LBNL-39908
UC-405

ERNEST ORLANDO LAWRENCE
BERKELEY NATIONAL LABORATORY

A Cartesian Grid Embedded Boundary Method for Poisson's Equation on Irregular Domains

Hans Johansen and Phillip Colella
Computing Sciences Directorate

January 1997

DISCLAIMER

LBNL-39908
UC-405

A Cartesian Grid Embedded Boundary Method for Poisson's Equation on Irregular Domains *

Hans Johansen
Department of Mechanical Engineering
University of California
Berkeley, CA 94720
Phillip Colella †
Center for Computational Sciences and Engineering
Lawrence Berkeley National Laboratory
Berkeley, CA 94720

January 31, 1997

Abstract

We present a numerical method for solving Poisson's equation, with variable coefficients and Dirichlet boundary conditions, on two-dimensional regions. The approach uses a finite-volume discretization, which embeds the domain in a regular Cartesian grid. We treat the solution as a cell-centered quantity, even when those centers are outside the domain. Cells that contain a portion of the domain boundary use conservative differencing of second-order accurate fluxes, on each cell volume. The calculation of the boundary flux ensures that the conditioning of the matrix is relatively unaffected by small cell volumes. This allows us to use multi-grid iterations with a simple point relaxation strategy. We have combined this with an adaptive mesh refinement (AMR) procedure. We provide evidence that the algorithm is second-order accurate on various exact solutions, and compare the adaptive and non-adaptive calculations.

1 Introduction

In this paper we present a numerical method for solving the variable-coefficient Poisson equation, with Dirichlet boundary conditions,

$$\nabla\cdot\beta\nabla\phi = \rho \text{ on } \Omega,\ \phi = g \text{ on } \partial\Omega, \tag{1}$$

on a bounded two-dimensional region Ω, where $\beta = \beta(x, y) > 0$. Our approach uses a finite-volume discretization, which embeds the domain in a regular Cartesian grid. We treat the solution as cell-centered on a rectangular grid, even when the cell centers are outside the domain. We discretize (1)

*Research supported at UC Berkeley by the US Department of Energy Mathematical, Computing, and Information Sciences Division, Grants DE-FG03-94ER25205, DE-FG03-92ER25140, and by the US Air Force Office of Scientific Research, AASERT Grant No. F49620-93-1-0521; and at the Lawrence Berkeley National Laboratory by the US Department of Energy Mathematical, Computing, and Information Sciences Division, contract DE-AC03-76SF00098.

†Professor in Residence, Mechanical Engineering Department, UC Berkeley

on each cell by applying the divergence theorem on the intersection of that cell with Ω. This leads to a conservative, finite–volume discretization on the cells that intersect $\partial\Omega$. Thus, the discretized operator is centered at the centroids of partially covered cells, in contrast to the solution values, which are centered on the rectangular grid. The fluxes at the cell edges are computed using second–order accurate differences of the cell-centered values of the solution. In cells away from the boundary, the algorithm reduces to the standard five-point discretization for (1), with a truncation error that is second order in the mesh spacing. On the boundary, this discretization results in a first-order truncation error; however, this boundary truncation error induces a solution error that is third-order in the mesh spacing, so that the overall solution is second-order accurate. For each partially covered cell, the flux through the boundary is calculated using only values from other cells. This leads to a linear system whose conditioning properties are uniform independent of the smallest partial cell volume, and are essentially the same as that of a problem without irregular boundaries having the the same rectangular mesh spacing. This allows us to use multi-grid iterations with a simple domain-decomposition point relaxation strategy. We have combined this with an adaptive mesh refinement (AMR) procedure, based on the block–structured approach of Berger and Oliger [9]. We show evidence that the algorithm is second-order accurate for various exact solutions, and compare the adaptive and non-adaptive calculations.

Our motivation is to provide a conservative discretization of engineering problems, such as viscous fluid flow or heat conduction, on changing domains. Numerical algorithms for these applications require the solution of elliptic equations on irregular domains. Generally, such equations are derived from a conservation law by using a control volume analysis, along with assumptions about the fluxes of conserved quantities through the surface. This point of view, when applied to a numerical method, has traditionally led to conservative finite-volume formulations. In particular, Cartesian grid embedded boundary methods can have advantages over structured or unstructured grid methods, because of simpler grid generation. The underlying regular grid also allows the use of simpler data structures and numerical methods over a majority of the domain. Accuracy is maintained at the boundaries using a more complicated algorithm, but this extra work is on a one-dimension-smaller set of points.

