COMM. APP. MATH. AND COMP. SCI.
Vol. , No. , 2017
dx.doi.org/10.2140/camcos.2017..101

A FOURTH-ORDER CARTESIAN GRID EMBEDDED BOUNDARY METHOD FOR POISSON'S EQUATION

DHARSHI DEVENDRAN, DANIEL T. GRAVES,
HANS JOHANSEN AND TERRY LIGOCKI

In this paper, we present a 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 nonsmooth geometries.

1. Introduction

There are many numerical approaches to solve Poisson's equation in complex geometries. Green function approaches [26; 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 multimaterial 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 three-dimensional domains is still an expensive, globally coupled computation, and an open area of research [30].

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, the discrete domain is the intersection of the complex geometry with a regular Cartesian grid. Such

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.
MSC2010: 65M08, 65M50.
Keywords: Poisson equation, finite volume methods, high order, embedded boundary.

intersections are local, and can be calculated very efficiently in parallel, enabling fast computation of solution-dependent moving boundaries [1; 33; 27]. Cut cells have been used successfully to solve Poisson's equation in finite volume [19; 32] and finite difference [14; 24] 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; 32]. 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 [32] has been used in many larger applications, including incompressible Navier–Stokes with moving boundaries [27], 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 number of degrees of freedom that are required to achieve a given degree of accuracy.

Realistic boundaries can have discontinuities in their derivatives. For example, it is common to make a geometric description out of the intersection of several simpler geometries. We show that these discontinuities can have profound effects upon accuracy. We provide a strategy for maintaining higher-order accuracy in the presence of geometric discontinuities using geometric regularization with a smoothing length which can be controlled. We show that the rate at which this smoothing length converges with grid refinement matters greatly.

2. Algorithm

The algorithm is described in several steps. First, we introduce the embedded boundary finite volume discretization for Poisson's equation. Then we obtain a Taylor-series-based interpolant of the solution and operator that is compatible with cell- and face-averaged quantities, and achieves the desired order of accuracy. Lastly, we introduce a weighted least-squares approach that uses nearest neighbor values to stably interpolate the quantities needed by the finite volume operator.

2.1. ***Finite volume discretization.*** Given a charge density ρ, Poisson's equation for the potential ϕ can be written as

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

Integrating this over a control volume V and applying the divergence theorem yields

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

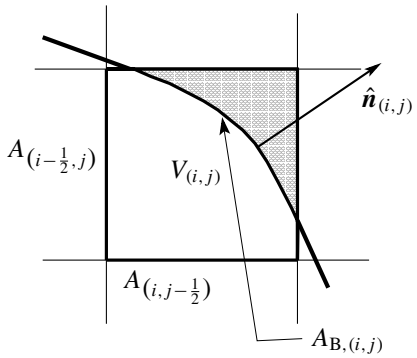


Figure 1. Illustration of cut cell notation. The shaded region is outside the solution domain. The volume $V_{\boldsymbol{i}} = V_{(i,j)}$ is connected to other volumes via the faces aligned with the coordinate planes. The EB face is formed by the intersection of the embedded boundary and the cell.

where $\hat{\boldsymbol{n}}$ is the outward-facing unit normal to the surface.

Our volumes are rectangular control volumes on a Cartesian mesh, cut by an embedded boundary. Formally, the underlying description of space is given by

$$\Upsilon_{\boldsymbol{i}} = \left[(\boldsymbol{i} - \tfrac{1}{2}\boldsymbol{u})h, (\boldsymbol{i} + \tfrac{1}{2}\boldsymbol{u})h\right], \quad \boldsymbol{i} \in \mathbb{Z}^{\mathcal{D}},$$

where $\mathcal{D}$ is the dimensionality of the problem, h is the mesh spacing, and $\boldsymbol{u}$ is the vector whose entries are all one. Note we use bold font $\boldsymbol{u} = (u_1, \ldots, u_d, \ldots, u_{\mathcal{D}})$ to indicate a vector quantity. Given an irregular domain Ω, we obtain control volumes $V_{\boldsymbol{i}} = \Upsilon_{\boldsymbol{i}} \cap \Omega$ and faces $A_{\boldsymbol{i} \pm \frac{1}{2}\boldsymbol{e}_d}$, which are the intersection of the boundary of $\partial V_{\boldsymbol{i}}$ with the coordinate planes $\left\{\boldsymbol{x} : x_d = \left(i_d \pm \frac{1}{2}\right)h\right\}$ ($\boldsymbol{e}_d$ is the unit vector in the d direction). We also define $A_{\mathrm{B},\boldsymbol{i}}$ to be the intersection of the boundary of the irregular domain with the Cartesian control volume: $A_{\mathrm{B},\boldsymbol{i}} = \partial\Omega \bigcap \Upsilon_{\boldsymbol{i}}$. Figure 1 illustrates a volume cut by an embedded boundary.

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

$$\int_{V_i} \nabla \cdot (\nabla\phi)\, dV = \sum_{d=1}^{\mathcal{D}} \left(\int_{A_{i+\frac{1}{2}e_d}} F_d\, dA - \int_{A_{i-\frac{1}{2}e_d}} F_d\, dA + \int_{A_{\mathrm{B},i}} F_d \hat{\boldsymbol{n}}_d\, dA \right). \quad (3)$$

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

$$\langle\phi\rangle_{\boldsymbol{i}} = \frac{1}{|V_{\boldsymbol{i}}|} \int_{V_i} \phi\, dV. \quad (4)$$

The average flux over a coordinate face is defined as

$$\langle F_d \rangle_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d} = \frac{1}{|A_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d}|} \int\limits_{A_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d}} F_d \, dA,$$

and the average flux at the irregular face is given by

$$\langle F_d \, \hat{\boldsymbol{n}}_d \rangle_{\mathrm{B},\boldsymbol{i}} = \frac{1}{|A_{\mathrm{B},\boldsymbol{i}}|} \int\limits_{A_{\mathrm{B},\boldsymbol{i}}} F_d \hat{\boldsymbol{n}}_d \, dA.$$

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

$$\kappa = h^{-\mathcal{D}} |V_{\boldsymbol{i}}|. \tag{5}$$

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

$$\begin{aligned} \kappa L(\phi)_{\boldsymbol{i}} &= \kappa \langle \nabla \cdot \boldsymbol{F} \rangle_{\boldsymbol{i}} \\ &= \frac{1}{h^{\mathcal{D}}} \int_{V_{\boldsymbol{i}}} \nabla \cdot \boldsymbol{F} \, dV \\ &= \frac{1}{h^{\mathcal{D}}} \sum_{d=1}^{\mathcal{D}} \left(|A_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}| \langle F_d \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} - |A_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}_d}| \langle F_d \rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}_d} + |A_{\mathrm{B},\boldsymbol{i}}| \langle F_d \, \hat{\boldsymbol{n}}_d \rangle_{\mathrm{B},\boldsymbol{i}} \right). \end{aligned} \tag{6}$$

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

$$\kappa \langle \nabla \cdot \nabla \phi \rangle_{\boldsymbol{i}} = \kappa \langle \rho \rangle_{\boldsymbol{i}} \tag{7}$$

for ϕ [19; 32], 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 [29; 27; 32; 13; 15; 28; 11]. In those algorithms, the face-averages of $\nabla\phi$ are approximated to second order by pointwise values at the centroids of faces. These fluxes are constructed by pointwise differencing those cell-centered values of ϕ.

2.2. ***Taylor series expansions for average quantities.*** In our discretization, we use the cell-averages of ϕ directly in the local polynomial expansion of ϕ that matches the boundary conditions to some order of accuracy. We use this polynomial expansion to construct a more accurate approximation to the face-averaged flux.

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

$$(\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{p}} = \prod_{d=1}^{\mathcal{D}} (x_d - \bar{x}_d)^{p_d}, \qquad \boldsymbol{p}! = \prod_{d=1}^{\mathcal{D}} p_d!\,.$$

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

$$m_{\boldsymbol{i}}^{\boldsymbol{p}}(\bar{\boldsymbol{x}}) = \int_{V_{\boldsymbol{i}}} (\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{p}}\, dV. \tag{8}$$

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

$$m_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d}^{\boldsymbol{p}}(\bar{\boldsymbol{x}}) = \int_{A_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d}} (\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{p}}\, dA. \tag{9}$$

We define two moments corresponding to the embedded boundary face $A_{\mathrm{B},\boldsymbol{i}}$,

$$m_{\mathrm{B},\boldsymbol{i}}^{\boldsymbol{p}}(\bar{\boldsymbol{x}}) = \int_{A_{\mathrm{B},\boldsymbol{i}}} (\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{p}}\, dA \tag{10}$$

and

$$m_{\mathrm{B},\boldsymbol{i},d}^{\boldsymbol{p}}(\bar{\boldsymbol{x}}) = \int_{A_{\mathrm{B},\boldsymbol{i}}} (\boldsymbol{x}-\bar{\boldsymbol{x}})^{\boldsymbol{p}}\, \hat{\boldsymbol{n}}_d(\boldsymbol{x})\, dA, \tag{11}$$

where $\hat{\boldsymbol{n}}_d$ is the d-th component of the outward-facing unit normal to the EB face.

