitrary order accuracy. *J. Comput. Phys.*, 125:415–424, 1996.

[17] V. Hernandez, J. E. Roman, and V. Vidal. SLEPc: Scalable Library for Eigenvalue Problem Computations. *Lecture Notes in Computer Science*, 2565:377–391, 2003.

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

[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 Finite Difference Methods for Elliptic and Parabolic Partial Differential Equations on Irregular Domains*. PhD thesis, University of California, Berkeley, 1997.

[21] L.D. Landau and E. M. Lifshitz. *Fluid Mechanics*. Pergammon Press, Oxford, second edition, 1987.

[22] Randall J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhauser-Verlag, Basel, Boston, Berlin, 1990.

[23] R.J. LeVeque and Z. Ling. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM Journal of Numerical Analysis*, 31(4):1019–1044, 1994.

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

[25] A. McKenney, L. Greengard, and A. Mayo. A fast poisson solver for complex geometries. *J. Comput. Phys.*, 118:348–355, 1995.

[26] G. H. Miller and D. Trebotich. An embedded boundary method for the navier-stokes equations on a time-dependent domain. *Communications in Applied Mathematics and Computational Science*, 7:1–31, 2012.

[27] A. Nonaka, D. Trebotich, G. H. Miller, D. T. Graves, and P. Colella. A higher-order upwind method for viscoelastic flow. *Comm. App. Math. and Comp. Sci.*, 4:57–83, 2009.

[28] 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.

[29] S. Pirzadeh. Advanced unstructured grid generation for complex aerodynamic applications. *AIAA Journal*, 48(5):904–915, 2010.

[30] V. L. Rvachev. On analytical description of some geometric objects. *Reports (Doklady) of Academy of Sciences, USSR*, 1963.

[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. *Journal of Computational Physics*, 211(2):531–550, January 2006.

[32] P. Schwartz, J. Percelay, T. Ligocki, H. Johansen, D. T. Graves, D. Devendran, P. Colella, and E. Ateljevich. High accuracy embedded boundary grid generation using the divergence theorem. *Communications in Applied Mathematics and Computational Science 10-1.*

[33] Vadim Shapiro. Semi-analytic geometry with r-functions. *Acta Numerica*, pages 1–65, 2007.

[34] S. L. Sobolev. *Partial Differential Equations of Matematical Physics.* Dover Publications, New York, NY, 1964.

[35] Gilbert Strang. *Linear Algebra and its Applications.* Academic Press, New York, NY, 1976.

[36] Eitan Tadmor and Jared Tanner. Adaptive mollifiers for high resolution recovery of piecewise smooth data from its spectral information. *Found. Comput. Math.*, 2:155–189, 2002.

[37] D. Trebotich, G. H. Miller, and M. D. ByBee. A penalty method to model particle interactions in dna-laden flows. *Journal of Nanoscience and Nanotechnology*, 8:3749–3756, 2008.