The approach taken here is motivated by two sets of ideas. The first is that of using conservative volume-of-fluid representations of fronts and irregular boundaries [12, 7, 27, 2]. In this approach, the irregular boundary geometry is represented locally by intersecting the domain Ω with each rectangular cell and approximating the operator using a conservative, finite volume discretization. These methods have been very successful for a variety of problems involving hyperbolic conservation laws in in two and three space dimensions, particularly when used in concert with AMR. The second set of ideas motivating our approach is that of Young, et al. [31], in their treatment of steady transonic potential flow around complex bodies. They used a variational formulation based on rectangular finite elements, where nodal values of the solution could be inside or outside the domain. However, the corresponding volume integrals were only over the regions of each cell, that were inside the physical domain. These two sets of ideas were first combined for solving the incompressible Euler equations using a projection method, in [5]; the algorithm required solving a Poisson equation with Neumann boundary conditions. They included both variational and conservative (MAC-based) forms of the projection operator. We have modified the conservative formulation in [5], to make it formally consistent, and added a means of imposing Dirichlet boundary conditions that maintains the good conditioning of the matrix.

A variety of finite difference discretizations for (1) for the case of irregular boundaries have been presented; a good summary can be found in [20]. The "immersed boundary" method ([28] for example), uses discrete delta functions on domain boundaries, to enforce no-flow boundary conditions for incompressible flows on changing domains. This method is extremely flexible, although it

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

has been shown to lose accuracy in some situations [20]. A related approach called the "immersed interface" method [20], uses a rotated coordinate system and interface jump conditions to find a stencil with genuinely first-order accurate truncation error. This has been successfully applied to a variety of problems with immersed boundaries [21], and has recently been augmented with fast solution methods, such as GMRES [22] and multi-grid [1] algorithms. The practical extension of this method to problems in three dimensions, and those with variable coefficients, is still being pursued.

Another approach was presented by McKenney, et. al. [25], which used a fast multipole and boundary integral method for Laplace's equation, in conjunction with a finite-difference method for Poisson's equation with discontinuous right-hand side [24]. Their method was second-order accurate, even in very complicated regions, and had near-optimal work estimates. Extension of these methods to the variable-coefficient case or to three dimensions is pending. One significant contribution to the approach has been made by Greengard and Lee [16]. They combined a similar integral equation approach with spectral approximation on an adaptive quad-tree data structure. The resulting combination was extremely well-suited for smooth right-hand sides with compact support.

Adaptive solutions of problems like (1) have been dominated by the finite element method ([6, 15, 19], in addition to many others). This approach has the advantage of a rigorous theoretical framework, and a vast number of optimized commercial implementations. Two factors that must be considered, however, are grid generation strategies for complicated domains, and performance of the resulting data structures. Generally, when applying the finite-element method to moving boundary problems, one must take great care that the grid generated is of good quality everywhere (see, for example [30]). In addition, close attention must be given to efficient organization of the resulting data structure.

For the remainder of this paper, we will give the details of the algorithm and its implementation. In Section 2, we describe the discretization in one dimension, and provide some analysis of the accuracy of the method, as well as the conditioning of the resulting linear system. We then describe the non-adaptive algorithm for two dimensions in Section 3. In Section 4, we discuss our multi-grid iterative method; Section 5 explains the modifications needed to include adaptive mesh refinement. In Section 6, we present numerical test cases and demonstrate the method's accuracy. Finally, the last section contains our conclusions and plans for future work.

2 One-Dimensional Algorithm

Consider the Poisson equation with Dirichlet boundary conditions, in one dimension:

$$\varphi_{xx} = \rho \text{ for } x \in [0, \ell], \text{with } \varphi(0) = \Phi^0, \ \varphi(\ell) = \Phi^f. \tag{2}$$

We discretize the interval $[0, \ell]$ with N finite difference cells, by first choosing a volume fraction for the last cell, $\Lambda \in (0, 1]$, and then defining the grid spacing as

$$\Delta x = \frac{\ell}{N - 1 + \Lambda} .$$

Then the size of each finite-difference cell is Δx, except for cell N which abuts $x = \ell$, which is $\Delta x \Lambda$ wide. We denote by $x_{i+\frac{1}{2}}$ the locations of the edges of the cells; thus, $x_{i+\frac{1}{2}} = i\Delta x, \ i = 0, \ldots, N-1$, while $x_{N+\frac{1}{2}} = \ell$. Our discretized solution is denoted as $\phi_i, \ i = 1, \ldots, N$, the values of which are