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

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

where we use the multi-index partial derivative notation

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

If we put the expansion (12) into one of the integrals in (3) over a coordinate-aligned face and set $c_q = \frac{1}{q!}\phi^{(q)}(\bar{x})$, we get

$$\int_{A_{i\pm\frac{1}{2}e_d}} \frac{\partial \phi}{\partial x_d}\, dA = \sum_{|q|<=Q} q_d c_q \int_{A_{i\pm\frac{1}{2}e_d}} (x-\bar{x})^{q-e_d}\, dA + O(h^Q) \tag{14}$$

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

The flux equation at the irregular boundary becomes

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

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

All the moments can be generated to any order as shown in Schwartz, et al. [33]. Therefore, generating an $O(h^Q)$ algorithm for Poisson's equation reduces to finding the coefficients c_q.

2.3. ***Weighted least-squares interpolants.*** We define $\mathcal{N}_{i+\frac{1}{2}e_d}$ to be the set of volumes in the neighborhood of face $i+\frac{1}{2}e_d$. Our neighborhood algorithm is described in Section 2.3.3. We put the expansion (12) into (4) for every volume $V_j \in \mathcal{N}_{i+\frac{1}{2}e_d}$:

$$\langle\phi\rangle_j = \frac{1}{|V_j|}\sum_{|q|<=Q} c_q \int_{V_j} (x-\bar{x}_{i+\frac{1}{2}e_d})^q\, dV \tag{18}$$

$$= \frac{1}{|V_j|}\sum_{|q|<=Q} c_q m^q_j(\bar{x}_{i+\frac{1}{2}e_d}), \tag{19}$$

where $\bar{x}_{i+\frac{1}{2}e_d} = h\left(i+\frac{1}{2}e_d\right)$ is the center of the target face. This forms a system of equations for the coefficients c_q.

Define C to be a column vector composed of the Taylor coefficients c_q. In C, the powers of q are listed in lexicographical order. For example, in two dimensions, 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 $\langle\phi\rangle_j$ such that $V_j \in \mathcal{N}_{i+\frac{1}{2}e_d}$. Define M to be the matrix of the volume moments of the neighbors normalized by their

volumes. Each row of M corresponds to a particular neighbor. In particular, suppose $\mathcal{N}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = \{V_{j_1}, \ldots, V_{j_N}\}$. Then the l-th row of M are the moments for neighbor V_{j_l}. Each column of M corresponds to a particular power $\boldsymbol{q}$. Because the volume of $V_{\boldsymbol{j}}$ is $m^z_{\boldsymbol{j}}$, the first column consists of ones.

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

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

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 centered around the target face at $\bar{\boldsymbol{x}}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = h\left(\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}_d\right)$. The system of equations formed by (19) over the neighborhood $\mathcal{N}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ takes the form

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

Say there are P coefficients we need and N neighbors in $\mathcal{N}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$. If $N > P$, we have an over-determined 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 Section 2.3.2. Taking the Moore–Penrose pseudoinverse, we solve for the Taylor coefficients

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

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 $G = [\cdots q_d m^{\boldsymbol{q}-\boldsymbol{e}^d}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} \cdots]$, where $|\boldsymbol{q}| <= Q$. Then, (15) becomes

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

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 $S_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ acting on the solution, $|A_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}|\langle F_d\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = S^T_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}\Phi$, where

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

At every face in the domain, we solve for the stencil $S_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$. For faces near the domain boundaries and the embedded boundary we add boundary equations to the

system (22). This is discussed in Section 2.3.1. We solve for our stencils using the singular value decomposition framework from LAPACK [5].

Putting the flux stencils from (23) into (6), we get the stencil for our operator κL:

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

where $S_{\boldsymbol{i}}^{\mathrm{EB}}$, the stencil for the embedded boundary flux, is discussed in Section 2.3.1.

2.3.1. *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, including the boundary conditions in the interpolant (22) provides additional rows in the matrix that can compensate for ill-conditioned rows from small cells.

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.

We add equations that contain boundary condition information to the system (19) used to compute polynomial coefficients. This is done for two reasons: first, it increases the rank of the interpolation matrix if there are small cells. Second, in combination with the weighting matrix described in Section 2.3.2, it "smoothly" incorporates boundary conditions into interior cells that are near the embedded boundary, so the interpolants at nearby faces are more consistent with each other. We have found that this greatly improves the spectrum of the resulting operator.

Specifically, suppose a volume V_j in the neighbor set $\mathcal{N}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ contains a domain face $\boldsymbol{j}+\frac{1}{2}\boldsymbol{e}_d$ which has a Dirichlet boundary condition $\langle\phi\rangle_{\boldsymbol{j}+\frac{1}{2}\boldsymbol{e}_d} = \phi_{\mathrm{DB}}$. We add the equation

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

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

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

to the system (19). If V_j contains an embedded boundary face with a Dirichlet boundary condition $\langle\phi\rangle_i^{\mathrm{EB}} = \phi_{\mathrm{EB}}$ we add the appropriate form of (17):

$$\phi_{\mathrm{EB}} = \frac{1}{|A_{\mathrm{B}_i}|} \sum_{|\boldsymbol{q}|<=Q} c_{\boldsymbol{q}} \int_{A_{\mathrm{B}_i}} (\boldsymbol{x} - \bar{\boldsymbol{x}})^{\boldsymbol{q}} n_d \, dA \tag{28}$$

$$= \frac{1}{|A_{\mathrm{B}_i}|} \sum_{|\boldsymbol{q}|<=Q} c_{\boldsymbol{q}} m^{\boldsymbol{q}}_{\mathrm{B}_i,d}(\bar{\boldsymbol{x}}) \tag{29}$$

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

2.3.2. *Weighting matrix.* Using weighted least squares adds a great deal of flexibility to the least-squares system solver, in that we do not need to carefully choose neighbor sets that are both optimally minimal, and produce a well-conditioned interpolation. As the simplest choice, we use a diagonal weighting matrix W in (23), which 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]. If the volume being weighted is $\boldsymbol{j}$ and the target face is $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}_d$, the weight value $W_{\boldsymbol{j},\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ is chosen to be

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

where $D_{\boldsymbol{j},\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}$ is the Euclidean distance between the volume center and the target face center. We have found that the choice of weighting function strongly influences the locality of the resulting stencil, and thus the eigenvalues and stability of the resulting operator. Using this weighting, we find that our stencil values in the interior appear to be a perturbation off of a standard second-order stencil, while operator eigenvalues are stable despite small or missing neighboring cells near the embedded boundary.

To understand the effect of the weighting matrix power, we can apply discrete Fourier analysis [23] to the Poisson problem in a one-dimensional periodic domain. For an eigenmode $\phi = e^{i\beta x}$, the exact differential operator is $\partial_{xx}\phi = \lambda\phi$, where $\lambda = -\beta^2$. Our discrete operator based on (23) can be written as:

$$L\Phi = \frac{1}{h}(S_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} - S_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}_d})\Phi, \quad L\Phi \equiv \sum_{i=-N/2}^{N/2} s_i\phi_i \,. \tag{30}$$

From this it is straightforward to calculate the eigenvalues λ_L of our discrete operator, which we have plotted in Figure 2 for $N = 14$. Note that the weight power p in $W = D^p$ has a dramatic effect for $p \geq -4$, and relatively little effect for $p < -4$. We believe this is because the entries of Vandermonde-like matrix M in (22), which increase like $O(D^4)$, can be counteracted with $W = D^{-5}$ or greater. This allows us

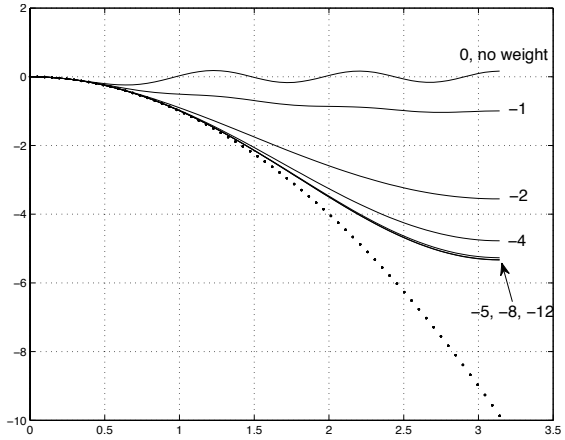


Figure 2. Discrete Fourier analysis of the effect of the weighting matrix power in one dimension, plotting wave number $\beta \in [0, \pi)$ versus operator eigenvalue λ for mode $\phi = e^{i\beta x}$. The dotted line is the exact differential operator, $\lambda = -\beta^2$. The solid lines represent the discrete operator (30) with $N = 14$ points in the stencil, and different weight powers p, for $W = D^p$.