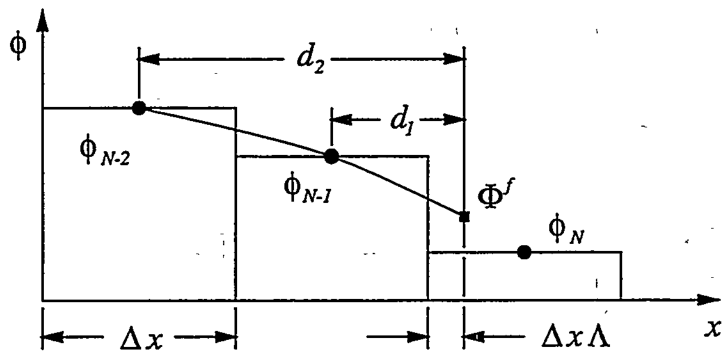


Figure 1: Diagram of the second-order stencil for the gradient at $x = \ell$. A quadratic polynomial is fitted to the two values of ϕ in neighboring cells, and the value at the interface; the value in the last cell is not used in the calculation.

centered at the centers of cells of length Δx:

$$\phi_i \approx \varphi\left(\left(i - \frac{1}{2}\right)\Delta x\right),\ i = 1, \ldots, N$$

Note that ϕ_N is assumed to be centered at the center of the regular, "Cartesian" cell, rather than at the center of the last irregular cell, even if the center of the Cartesian cell is outside the problem domain. In that case, we are assuming that the solution φ can be extended smoothly a small distance beyond the rightmost boundary, and that discrete solution approximates the value of that extended solution. The discretized right-hand side, however, is centered on the irregular cell:

$$\bar{\rho}_i = \rho\left(\frac{x_{i-\frac{1}{2}} + x_{i+\frac{1}{2}}}{2}\right)$$

Our approach is then based on a conservative discretization of (2) on each full or partial cell.

$$(L\phi)_i = \frac{F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}}{x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}} = \bar{\rho}_i \tag{3}$$

On interior edges, we use centered differences to approximate gradients on cell edges:

$$F_{i+\frac{1}{2}} = \frac{\phi_{i+1} - \phi_i}{\Delta x},\ i = 1, \ldots, N-1$$

Note that this same gradient discretization is used on the interior edge of the partial cell, N, abutting $x = \ell$ (Figure 1). This expresses the idea that values of the solution are cell-centered, even if those centers are *outside* the domain. In addition, these gradients are accurate to $O(\Delta x^2)$, and in the interior of the domain, the discretization (3) reduces to the standard three-point finite difference scheme. It is well-known that the cancellation of these errors in the gradient, for constant grid spacing, yields a second-order accurate discretization of (2).

To approximate a gradient at $x = 0$, we fit a quadratic polynomial through the values Φ^0, ϕ_1, and ϕ_2, and evaluate its slope at $x = 0$:

$$F_{\frac{1}{2}} = \frac{1}{3\Delta x}\left(9\phi_1 - \phi_2 - 8\Phi^0\right)$$

This is a standard, second-order finite difference discretization. For the gradient at $x = \ell$ we apply a similar one-sided difference stencil, but using values only in other cells. The second-order difference stencil can be written as

$$F_{N+\frac{1}{2}} = \frac{1}{d_2 - d_1}\left(\left(\Phi^f - \phi_{N-1}\right)\frac{d_2}{d_1} - \left(\Phi^f - \phi_{N-2}\right)\frac{d_1}{d_2}\right) \tag{4}$$

The difference formula is depicted in Figure 1, for the gradient at $x = \ell$. For partial cell N abutting $x = \ell$, the resulting difference formula is

$$\frac{1}{\Delta x\ \Lambda}\left(F_{N+\frac{1}{2}} - \frac{(\phi_N - \phi_{N-1})}{\Delta x}\right) = \bar{\rho}_N\ , \tag{5}$$

where $\bar{\rho}_N$ is value of ρ, at the center of the irregular cell N.