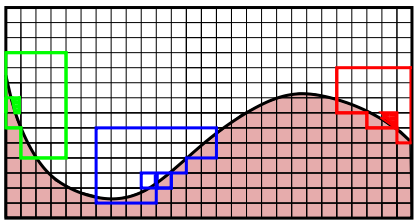


Figure 3. Neighbors of faces cut by the embedded boundary. Geometric constraints can greatly alter the number of neighbors available within a given radius

to not be concerned with how many neighbors are in our stencil, and maintains the desired accuracy and stability properties of the differential operator. This analysis can be extended to two or more dimensions, and for other operators, which will be the focus of a future paper.

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

$$\mathcal{N}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} = \left\{ \boldsymbol{j} : i_d - j_d < R \text{ or } j_d - (i_d + 1) < R \text{ for any } 1 \le d \le \mathcal{D} \right\}. \tag{31}$$

We use this many cells because we need enough cells in the system (22) so that the system will be over-determined even in the case where the embedded boundary cuts out half of the cells in the neighborhood. Figure 3 illustrates how the number of neighbor volumes can vary due to geometric constraints. We detect if there are not enough cells for any given face and, for that face, we use a larger R.

2.4. ***Multigrid algorithm.*** Once our stencils are calculated, we use the Chombo infrastructure [9; 10], which uses the Martin and Cartwright multigrid algorithm [25], to solve the system. The bottom solver for our multigrid algorithm is the PETSc algebraic multigrid solver [4; 2; 3]. For eigenvalue calculations, we use the SLEPc infrastructure [18; 17; 7]. For details of our adaptive multigrid algorithm, see Devendran et al. [12].

3. 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 the L_1, L_2, and L_∞ norms. Given an error field E^h, whose resolution is h, defined on volumes V_i in Ω, and a norm operator $\|\cdot\|$, 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 the maximum value of that field while the L_1 and L_2 norms are integral norms. These take the form

$$\begin{aligned}
\|E\|_\infty &= \max_{i\in\Omega} |E_i|,\\
\|E\|_1 &= \frac{1}{|V_\Omega|}\int_\Omega |E_i|\,dV = \frac{1}{|V_\Omega|}\sum_{i\in\Omega} |E_i||V_i|,\\
\|E\|_2 &= \left(\frac{1}{|V_\Omega|}\int_\Omega |E_i|^2\,dV\right)^{\frac{1}{2}} = \left(\frac{1}{|V_\Omega|}\sum_{i\in\Omega} |E_i|^2|V_i|\right)^{\frac{1}{2}},
\end{aligned}$$

where $|V_\Omega|$ 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:

$$T = \kappa(L(\phi^e) - L^e(\phi^e)), \tag{32}$$

where $\kappa L(\phi)$ is given in (6) and

$$L^e(\phi^e)_i = \int_i \nabla\cdot(\nabla\phi^e)\,dV. \tag{33}$$

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 (7)). The solution error ϵ is given by comparing the computed solution ϕ to

$\mathcal{D}$	norm	$\|\epsilon^{2h}\|$	ϖ	$\|\epsilon^{h}\|$
2	L_∞	$1.290\cdot10^{-4}$	2.60	$2.130\cdot10^{-5}$
2	L_1	$5.336\cdot10^{-6}$	3.99	$3.358\cdot10^{-7}$
2	L_2	$1.200\cdot10^{-5}$	3.55	$1.022\cdot10^{-6}$
3	L_∞	$9.222\cdot10^{-4}$	3.86	$6.334\cdot10^{-5}$
3	L_1	$1.071\cdot10^{-5}$	4.00	$6.687\cdot10^{-7}$
3	L_2	$2.507\cdot10^{-5}$	3.66	$1.984\cdot10^{-6}$

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 6, and $h = 1/128$.

the exact solution ϕ^e:

$$\epsilon = \phi - \phi^e. \tag{34}$$

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 $\boldsymbol{x}_0$, we use the exterior of an ellipse of the form

$$\sum_{d=1}^{\mathcal{D}} \frac{x_d^2 - x_{0,d}^2}{r_d^2} = 0, \tag{35}$$

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

$$\phi^e = \prod_{d=1}^{\mathcal{D}} \cos(\pi x_d). \tag{36}$$

3.1. ***Truncation error.*** In Table 1, we present truncation error rates for the case where the domain has Neumann boundary conditions and the irregular boundary has Dirichlet boundary conditions ($\phi|_{\partial\Omega} = \phi^e$). 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 ($\nabla\phi \cdot \hat{\boldsymbol{n}} = \nabla\phi^e \cdot \hat{\boldsymbol{n}}$). 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_∞ error 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_∞	$1.979\cdot10^{-4}$	2.99	$2.485\cdot10^{-5}$
2	L_1	$1.423\cdot10^{-5}$	3.95	$9.184\cdot10^{-7}$
2	L_2	$3.897\cdot10^{-5}$	3.46	$3.530\cdot10^{-6}$
3	L_∞	$4.490\cdot10^{-4}$	2.22	$9.645\cdot10^{-5}$
3	L_1	$2.698\cdot10^{-5}$	3.95	$1.747\cdot10^{-6}$
3	L_2	$6.697\cdot10^{-5}$	3.44	$6.161\cdot10^{-6}$

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 the ellipse shown in Figure 6, and $h = 1/128$.

$\mathcal{D}$	norm	$\|\epsilon^{2h}\|$	ϖ	$\|\epsilon^{h}\|$
2	L_∞	$1.626\cdot10^{-7}$	3.94	$1.060\cdot10^{-8}$
2	L_1	$8.934\cdot10^{-8}$	3.91	$5.952\cdot10^{-9}$
2	L_2	$1.032\cdot10^{-7}$	3.93	$6.783\cdot10^{-9}$
3	L_∞	$3.060\cdot10^{-7}$	3.97	$1.954\cdot10^{-8}$
3	L_1	$1.955\cdot10^{-7}$	3.95	$1.265\cdot10^{-8}$
3	L_2	$2.154\cdot10^{-7}$	3.96	$1.386\cdot10^{-8}$

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 $h = 1/128$.

3.2. ***Solution error.*** In [19; 20], the authors show that a method can have a lower-order truncation error on the embedded boundary (which is a codimension one set) than in the interior and still maintain the proper order for the solution error. We solve $\kappa L\phi = \kappa L(\phi^e)$ and compute the solution error. For this test ϕ^e is given by (36). In Table 3 we present solution error rates for the case where the domain has Neumann boundary conditions and the irregular boundary has Dirichlet boundary conditions ($\phi|_{\partial\Omega} = \phi_e$). We present solution error rates for the case where the domain and the irregular boundary have Neumann boundary conditions ($\nabla\phi\cdot\hat{\boldsymbol{n}} = \nabla\phi^e\cdot\hat{\boldsymbol{n}}$) 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 Section 5.1.

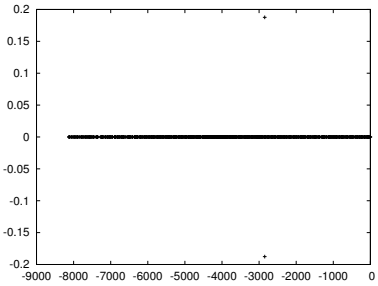
4. Operator eigenvalues

In this section, we compare the spectrum of our algorithm to the widely used, second-order algorithm presented by Schwartz et al. [32].

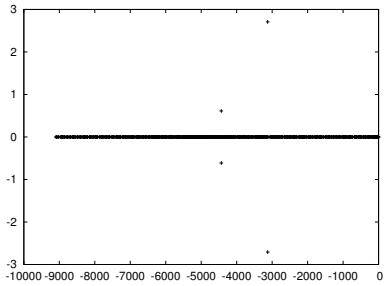
The eigenvalues of the Poisson operator will depend upon the geometry and resolution of the problem as well as the operator boundary conditions. Due to limitations in computational resources, we are only able to show the spectrum for

$\mathcal{D}$	norm	$\|\epsilon^{2h}\|$	ϖ	$\|\epsilon^{h}\|$
2	L_∞	$1.835 \cdot 10^{-7}$	3.96	$1.176 \cdot 10^{-8}$
2	L_1	$6.904 \cdot 10^{-8}$	3.95	$4.459 \cdot 10^{-9}$
2	L_2	$8.678 \cdot 10^{-8}$	3.96	$5.558 \cdot 10^{-9}$
3	L_∞	$3.879 \cdot 10^{-7}$	3.86	$2.669 \cdot 10^{-8}$
3	L_1	$9.325 \cdot 10^{-8}$	3.92	$6.175 \cdot 10^{-9}$
3	L_2	$1.315 \cdot 10^{-7}$	3.94	$8.559 \cdot 10^{-9}$

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 $h = 1/128$.



Second-order operator from [32].



Fourth-order operator.