The truncation error of this method can be completely analyzed. Let ϕ^e_i be the value of the exact solution at centers of Cartesian cells: $\phi^e_i = \varphi((i + \frac{1}{2})\Delta x)$. Then the truncation error τ is defined as

$$\tau_i = \bar{\rho}_i - (L\phi^e)_i$$

Note that τ, like ρ and $(L\phi^e)$, is centered on the irregular grid. The error $\xi = \phi - \phi^e$ satisfies the following system of equations:

$$L\xi = \tau,\ \Phi^0 = \Phi^f = 0 \tag{6}$$

We have the following error estimates for τ:

$$\begin{aligned} \tau_1 &= C_1 \Delta x \\ \tau_i &= C_i \Delta x^2, i = 2, \ldots, N-1 \\ \tau_N &= C_N \tfrac{\Delta x}{\Lambda} \end{aligned} \tag{7}$$

In the estimates (7), $C_1, \ldots, C_{N-1}$ are functions of Δx that are uniformly bounded in $\Delta x, i$, provided φ is smooth. C_N is a function of Δx and Λ that is uniformly bounded as both those quantities vary. At first glance this estimate of τ_N may seem singular as $\Lambda \to 0$, but the singularity is matched by that in the denominator of the expression defining the operator in (5). Ultimately, this leads to an estimate of $\xi = O(\Delta x^2)$, uniformly in Λ. We demonstrate this as follows.

Multiplying both sides of (3) by $x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$ and summing, we obtain

$$\begin{aligned} F_{i+\frac{1}{2}} &= F_{N+\frac{1}{2}} + \textstyle\sum_{i<j<N} \Delta x \tau_j + \Lambda \Delta x \tau_N \\ &= F_{N+\frac{1}{2}} + \Delta x^3 \textstyle\sum_{i<j<N} C_j + C_N \Delta x^2 \text{ if } i > 0 \\ &= F_{N+\frac{1}{2}} + \Delta x^3 \textstyle\sum_{1<j<N} C_j + C_N \Delta x^2 + C_1 \Delta x^2 \text{ if } i = 0 \\ &= F_{N+\frac{1}{2}} + D_{i+\frac{1}{2}} \Delta x^2 \end{aligned} \tag{8}$$

where the D_i's are uniformly bounded in Δx, i, and Λ. Given this expression for the fluxes, we can solve for the ξ's:

$$
\begin{aligned}
\xi_1 &= \tfrac{\Delta x}{8}(3F_{\frac{1}{2}} + F_{\frac{3}{2}}) \\
&= \tfrac{\Delta x}{2}F_{N+\frac{1}{2}} + E_1\Delta x^3 \\
\xi_i &= \xi_1 + \textstyle\sum_{1\le j<i} \Delta x F_{j+\frac{1}{2}},\ i = 2,\ldots,N \\
&= (i-\tfrac{1}{2})\Delta x F_{N+\frac{1}{2}} + E_i\Delta x^2
\end{aligned} \tag{9}
$$

Again, the E_i's are uniformly bounded in Δx, i and Λ. Combining (8), (9), and the boundary condition (4), we obtain the following relation for $F_{N+\frac{1}{2}}$:

$$
\begin{aligned}
\Delta x F_{N+\frac{1}{2}} &= -\tfrac{d_2}{d_1}\xi_{N-1} + \tfrac{d_1}{d_2}\xi_{N-2} \\
&= (-\tfrac{d_2}{d_1}(\ell - d_1) + \tfrac{d_1}{d_2}(\ell - d_2))F_{N+\frac{1}{2}} + (\tfrac{d_1}{d_2}E_{N-2} - \tfrac{d_2}{d_1}E_{N-1})\Delta x^2
\end{aligned} \tag{10}
$$

Solving for $F_{N+\frac{1}{2}}$, we finally obtain:

$$
F_{N+\frac{1}{2}} = \frac{d_1^2 E_{N-2} - d_2^2 E_{N-1}}{\ell(d_1+d_2)}\Delta x \tag{11}
$$

Thus, $F_{N+\frac{1}{2}}$ is $O(\Delta x^2)$, uniformly in Λ. From this and the estimates (9) we obtain the result that ξ is $O(\Delta x^2)$ uniformly in Λ.

We can obtain more detailed information regarding the effect of the larger truncation error in the irregular cell. We compute ξ_P, the contribution to ξ from τ_N separately, by solving

$$
\begin{aligned}
(L\xi_P)_i &= 0 \text{ if } i \neq N \\
(L\xi_P)_N &= \tau_N
\end{aligned} \tag{12}
$$

Using the explicit form of the solution given above, we find in that case that

$$
F_{i+\frac{1}{2}} = \frac{\tau_N \Delta x^2 \Lambda}{(\frac{d_2}{d_1} - \frac{d_1}{d_2})\ell} = O(\Delta x^3) \quad i = 0,\ldots,N
$$

so that $\xi_P = O(\Delta x^3)$, uniformly in Λ. Thus we observe that the apparently singular contribution to the truncation error in the irregular cell does not lead to a singularity in the error estimate, due to the multiplication by the length of the cell in (8). In fact, ξ_P, the contribution to the error, is two orders smaller than τ_N, uniformly in Λ.