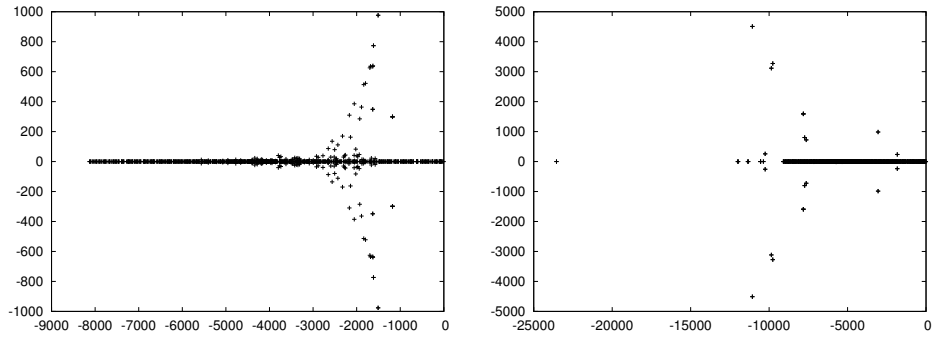
Figure 4. Plots of eigenvalues for the two-dimensional Poisson operators with Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain boundary. Real and imaginary parts on the x- and y-axes, respectively, with similar scales for each operator. The geometry implicit function is described by (35) and the resolution is 32^2.

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 Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain in Figure 4, right, We present the spectrum for the second-order operator with identical conditions in Figure 4, left. We also present the fourth-order spectrum for Dirichlet boundary conditions on the embedded boundary and Neumann boundary conditions on the domain in Figure 5, right. and the second-order spectrum in Figure 5, left. 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.

5. 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,



Second-order operator from [32]. Fourth-order operator.

Figure 5. Plots of eigenvalues for the two-dimensional Poisson operators with Dirichlet boundary conditions on the embedded boundary and Neumann boundary conditions on the domain boundary. Real and imaginary parts on the x- and y-axes, respectively, with different scales for each operator. The fourth-order operator is very similar to second-order one, but with a few eigenvalues with significantly larger real or imaginary components. The geometry implicit function is described by (35) and the resolution is 32^2.

however, one needs a sufficiently smooth description of the geometry. To achieve $O(h^P)$ accurate fluxes, Schwartz, Percelay et al. [33] show that all geometric moments in the calculation must also converge at $O(h^P)$.

Unfortunately, geometric descriptions are not always sufficiently smooth. This is not necessarily catastrophic. In [20], it is shown that large truncation errors can be ameliorated under certain circumstances; specifically, $O(1)$ 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. [32] algorithm for Poisson's equation. We compare the two algorithms using both a smooth and a nonsmooth 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

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

and a 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 in Ω at that

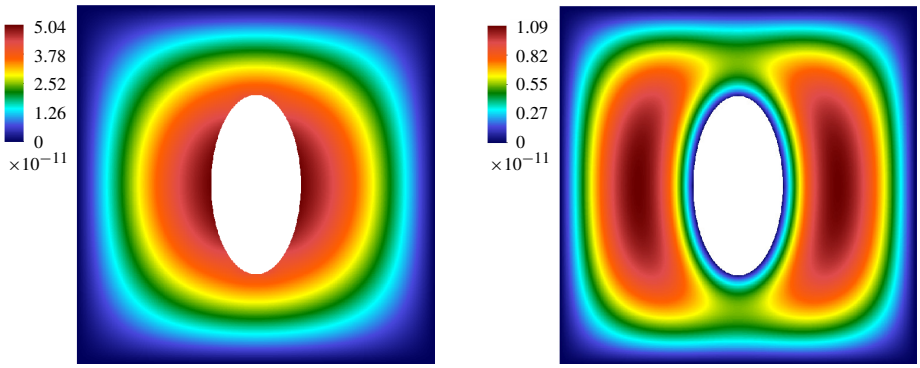


Figure 6. Solution error for the ellipse geometry in two dimensions using the current fourth-order algorithm with a resolution of 512^2. Left, using Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain boundary. Right, using Dirichlet boundary conditions both on the embedded and on the domain boundary.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	952	$1.419\cdot10^{-3}$	$3.354\cdot10^{-4}$	$4.144\cdot10^{-4}$	—
Schwartz	64^2	3752	$3.511\cdot10^{-4}$	$8.645\cdot10^{-5}$	$1.070\cdot10^{-4}$	1.95
Schwartz	128^2	14884	$8.639\cdot10^{-5}$	$2.186\cdot10^{-5}$	$2.709\cdot10^{-5}$	1.98
Schwartz	256^2	59312	$2.083\cdot10^{-5}$	$5.443\cdot10^{-6}$	$6.741\cdot10^{-6}$	2.00
Schwartz	512^2	236832	$5.167\cdot10^{-6}$	$1.354\cdot10^{-6}$	$1.677\cdot10^{-6}$	2.00
Schwartz	1024^2	946432	$1.272\cdot10^{-6}$	$3.380\cdot10^{-7}$	$4.185\cdot10^{-7}$	2.00
current	32^2	952	$2.786\cdot10^{-6}$	$1.041\cdot10^{-6}$	$1.316\cdot10^{-6}$	—
current	64^2	3752	$1.833\cdot10^{-7}$	$6.897\cdot10^{-8}$	$8.670\cdot10^{-8}$	3.91
current	128^2	14884	$1.176\cdot10^{-8}$	$4.459\cdot10^{-9}$	$5.557\cdot10^{-9}$	3.95
current	256^2	59312	$7.431\cdot10^{-10}$	$2.833\cdot10^{-10}$	$3.512\cdot10^{-10}$	3.97
current	512^2	236832	$5.045\cdot10^{-11}$	$1.941\cdot10^{-11}$	$2.396\cdot10^{-11}$	3.86

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. The convergence rates ϖ are calculated using L_1.

resolution This number of points represents the number of degrees of freedom in the calculation.

5.1. ***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 (35). Figure 6, left, shows a solution error plot with Neumann boundary conditions at the embedded boundary. Figure 6, right, shows a solution error plot with Dirichlet boundary conditions. Both cases show that the solution error is distributed throughout the domain.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	952	$5.415\cdot10^{-3}$	$1.322\cdot10^{-3}$	$1.715\cdot10^{-3}$	—
Schwartz	64^2	3752	$1.590\cdot10^{-3}$	$3.592\cdot10^{-4}$	$4.665\cdot10^{-4}$	1.87
Schwartz	128^2	14884	$4.581\cdot10^{-4}$	$9.569\cdot10^{-5}$	$1.243\cdot10^{-4}$	1.90
Schwartz	256^2	59312	$1.259\cdot10^{-4}$	$2.498\cdot10^{-5}$	$3.247\cdot10^{-5}$	1.93
Schwartz	512^2	236832	$3.430\cdot10^{-5}$	$6.449\cdot10^{-6}$	$8.381\cdot10^{-6}$	1.95
Schwartz	1024^2	946432	$9.289\cdot10^{-6}$	$1.647\cdot10^{-6}$	$2.142\cdot10^{-6}$	1.96
current	32^2	952	$7.190\cdot10^{-7}$	$3.169\cdot10^{-7}$	$3.841\cdot10^{-7}$	—
current	64^2	3752	$4.146\cdot10^{-8}$	$1.907\cdot10^{-8}$	$2.300\cdot10^{-8}$	4.05
current	128^2	14884	$2.527\cdot10^{-9}$	$1.173\cdot10^{-9}$	$1.400\cdot10^{-9}$	4.02
current	256^2	59312	$1.564\cdot10^{-10}$	$7.370\cdot10^{-11}$	$8.717\cdot10^{-11}$	3.99
current	512^2	236832	$1.093\cdot10^{-11}$	$5.195\cdot10^{-12}$	$6.109\cdot10^{-12}$	3.82

Table 6. Error vs. refinement comparison with the second-order Schwartz et al. algorithm with the elliptical geometry. This uses Dirichlet boundary conditions everywhere. The convergence rates ϖ are calculated using L_1.

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 show the expected convergence rate of 4 for both Neumann and Dirichlet boundary conditions at the irregular faces.

We also 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 (over one million degrees of freedom). We expect a different cross-over point depending on both resolution and the complexity of the boundary. Although our approach requires significantly more computation and memory, both due to setup (SVD-based solvers) and solution (using larger stencils), this impact is on a smaller-dimension domain (codimension 2 and 1 when $\mathcal{D}$ is 3 and 2, respectively) only near the embedded boundary. For a given size problem, there is likely a cross-over point where our algorithm delivers the same accuracy with many fewer total points.

5.2. ***Accuracy vs. resolution for a nonsmooth geometry.*** Now we 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 7. The implicit function is C^1 discontinuous. Figure 8, left, shows a solution error plot with Neumann boundary conditions. Figure 8, right, 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 Schwartz et al. algorithm for Dirichlet boundary

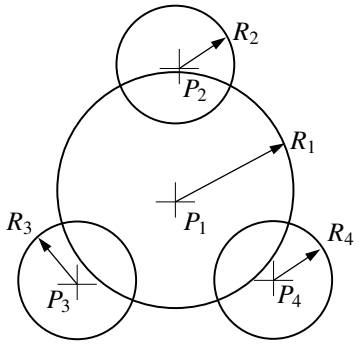


Figure 7. Diagram for our four circle geometry. The computational geometry is the region not covered by these four circles in the unit square, with centers $P_1 = (0.5, 0.5, 0.5)$, $P_2 = (0.5, 0.735, 0.5)$, $P_3 = (0.2965, 0.3825, 0.5)$, $P_4 = (0.7035, 0.3825, 0.5)$, and radii $R_1 = 0.2$, $R_2 = R_3 = R_4 = 0.1$.

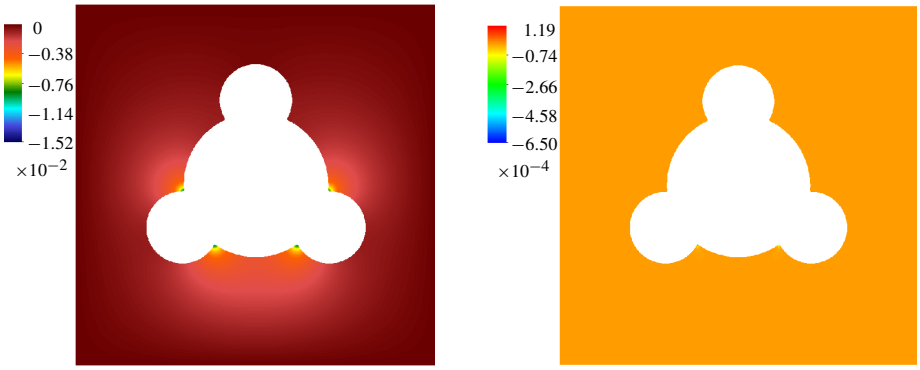


Figure 8. Solution error for the four circle geometry in two dimensions using the current fourth-order algorithm with a resolution of 512^2. Left, using Neumann boundary conditions on the embedded boundary and Dirichlet boundary conditions on the domain boundary. Right, using Dirichlet boundary conditions both on the embedded and on the domain boundary; the error is concentrated very near the cusps in the geometry.

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. For Neumann boundary conditions on the irregular faces, we do not even converge at second order. For Dirichlet boundary conditions, we show better convergence but it is generally less than fourth order.

5.3. ***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 [33] to refine the cells near circle intersections by a factor of 1024^2 in each direction. Since we

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	862	$3.525\cdot10^{-2}$	$3.006\cdot10^{-3}$	$4.625\cdot10^{-3}$	—
Schwartz	64^2	3370	$2.231\cdot10^{-2}$	$9.703\cdot10^{-4}$	$1.585\cdot10^{-3}$	1.63
Schwartz	128^2	13318	$1.291\cdot10^{-2}$	$5.649\cdot10^{-4}$	$1.134\cdot10^{-3}$	0.78
Schwartz	256^2	52930	$1.776\cdot10^{-3}$	$8.885\cdot10^{-5}$	$1.325\cdot10^{-4}$	2.66
Schwartz	512^2	211136	$3.576\cdot10^{-3}$	$1.403\cdot10^{-4}$	$2.192\cdot10^{-4}$	-0.66
Schwartz	1024^2	843316	$3.033\cdot10^{-3}$	$1.417\cdot10^{-4}$	$2.089\cdot10^{-4}$	-0.01
current	32^2	862	$5.751\cdot10^{-2}$	$4.869\cdot10^{-3}$	$7.208\cdot10^{-3}$	—
current	64^2	3370	$3.096\cdot10^{-2}$	$1.420\cdot10^{-3}$	$2.721\cdot10^{-3}$	1.77
current	128^2	13318	$2.817\cdot10^{-2}$	$2.132\cdot10^{-3}$	$3.204\cdot10^{-3}$	-0.58
current	256^2	52930	$1.969\cdot10^{-2}$	$1.383\cdot10^{-3}$	$2.020\cdot10^{-3}$	0.62
current	512^2	211136	$1.517\cdot10^{-2}$	$6.754\cdot10^{-4}$	$1.042\cdot10^{-3}$	1.03

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. The convergence rates ϖ are calculated using L_1.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	862	$2.191\cdot10^{-2}$	$1.333\cdot10^{-3}$	$1.750\cdot10^{-3}$	—
Schwartz	64^2	3370	$1.026\cdot10^{-2}$	$3.717\cdot10^{-4}$	$4.908\cdot10^{-4}$	1.84
Schwartz	128^2	13318	$2.850\cdot10^{-3}$	$9.703\cdot10^{-5}$	$1.300\cdot10^{-4}$	1.93
Schwartz	256^2	52930	$5.719\cdot10^{-4}$	$2.654\cdot10^{-5}$	$3.491\cdot10^{-5}$	1.87
Schwartz	512^2	211136	$8.178\cdot10^{-4}$	$6.686\cdot10^{-6}$	$9.117\cdot10^{-6}$	1.98
Schwartz	1024^2	843316	$8.979\cdot10^{-4}$	$1.620\cdot10^{-6}$	$2.330\cdot10^{-6}$	2.04
current	32^2	862	$1.818\cdot10^{-2}$	$1.604\cdot10^{-4}$	$5.435\cdot10^{-4}$	—
current	64^2	3370	$2.797\cdot10^{-3}$	$3.228\cdot10^{-5}$	$1.089\cdot10^{-4}$	2.31
current	128^2	13318	$2.317\cdot10^{-2}$	$4.557\cdot10^{-6}$	$7.391\cdot10^{-5}$	2.82
current	256^2	52930	$2.705\cdot10^{-3}$	$2.014\cdot10^{-7}$	$4.966\cdot10^{-6}$	4.49
current	512^2	211136	$6.502\cdot10^{-4}$	$5.940\cdot10^{-8}$	$2.263\cdot10^{-6}$	1.76

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.The convergence rates ϖ are calculated using L_1.

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 truncation error T (defined in (32)) and the solution error ϵ (defined in (34)), the error equation can be written

$$L(\epsilon) = T. \tag{37}$$

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 nonsingular component of the truncation error to be $T^{\mathrm{ns}} = T - T^{\mathrm{s}}$. We then compute the convergence rate of the solution error ϵ^{ns} in the absence of the singular points of the truncation error by solving

$$L\epsilon^{\mathrm{ns}} = T^{\mathrm{ns}} \tag{38}$$

with the appropriate homogeneous boundary conditions.

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

$$T^{\mathrm{s}}_{\boldsymbol{v}} = \begin{cases} T & \text{if } \boldsymbol{v} \in \mathcal{V}^{\mathrm{s}}, \\ 0 & \text{otherwise.} \end{cases} \tag{39}$$

We solve (38) using both the current fourth-order algorithm and the second-order Schwartz et al. algorithm. The comparisons with 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. Once we remove the singular part of the truncation error, we once again show consistent fourth-order accuracy with both Dirichlet and Neumann boundary conditions on the irregular faces.

6. 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 5.3 are predicated upon knowing a priori the singular points and the exact solution. 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.

algorithm	resolution	# points	$L_\infty(\epsilon^{ns})$	$L_1(\epsilon^{ns})$	$L_2(\epsilon^{ns})$	ϖ
Schwartz	32^2	862	$1.329\cdot10^{-3}$	$3.557\cdot10^{-4}$	$4.370\cdot10^{-4}$	—
Schwartz	64^2	3370	$2.881\cdot10^{-4}$	$8.740\cdot10^{-5}$	$1.073\cdot10^{-4}$	2.02
Schwartz	128^2	13320	$7.068\cdot10^{-5}$	$2.084\cdot10^{-5}$	$2.552\cdot10^{-5}$	2.06
Schwartz	256^2	52930	$1.777\cdot10^{-5}$	$5.171\cdot10^{-6}$	$6.327\cdot10^{-6}$	2.01
Schwartz	512^2	211136	$4.382\cdot10^{-6}$	$1.278\cdot10^{-6}$	$1.564\cdot10^{-6}$	2.01
Schwartz	1024^2	843316	$1.033\cdot10^{-6}$	$3.222\cdot10^{-7}$	$3.946\cdot10^{-7}$	1.98
current	32^2	862	$2.353\cdot10^{-6}$	$7.242\cdot10^{-7}$	$8.939\cdot10^{-7}$	—
current	64^2	3370	$1.864\cdot10^{-7}$	$5.838\cdot10^{-8}$	$7.354\cdot10^{-8}$	3.63
current	128^2	13318	$1.133\cdot10^{-8}$	$3.783\cdot10^{-9}$	$4.735\cdot10^{-9}$	3.94
current	256^2	52930	$7.215\cdot10^{-10}$	$2.405\cdot10^{-10}$	$2.993\cdot10^{-10}$	3.97
current	512^2	211136	$5.392\cdot10^{-11}$	$1.649\cdot10^{-11}$	$2.046\cdot10^{-11}$	3.86

Table 9. Convergence of the nonsingular 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.The convergence rates ϖ are calculated using L_1.