This fact can be understood from the point of view of potential theory. We can view the error equation (6) as being approximated by a continuous potential theory problem for (2), in which the charge is piecewise constant in cells with values given by the τ_i's. In that case, the contribution to the field ξ from τ_N, in the sense of (12 is given by a dipole located at ℓ of strength

$$
(\text{Total charge in the cell}) \times (\text{Length of the cell}) \approx \tau_N \times (\text{Length of the cell})^2 = O(\Delta x^3)
$$

uniformly in Λ. The reason this is a dipole, rather than a monopole with charge $\Lambda\Delta x\tau_N$, is that the effect of the homogeneous Dirichlet boundary condition at ℓ on the field induced by τ_N can be represented by an image charge with the same total charge, but of opposite sign, located at a distance $\frac{1}{2}\Lambda\Delta x$ to the right of ℓ. We have shown that the conclusion from this potential-theoretic model is rigorously correct in one dimension. For the extension of this algorithm to two space dimensions in the next section, we will use this idea to interpret the various contributions to the

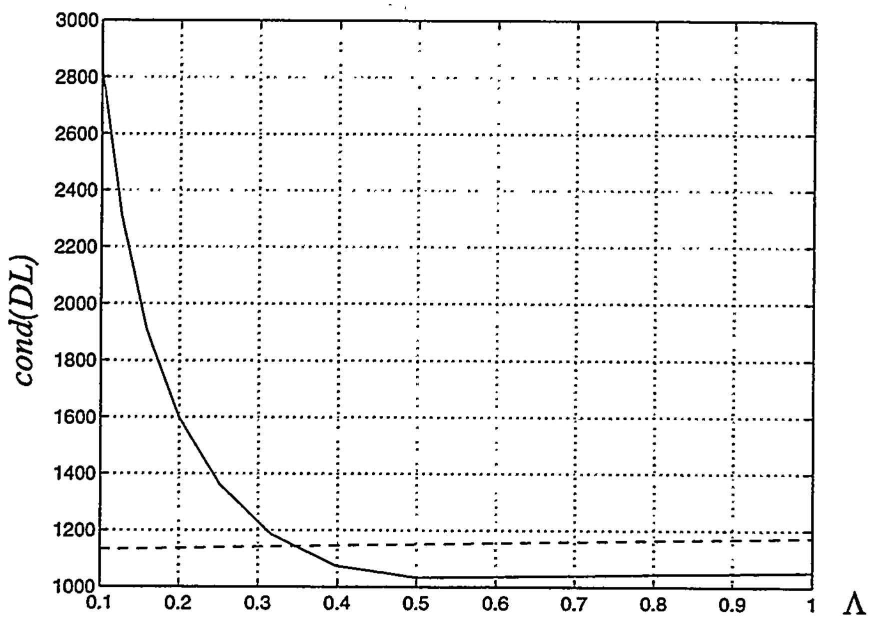


Figure 2: The effect of volume fraction Λ, on the two-norm condition number of the linear system, DL, where D is a diagonal matrix with ones on the diagonal, except for $D_{NN} = \Lambda$, for our system (dashed) and a piecewise-linear Galerkin discretization, with $N-1$ variables, on the same grid (solid).

error observed numerically.

Finally, we wish to emphasize that the use of a stencil for the irregular boundary flux that is well-separated from the boundary is essential. The use of such a stencil leads to the uniform boundedness of the conditioning of the linear algebra as Λ approaches zero. This is definitely not the case with more conventional Galerkin approximations on this kind of irregular grid. In Figure 2 we plot the condition number of the volume-weighted matrix versus Λ, with $N = 50$, along with that of a piecewise-linear Galerkin discretization, with $N-1$ degrees of freedom and the same cell sizes. Note that we have effectively eliminated the problem of poor conditioning in the presence of arbitrarily small volume fractions. The price we pay is that the matrix is not symmetric due to the gradients calculated from quadratic polynomials. Also, the solution may not satisfy a discrete maximum principle, since the centering of ϕ_N will be outside the domain if $\Lambda < \frac{1}{2}$.