algorithm	resolution	# points	$L_\infty(\epsilon^{ns})$	$L_1(\epsilon^{ns})$	$L_2(\epsilon^{ns})$	ϖ
Schwartz	32^2	862	$6.121\cdot10^{-3}$	$1.440\cdot10^{-3}$	$1.875\cdot10^{-3}$	—
Schwartz	64^2	3370	$1.552\cdot10^{-3}$	$3.911\cdot10^{-4}$	$5.066\cdot10^{-4}$	1.88
Schwartz	128^2	13320	$4.119\cdot10^{-4}$	$1.002\cdot10^{-4}$	$1.321\cdot10^{-4}$	1.96
Schwartz	256^2	52930	$1.355\cdot10^{-4}$	$2.669\cdot10^{-5}$	$3.515\cdot10^{-5}$	1.90
Schwartz	512^2	211136	$3.215\cdot10^{-5}$	$6.797\cdot10^{-6}$	$8.913\cdot10^{-6}$	1.97
Schwartz	1024^2	843316	$9.172\cdot10^{-6}$	$1.640\cdot10^{-6}$	$2.152\cdot10^{-6}$	2.05
current	32^2	862	$5.126\cdot10^{-7}$	$1.741\cdot10^{-7}$	$2.236\cdot10^{-7}$	—
current	64^2	3370	$2.684\cdot10^{-8}$	$1.080\cdot10^{-8}$	$1.338\cdot10^{-8}$	4.01
current	128^2	13318	$1.586\cdot10^{-9}$	$6.380\cdot10^{-10}$	$7.764\cdot10^{-10}$	4.08
current	256^2	52930	$9.650\cdot10^{-11}$	$3.882\cdot10^{-11}$	$4.683\cdot10^{-11}$	4.03
current	512^2	211136	$6.676\cdot10^{-12}$	$2.693\cdot10^{-12}$	$3.232\cdot10^{-12}$	3.84

Table 10. Convergence of the nonsingular part of the solution error vs. refinement for both the current algorithm and the Schwartz et al. algorithm. This uses Dirichlet boundary conditions everywhere.The convergence rates ϖ are calculated using L_1.

6.1. ***Smoothing the geometric description.*** Recall that, to generate our geometric moments using the algorithm described in [33], we must start with an implicit function $I(\boldsymbol{x})$ whose zero surface (or contour, in two dimensions) forms the embedded boundary. Consider the geometry described in Figure 7. The implicit function for each circle C_i, with radius r_i and center $\boldsymbol{y}_i$ is given by

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

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

$$I(\boldsymbol{x}) = \max_{1<=d<=4} C_i(\boldsymbol{x}). \tag{40}$$

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 [34] presents an alternative approach called R-functions (named for V. L. Rvačev, the originator of the concept [31]), in which logical functions such as maxima, minima and absolute values are replaced by differentiable functions. Though this method is far more numerically tractable, the functions that Shapiro presents do not have a well-defined length scale over which they smooth. The smoothing method described here provides both a well-defined smoothing length and is numerically tractable.

One way to write the maxima function used in (40) is by using an absolute value:

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

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

$$\max_\delta(a, b) = \tfrac{1}{2}(a + b + A_\delta(a - b)),$$

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

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

Since our algorithm is fourth order in fluxes, we use geometric moments to fourth order. The algorithm in [33] requires that the implicit function 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 is

$$\psi_\delta(x) = \begin{cases} \dfrac{4}{3\delta}\cos^4\left(\dfrac{\pi x}{2\delta}\right) & \text{if } -\delta \le x \le \delta, \\ 0 & \text{otherwise.} \end{cases}$$

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	862	$1.989\cdot10^{-2}$	$2.026\cdot10^{-3}$	$3.003\cdot10^{-3}$	—
Schwartz	64^2	3368	$2.593\cdot10^{-3}$	$2.686\cdot10^{-4}$	$3.564\cdot10^{-4}$	2.91
Schwartz	128^2	13316	$1.171\cdot10^{-3}$	$1.207\cdot10^{-4}$	$1.657\cdot10^{-4}$	1.15
Schwartz	256^2	52916	$2.522\cdot10^{-4}$	$3.012\cdot10^{-5}$	$4.093\cdot10^{-5}$	2.00
Schwartz	512^2	211062	$6.144\cdot10^{-5}$	$7.472\cdot10^{-6}$	$1.014\cdot10^{-5}$	2.01
Schwartz	1024^2	843004	$1.417\cdot10^{-5}$	$1.798\cdot10^{-6}$	$2.436\cdot10^{-6}$	2.05
current	32^2	862	$1.525\cdot10^{-1}$	$1.166\cdot10^{-2}$	$1.905\cdot10^{-2}$	—
current	64^2	3368	$1.739\cdot10^{-3}$	$5.812\cdot10^{-5}$	$1.102\cdot10^{-4}$	7.64
current	128^2	13316	$7.054\cdot10^{-5}$	$3.077\cdot10^{-6}$	$5.886\cdot10^{-6}$	4.23
current	256^2	52916	$3.593\cdot10^{-6}$	$4.053\cdot10^{-8}$	$8.884\cdot10^{-8}$	6.24
current	512^2	211062	$1.425\cdot10^{-7}$	$9.227\cdot10^{-9}$	$1.473\cdot10^{-8}$	2.13

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 on the domain boundary and Neumann on the embedded boundary. Here we set the geometric regularization length to a constant $\delta = 0.01$. The convergence rates ϖ are calculated using L_1.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	862	$1.184\cdot10^{-2}$	$1.869\cdot10^{-3}$	$2.522\cdot10^{-3}$	—
Schwartz	64^2	3368	$1.715\cdot10^{-3}$	$4.142\cdot10^{-4}$	$5.414\cdot10^{-4}$	2.17
Schwartz	128^2	13316	$5.922\cdot10^{-4}$	$1.023\cdot10^{-4}$	$1.356\cdot10^{-4}$	2.01
Schwartz	256^2	52916	$1.355\cdot10^{-4}$	$2.676\cdot10^{-5}$	$3.527\cdot10^{-5}$	1.93
Schwartz	512^2	211062	$3.215\cdot10^{-5}$	$6.784\cdot10^{-6}$	$8.892\cdot10^{-6}$	1.97
Schwartz	1024^2	843004	$8.063\cdot10^{-6}$	$1.644\cdot10^{-6}$	$2.159\cdot10^{-6}$	2.04
current	32^2	862	$7.904\cdot10^{-3}$	$4.768\cdot10^{-5}$	$2.992\cdot10^{-4}$	—
current	64^2	3368	$9.380\cdot10^{-5}$	$1.418\cdot10^{-6}$	$4.421\cdot10^{-6}$	5.07
current	128^2	13316	$2.921\cdot10^{-6}$	$9.434\cdot10^{-9}$	$5.098\cdot10^{-8}$	7.23
current	256^2	52916	$2.745\cdot10^{-7}$	$2.839\cdot10^{-10}$	$2.146\cdot10^{-9}$	5.05
current	512^2	211062	$3.223\cdot10^{-9}$	$3.365\cdot10^{-12}$	$3.125\cdot10^{-11}$	6.39

Table 12. 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 $\delta = 0.01$. The convergence rates ϖ are calculated using L_1.

which 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 of the absolute value function.

6.2. ***Regularization length scale and grid refinement.*** Now we investigate how one picks the length scale δ. For the piecewise-smooth geometric description presented in Section 5.2, we present four different choices for δ and see how the accuracy changes with grid refinement. First we use a constant $\delta = 0.01$. Second,

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	828	$6.864\cdot10^{-3}$	$1.503\cdot10^{-3}$	$1.960\cdot10^{-3}$	—
Schwartz	64^2	3238	$1.628\cdot10^{-3}$	$3.894\cdot10^{-4}$	$5.070\cdot10^{-4}$	1.94
Schwartz	128^2	12810	$5.351\cdot10^{-4}$	$1.014\cdot10^{-4}$	$1.345\cdot10^{-4}$	1.94
Schwartz	256^2	50918	$1.404\cdot10^{-4}$	$2.669\cdot10^{-5}$	$3.516\cdot10^{-5}$	1.92
Schwartz	512^2	203052	$4.085\cdot10^{-5}$	$6.808\cdot10^{-6}$	$8.933\cdot10^{-6}$	1.97
Schwartz	1024^2	810964	$9.834\cdot10^{-6}$	$1.640\cdot10^{-6}$	$2.153\cdot10^{-6}$	2.05
current	32^2	828	$9.448\cdot10^{-5}$	$1.965\cdot10^{-6}$	$5.407\cdot10^{-6}$	—
current	64^2	3326	$9.659\cdot10^{-7}$	$1.622\cdot10^{-8}$	$4.080\cdot10^{-8}$	6.92
current	128^2	13266	$2.179\cdot10^{-8}$	$8.458\cdot10^{-10}$	$1.295\cdot10^{-9}$	4.26
current	256^2	52886	$2.110\cdot10^{-8}$	$9.797\cdot10^{-11}$	$2.689\cdot10^{-10}$	3.10
current	512^2	211088	$1.028\cdot10^{-8}$	$2.417\cdot10^{-11}$	$1.331\cdot10^{-10}$	2.01

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 $\delta = 4h$. The convergence rates ϖ are calculated using L_1.

we choose delta to vary linearly with h ($\delta = 4h$). Third, we choose $\delta = \sqrt{R_1 h}$, where $R_1 = 0.2$ is the radius of the largest circle in Figure 7. Finally, we choose

$$\delta = 0.1\sqrt[4]{R_1^3 h}.$$

The convergence rates for the four choices are quite different.

First, we set our geometric regularization length to a constant $\delta = 0.01$. Tables 11 and 12 show error rates for Neumann and Dirichlet 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. We also show excellent convergence rates with both Neumann and Dirichlet boundary conditions at the irregular faces.

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. Our convergence rates in this case (again, especially with Neumann boundary conditions), are erratic.

Next, we set our geometric regularization length to $\delta = \sqrt{R_1 h}$. Tables 15 and 16 show the error rates for Neumann and Dirichlet 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. Our convergence rates here for Dirichlet boundary conditions at the embedded boundary are fourth order. For Neumann boundary conditions at the irregular faces, our convergence rates here are more erratic.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	828	$1.232\cdot10^{-3}$	$3.357\cdot10^{-4}$	$4.141\cdot10^{-4}$	—
Schwartz	64^2	3238	$4.626\cdot10^{-4}$	$1.379\cdot10^{-4}$	$1.736\cdot10^{-4}$	1.28
Schwartz	128^2	12810	$2.328\cdot10^{-4}$	$5.009\cdot10^{-5}$	$6.474\cdot10^{-5}$	1.46
Schwartz	256^2	50918	$1.477\cdot10^{-4}$	$2.196\cdot10^{-5}$	$2.940\cdot10^{-5}$	1.19
Schwartz	512^2	203052	$8.893\cdot10^{-5}$	$9.725\cdot10^{-6}$	$1.335\cdot10^{-5}$	1.17
Schwartz	1024^2	810964	$4.877\cdot10^{-5}$	$4.166\cdot10^{-6}$	$5.785\cdot10^{-6}$	1.22
current	32^2	828	$1.683\cdot10^{-3}$	$1.122\cdot10^{-4}$	$2.043\cdot10^{-4}$	—
current	64^2	3326	$6.971\cdot10^{-6}$	$5.483\cdot10^{-7}$	$9.162\cdot10^{-7}$	7.67
current	128^2	13266	$6.512\cdot10^{-7}$	$6.887\cdot10^{-8}$	$9.896\cdot10^{-8}$	2.99
current	256^2	52886	$8.852\cdot10^{-7}$	$7.875\cdot10^{-8}$	$1.125\cdot10^{-7}$	-0.193
current	512^2	211088	$4.898\cdot10^{-7}$	$3.394\cdot10^{-8}$	$5.000\cdot10^{-8}$	1.21

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 $\delta = 4h$. The convergence rates ϖ are calculated using L_1.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	846	$1.578\cdot10^{-3}$	$5.115\cdot10^{-4}$	$6.401\cdot10^{-4}$	—
Schwartz	64^2	3312	$5.006\cdot10^{-4}$	$1.460\cdot10^{-4}$	$1.845\cdot10^{-4}$	1.80
Schwartz	128^2	13088	$2.112\cdot10^{-4}$	$4.692\cdot10^{-5}$	$6.028\cdot10^{-5}$	1.63
Schwartz	256^2	52022	$8.354\cdot10^{-5}$	$1.484\cdot10^{-5}$	$1.940\cdot10^{-5}$	1.66
Schwartz	512^2	20751	$3.258\cdot10^{-5}$	$4.992\cdot10^{-6}$	$6.646\cdot10^{-6}$	1.57
Schwartz	1024^2	82879	$1.000\cdot10^{-5}$	$1.449\cdot10^{-6}$	$1.944\cdot10^{-6}$	1.78
current	32^2	846	$6.139\cdot10^{-5}$	$4.725\cdot10^{-6}$	$8.113\cdot10^{-6}$	—
current	64^2	3330	$1.274\cdot10^{-6}$	$8.435\cdot10^{-8}$	$1.691\cdot10^{-7}$	5.80
current	128^2	13252	$4.698\cdot10^{-7}$	$4.297\cdot10^{-8}$	$6.226\cdot10^{-8}$	0.973
current	256^2	52794	$8.422\cdot10^{-8}$	$7.356\cdot10^{-9}$	$1.057\cdot10^{-8}$	2.54
current	512^2	210856	$2.127\cdot10^{-8}$	$1.846\cdot10^{-9}$	$2.645\cdot10^{-9}$	1.99

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 on the domain boundary and Neumann on the embedded boundary. Here we set the geometric regularization length to $\delta = \sqrt{R_1 h}$. The convergence rates ϖ are calculated using L_1.

Finally we set geometric regularization length to $\delta = 0.1\sqrt[4]{R_1^3 h}$. Tables 17 and 18 show the error rates for Dirichlet and Neumann boundary conditions at the cut faces, respectively. We see excellent convergence rates and error values for both types of boundary condition.

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 as well as its boundary conditions.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	846	$6.581\cdot10^{-3}$	$1.496\cdot10^{-3}$	$1.993\cdot10^{-3}$	—
Schwartz	64^2	3312	$1.890\cdot10^{-3}$	$4.014\cdot10^{-4}$	$5.254\cdot10^{-4}$	1.89
Schwartz	128^2	13088	$4.148\cdot10^{-4}$	$9.815\cdot10^{-5}$	$1.295\cdot10^{-4}$	2.03
Schwartz	256^2	52022	$1.355\cdot10^{-4}$	$2.626\cdot10^{-5}$	$3.447\cdot10^{-5}$	1.90
Schwartz	512^2	207510	$3.214\cdot10^{-5}$	$6.751\cdot10^{-6}$	$8.842\cdot10^{-6}$	1.95
Schwartz	1024^2	828794	$8.573\cdot10^{-6}$	$1.643\cdot10^{-6}$	$2.157\cdot10^{-6}$	2.03
current	32^2	846	$5.402\cdot10^{-6}$	$2.143\cdot10^{-7}$	$3.879\cdot10^{-7}$	—
current	64^2	3330	$2.792\cdot10^{-7}$	$1.039\cdot10^{-8}$	$1.807\cdot10^{-8}$	4.36
current	128^2	13252	$1.421\cdot10^{-8}$	$5.089\cdot10^{-10}$	$7.016\cdot10^{-10}$	4.35
current	256^2	52794	$2.260\cdot10^{-9}$	$3.414\cdot10^{-11}$	$7.429\cdot10^{-11}$	3.89
current	512^2	210856	$4.074\cdot10^{-10}$	$2.406\cdot10^{-12}$	$5.238\cdot10^{-12}$	3.82

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 everywhere. Here we set the geometric regularization length to $\delta = \sqrt{R_1 h}$. The convergence rates ϖ are calculated using L_1.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	846	$8.863\cdot10^{-3}$	$1.775\cdot10^{-3}$	$2.356\cdot10^{-3}$	—
Schwartz	64^2	3312	$1.700\cdot10^{-3}$	$4.134\cdot10^{-4}$	$5.400\cdot10^{-4}$	2.10
Schwartz	128^2	13088	$6.486\cdot10^{-4}$	$1.026\cdot10^{-4}$	$1.361\cdot10^{-4}$	2.01
Schwartz	256^2	52022	$1.753\cdot10^{-4}$	$2.692\cdot10^{-5}$	$3.557\cdot10^{-5}$	1.93
Schwartz	512^2	207510	$3.698\cdot10^{-5}$	$6.802\cdot10^{-6}$	$8.921\cdot10^{-6}$	1.98
Schwartz	1024^2	828794	$1.002\cdot10^{-5}$	$1.640\cdot10^{-6}$	$2.152\cdot10^{-6}$	2.05
current	32^2	846	$2.322\cdot10^{-3}$	$1.896\cdot10^{-5}$	$9.508\cdot10^{-5}$	—
current	64^2	3330	$8.003\cdot10^{-5}$	$1.197\cdot10^{-6}$	$3.742\cdot10^{-6}$	3.98
current	128^2	13252	$3.913\cdot10^{-6}$	$7.621\cdot10^{-9}$	$6.239\cdot10^{-8}$	7.29
current	256^2	52794	$6.676\cdot10^{-7}$	$7.522\cdot10^{-10}$	$6.327\cdot10^{-9}$	3.34
current	512^2	210856	$5.136\cdot10^{-8}$	$5.920\cdot10^{-11}$	$4.836\cdot10^{-10}$	3.66

Table 17. 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 $\delta = 0.1\sqrt[4]{R_1^3 h}$. The convergence rates ϖ are calculated using L_1.