3 Two-Dimensional Case

The algorithm in the previous section extends naturally to more space dimensions, because it is based on a finite-volume formulation. The dependent variables ϕ are cell-centered on a uniform rectangular grid: $\phi_{i,j} \approx \varphi((i-\frac{1}{2})\Delta x, (j-\frac{1}{2})\Delta y)$, where φ is a solution to (1). The operator is discretized by integrating (1) over the control volume of each cell; however, to calculate this integral we must first define how the domain boundary is represented. We use a piecewise-linear representation in each cell, which is defined by the intersection of the domain boundary, or "front," with the cell edges (Figure 3). The volume fraction and front normal are then determined from this representation. In each cell (i,j), a simple relationship exists between the inward-facing normal n,

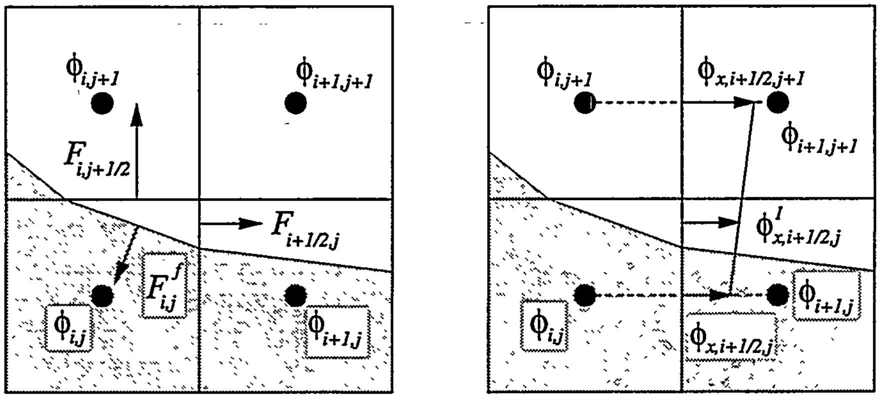


Figure 3: Diagram showing (a) the control volume formulation, which is based on the divergence of appropriately-centered fluxes, and (b) how a properly-centered normal derivative is found by interpolating between two neighboring values.

the area of the front A_f, and the area fractions, called apertures, $a \in [0,1]$, of the cell edges:

$$A^f_{i,j}\mathbf{n}_{i,j} = \Delta x\left(a_{i+\frac{1}{2},j} - a_{i-\frac{1}{2},j}\right)\hat{\imath} + \Delta y\left(a_{i,j+\frac{1}{2}} - a_{i,j-\frac{1}{2}}\right)\hat{\jmath}\ . \tag{13}$$

For full cells, all aperture values are unity, implying that A^f is zero, and $\Lambda = 1$, while in partial cells A^f is non-zero. Note also that this interpretation disallows very narrow (with width less than Δx) "fingers" of the boundary that enter and exit through the same cell edge. A similar algorithm can be used for three dimensions, where the cell faces are defined analogously, and they in turn define the front normal and area. See [27] for a discussion of this kind of geometry discretization, and some of its limitations.

The first step in the derivation is to integrate (1) over each cell's control volume, and take the divergence of surface fluxes. In order to best approximate the surface integral of these fluxes, they are centered at the midpoint of each full or partial edge, as in Figure 3(a). The resulting difference operator can be written as

$$\begin{aligned}(L\phi)_{i,j} = &\ \frac{1}{\Delta x \Delta y \Lambda_{i,j}}\left(F_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j} + F_{i,j+\frac{1}{2}} - F_{i,j-\frac{1}{2}} - F^f_{i,j}\right) \\ = &\ \bar{\rho}_{i,j}\ ,\end{aligned} \tag{14}$$

where we have introduced the volume fraction, $\Lambda_{i,j} \in [0,1]$, and F, the flux through each surface of the control volume in Figure 3(a). For full edges, the flux is found by first calculating a gradient of ϕ normal to the face, using central differencing of neighboring cell values. Note again that ϕ is treated as a cell-centered quantity, even when that center is beyond the domain boundary (as demonstrated in Figure 3). Finally, to calculate F, the gradient is multiplied by β, evaluated at the midpoint of the face, and the area of the face. For the full edge at $(i+\frac{1}{2},j)$, this reduces to

$$F_{i+\frac{1}{2},j} = \Delta y\ \beta_{i+\frac{1}{2},j}\frac{(\phi_{i+1,j} - \phi_{i,j})}{\Delta x}\ . \tag{15}$$

For full cells, it is obvious that this reduces (14) to the standard five-point finite difference stencil