7. 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 algorithm but is much more accurate with a sufficiently smooth geometric description. We show that the effect of geometric discontinuities on error rates can be profound.

algorithm	resolution	# points	$L_\infty(\epsilon)$	$L_1(\epsilon)$	$L_2(\epsilon)$	ϖ
Schwartz	32^2	846	$9.291\cdot10^{-3}$	$1.131\cdot10^{-3}$	$1.540\cdot10^{-3}$	—
Schwartz	64^2	3312	$2.222\cdot10^{-3}$	$2.422\cdot10^{-4}$	$3.190\cdot10^{-4}$	2.22
Schwartz	128^2	13088	$1.348\cdot10^{-3}$	$1.408\cdot10^{-4}$	$1.940\cdot10^{-4}$	7.82
Schwartz	256^2	52022	$3.738\cdot10^{-4}$	$4.130\cdot10^{-5}$	$5.694\cdot10^{-5}$	1.76
Schwartz	512^2	207510	$1.141\cdot10^{-4}$	$1.087\cdot10^{-5}$	$1.500\cdot10^{-5}$	1.92
Schwartz	1024^2	828794	$3.202\cdot10^{-5}$	$3.167\cdot10^{-6}$	$4.370\cdot10^{-6}$	1.77
current	32^2	846	$4.085\cdot10^{-2}$	$3.202\cdot10^{-3}$	$5.722\cdot10^{-3}$	—
current	64^2	3330	$1.254\cdot10^{-3}$	$3.802\cdot10^{-5}$	$7.107\cdot10^{-5}$	6.39
current	128^2	13252	$1.189\cdot10^{-4}$	$6.829\cdot10^{-6}$	$1.186\cdot10^{-5}$	2.47
current	256^2	52794	$1.318\cdot10^{-5}$	$6.356\cdot10^{-7}$	$1.223\cdot10^{-6}$	3.42
current	512^2	210856	$1.134\cdot10^{-6}$	$5.400\cdot10^{-8}$	$8.949\cdot10^{-8}$	3.55

Table 18. 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 $\delta = 0.1\sqrt[4]{R_1^3 h}$. The convergence rates ϖ are calculated using L_1.

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. We present one such refinement scheme which performs quite well for both Neumann and Dirichlet boundary conditions at cut faces.

Acknowledgment

The authors would like to thank Dr. Phillip Colella for his technical advice and insight.

References

[1] M. J. Aftosmis, M. J. Berger, and J. E. Melton, *Robust and efficient cartesian mesh generation for component-based geometry*, AIAA Journal **36** (1998), no. 6, 952–960.

[2] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, and H. Zhang, *PETSc users manual*, Tech. Report ANL-95/11 - Revision 3.5, Argonne National Laboratory, 2014.

[3] ———, *PETSc Web page*, 2014.

[4] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, Modern software tools in scientific computing (E. Arge, A. M. Bruaset, and H. P. Langtangen, eds.), Birkhäuser, 1997, pp. 163–202. Zbl

[5] V. A. Barker, L. S. Blackford, J. Dongarra, J. Du Croz, S. Hammarling, M. Marinova, J. Waśniewski, and P. Yalamov, *LAPACK95 users' guide*, Software, Environments, and Tools, no. 13, SIAM, Philadelphia, PA, 2001. MR Zbl

[6] S. C. Brenner and L. R. Scott, *The mathematical theory of finite element methods*, 3rd ed., Texts in Applied Mathematics, no. 15, Springer, New York, 2008. MR

[7] C. Campos, J. E. Roman, E. Romero, and A. Tomas, *SLEPc users manual*, Tech. 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*, J. Comput. Phys. **155** (1999), no. 2, 468–498. MR Zbl

[9] P. Colella, D. T. Graves, T. J. Ligocki, D. F. Martin, D. Modiano, D. B. Serafini, and B. V. Straalen, *Chombo software package for AMR applications: design document*, Tech. Report LBNL-6616E, LBNL, July 2014.

[10] P. Colella, D. T. Graves, T. J. Ligocki, G. Miller, D. Modiano, P. Schwartz, B. V. Straalen, J. Pillod, D. Trebotich, and M. Barad, *EBChombo software package for Cartesian grid, embedded boundary application*, Tech. Report LBNL-6615E, LBNL, 2014.

[11] P. Colella, D. T. Graves, B. J. Keen, and D. Modiano, *A Cartesian grid embedded boundary method for hyperbolic conservation laws*, J. Comput. Phys. **211** (2006), no. 1, 347–366. MR Zbl

[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*, Tech. Report LBNL-1004329, LBNL, 2014.

[13] Z. Dragojlovic, F. Najmabadi, and M. Day, *An embedded boundary method for viscous, conducting compressible flow*, J. Comput. Phys. **216** (2006), no. 1, 37–51. MR Zbl

[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** (2005), no. 2, 577–601. MR Zbl

[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*, Commun. Appl. Math. Comput. Sci. **8** (2013), no. 1, 99–122. MR Zbl

[16] L. Greengard and J.-Y. Lee, *A direct adaptive Poisson solver of arbitrary order accuracy*, J. Comput. Phys. **125** (1996), no. 2, 415–424. MR Zbl

[17] V. Hernández, J. E. Román, and V. Vidal, *SLEPc: scalable library for eigenvalue problem computations*, High performance computing for computational science (Berlin) (J. M. L. M. Palma, A. A. Sousa, J. Dongarra, and V. Hernández, eds.), Lecture Notes in Computer Science, no. 2565, Springer, 2003, pp. 377–391. Zbl

[18] V. Hernandez, J. E. Roman, and V. Vidal, *SLEPc: a scalable and flexible toolkit for the solution of eigenvalue problems*, ACM Trans. Math. Software **31** (2005), no. 3, 351–362. MR

[19] H. Johansen and P. Colella, *A Cartesian grid embedded boundary method for Poisson's equation on irregular domains*, J. Comput. Phys. **147** (1998), no. 1, 60–85. MR Zbl

[20] H. S. Johansen, *Cartesian grid embedded boundary finite difference methods for elliptic and parabolic partial differential equations on irregular domains*, Ph.D. thesis, University of California, Berkeley, 1997.

[21] L. D. Landau and E. M. Lifshitz, *Fluid mechanics*, 2nd ed., Course of Theoretical Physics, no. 6, Pergamon, Oxford, 1987. MR Zbl

[22] R. J. LeVeque, *Numerical methods for conservation laws*, Birkhäuser, Basel, 1990. MR Zbl

[23] ———, *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, SIAM, Philadelphia, 2007. MR Zbl

[24] R. J. LeVeque and Z. L. Li, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal. **31** (1994), no. 4, 1019–1044. MR Zbl

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

[26] A. McKenney, L. Greengard, and A. Mayo, *A fast Poisson solver for complex geometries*, J. Comput. Phys. **118** (1995), no. 2, 348–355. MR Zbl

[27] G. H. Miller and D. Trebotich, *An embedded boundary method for the Navier–Stokes equations on a time-dependent domain*, Commun. Appl. Math. Comput. Sci. **7** (2012), no. 1, 1–31. MR Zbl

[28] A. Nonaka, D. Trebotich, G. Miller, D. Graves, and P. Colella, *A higher-order upwind method for viscoelastic flow*, Commun. Appl. Math. Comput. Sci. **4** (2009), 57–83. MR Zbl

[29] 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** (1995), no. 2, 278–304. MR Zbl

[30] S. Z. Pirzadeh, *Advanced unstructured grid generation for complex aerodynamic applications*, AIAA Journal **48** (2010), no. 5, 904–915.

[31] V. L. Rvačev, *An analytic description of certain geometric objects*, Dokl. Akad. Nauk SSSR **153** (1963), 765–767, In Russian; translated in Soviet Math. Dokl. **4** (1963), 1750–1753. MR Zbl

[32] 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** (2006), no. 2, 531–550. MR Zbl

[33] P. Schwartz, J. Percelay, T. J. Ligocki, H. Johansen, D. T. Graves, D. Devendran, P. Colella, and E. Ateljevich, *High-accuracy embedded boundary grid generation using the divergence theorem*, Commun. Appl. Math. Comput. Sci. **10** (2015), no. 1, 83–96. MR Zbl

[34] V. Shapiro, *Semi-analytic geometry with R-functions*, Acta Numer. **16** (2007), 239–303. MR Zbl

[35] G. Strang, *Linear algebra and its applications*, Academic, New York, 1976. MR Zbl

[36] E. Tadmor and J. Tanner, *Adaptive mollifiers for high resolution recovery of piecewise smooth data from its spectral information*, Found. Comput. Math. **2** (2002), no. 2, 155–189. MR Zbl

[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** (2008), no. 7, 3749–3756.

Received March 25, 2016. Revised December 19, 2016.

DHARSHI DEVENDRAN: pdevendran@gmail.com
Applied Numerical Algorithms Group (ANAG), Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

DANIEL T. GRAVES: DTGraves@lbl.gov
Computational Research Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

HANS JOHANSEN: hjohansen@lbl.gov
Applied Numerical Algorithms Group (ANAG), Computational Research Division, Lawrence Berkeley National Laboratory, MS 50A1148, One Cyclotron Road, Berkeley, CA 94720, United States

TERRY LIGOCKI: TJLigocki@lbl.gov
Applied Numerical Algorithms Group (ANAG), Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States