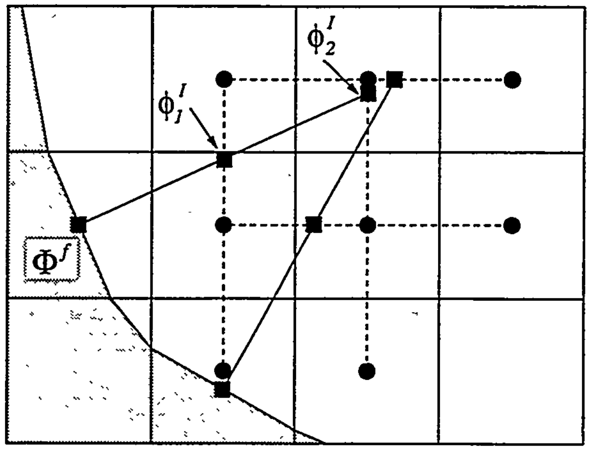


Figure 4: Diagram of the second-order stencil for the gradient normal to the interface, q^f. Two values are found on the neighboring grid lines, using quadratic interpolation. The gradient is then calculated by fitting a parabola to the interpolated values and the value at the interface.

for (1). In partial cells, some of the apertures a are non-unity, implying that A^f is non-zero. In that case, we must do some additional work to construct second-order accurate fluxes.

On a partial edge, the centering of the gradient and β should still be the midpoint of that edge. Therefore, to calculate the normal gradient, we have chosen to linearly interpolate between values at the midpoints of full edges. More specifically, in Figure 3(b), the partial edge $(i+\frac{1}{2}, j)$ has midpoint m, aperture a, and neighboring edge $(i+\frac{1}{2}, j+1)$, the flux is found using the formula:

$$F_{i+\frac{1}{2},j} = a\,\Delta y\,\beta_m \left[\frac{(1+a)}{2}\frac{(\phi_{i+1,j}-\phi_{i,j})}{\Delta x} + \frac{(1-a)}{2}\frac{(\phi_{i+1,j+1}-\phi_{i,j+1})}{\Delta x}\right], \tag{16}$$

where the quantity in brackets is the interpolated gradient, $\phi^I_{x,i+\frac{1}{2},j}$, at m. This reduces to (15) when $a=1$, and provides a second-order accurate approximation of the fluxes through cell edges.

To obtain a consistent discretization of (14), F^f should also be based on quantities centered at the midpoint of the front. Because only the normal component of the gradient contributes to the resulting flux, we have chosen to calculate it using values along a line normal to the interface, and passing through its midpoint (see Figure 4). As in one dimension, we employ a three-point gradient stencil, using values from cells other than the current cell. To do this, we select the first pair of grid lines that intersect with the line normal to the interface, but do not pass through the current cell. We then interpolate between values along each grid line (marked with circles in Figure 4), to the intersection points (marked with boxes in Figure 4). To obtain a second-order accurate gradient, we must use quadratic polynomial interpolation along grid lines, and then apply the gradient formula as in one dimension:

$$q^f = \frac{1}{d_2 - d_1}\left(\frac{d_2}{d_1}\left(\Phi^f - \phi_1^I\right) - \frac{d_1}{d_2}\left(\Phi^f - \phi_2^I\right)\right) \tag{17}$$

Here we have used Φ^f for the value of ϕ on the front; this is given by the Dirichlet boundary condition at the front's midpoint. Interpolation along grid lines determines ϕ_1^I and ϕ_2^I, at the points distance d_1 and d_2 away from the interface. Finally, we can evaluate the interface flux,

$$F^f = \beta^f A^f q^f \ , \tag{18}$$

given β^f, the value of β at the midpoint of the front.

By constructing the gradients in this fashion, we impose one more constraint on the discretization of the domain: the interpolation stencil must not reach into cells with zero volume. For the quadratic gradient stencil, this may imply certain constraints on the discretization of the domain. However, the fact that a zero-volume cell is within two rows of another partial cell would indicate that the local boundary is substantially under-resolved. Such domains are more appropriately treated with adaptive mesh refinement, which is described in Section 6.

Using arguments similar to the one dimensional case, we can compute the local truncation error. We assume that $\varphi = \varphi(x, y)$ is a smooth solution to (1), for the case that ρ, β, and $\partial\Omega$ are smooth. We further assume that φ can be extended smoothly to a slightly larger open set containing Ω. Then for Δx, Δy sufficiently small, we can define the truncation error $\tau_{i,j}$:

$$\begin{aligned} \tau_{i,j} &= \rho_{i,j} - (L\phi^e)_{i,j} \\ \phi^e_{i,j} &= \varphi((i-\tfrac{1}{2})\Delta x, (j-\tfrac{1}{2})\Delta y) \end{aligned} \tag{19}$$

Note that here, as in one dimension, the dependent variable ϕ^e is centered on the rectangular Cartesian grid, while the truncation error is centered at the centroid of the partial cells. In that case, we have the following estimates of the truncation error:

$$\begin{aligned} \tau_{i,j} &= C_{i,j}\Delta x^2 \text{ for interior cells} \\ &= C_{i,j}\frac{\Delta x}{\Lambda} \text{ for partial cells} \end{aligned} \tag{20}$$

Here $\Delta x = \frac{\Delta y}{\alpha}$ for some fixed $\alpha > 0$, and the coefficients $C_{i,j}$ are bounded independent of Δx, Λ, and (i, j). For interior cells, we obtain the standard centered-difference cancellation of error so that the local truncation error is $O(\Delta x^2)$. On the partial cells, that cancellation does not take place, so that the standard Taylor-expansion arguments, plus the fact that the truncation error in the flux calculations is $O(\Delta x^2)$, lead to the estimate given above. Based on a similar potential-theoretic argument as discussed in the one-dimensional case, we expect that the estimates (19) are sufficient to guarantee second order accuracy of the the solution. Specifically, we consider the error equation

$$L(\xi) = \tau \ , \ \xi = \phi - \phi^e \tag{21}$$

If we approximate this as a continuous potential theory problem with a piecewise constant charge $\tau_{i,j}$ on each cell, we expect the contribution of each cell to ξ to be proportional to the total charge on that cell. For an interior cell (i, j), the total charge is $\tau_{i,j} \times \alpha\Delta x^2 = O(\Delta x^4)$. There are $O(\frac{1}{\Delta x^2})$ such cells, leading to a contribution of $O(\Delta x^2)$ to ξ. The total charge in an partial cell (i, j) is $\tau_{i,j} \times \Lambda\alpha\Delta x^2 = O(\Delta x^3)$, uniformly in Λ. However, the contribution to ξ from that charge is a dipole field that is one order smaller in Δx, i.e. $O(\Delta x^4)$. This is because of the influence of the homogeneous Dirichlet boundary condition, whose effect on the field induced by the charge in the partial cell can be represented as an image charge of the same strength but of opposite sign located a distance $O(\Delta x)$ away from the partial cell just outside the boundary of the domain. There are $O(\frac{1}{\Delta x})$ such cells, so that the contribution to the error is $O(\Delta x^3)$, uniformly with respect to the range of values taken on by the $\Lambda_{i,j}$'s. We will verify in detail this behavior in our discussion of the results below.

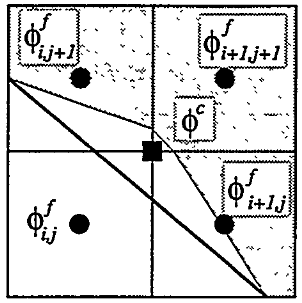


Figure 5: Diagram of the coarsening strategy for the multi-grid method. The coarse grid preserves the apertures and volumes of the fine grid, but uses a coarser, piece-wise linear representation.

4 Multi-Grid Iterations

In order to efficiently find the solution to the linear system derived from (14), we have adopted the use of multi-grid iterations [10]. The multi-grid method is based on combining simple point-relaxation schemes and a hierarchy of coarser grids. After applying point relaxation on the finest grid, a correction term is found by representing the fine-grid residual on the next coarsest grid, and using point-relaxation there. This is applied recursively, down the hierarchy of grids, until the problem is coarsened enough to be solved directly. The correction terms are then interpolated back up the hierarchy, while applying point-relaxation at each level. In all, this is called a multi-grid "V-cycle". Multi-grid methods have the dual benefit of low memory overhead and theoretically-optimal convergence rate. Generally, the method's difficulties are in defining appropriate "coarsened" operators, along with restriction and interpolation functions; poor choices can result in significantly slower convergence.

The grid hierarchy is generated as follows. The coarse grid's spacing in each direction is twice the fine grid's, and a coarse grid's apertures and normals are defined exactly like those of a fine grid: intersection points of the domain boundary with coarse-cell edges define the apertures, which in turn define A^f and $\mathbf{n}$ (Figure 5). However, a coarse cell's volume is defined as the sum of its corresponding four fine-cell volumes; this is required to maintain the flux-difference form of (14). The interface gradient stencil is then determined from this coarse interface representation. This definition of the geometry does have one drawback: it still requires that the interface not cross any *coarse* cell edge more than once. In addition, the limitations of the finite-difference stencil for q^f must be considered. On very coarse grids, these constraints can be violated, and so they determine the end the coarse-grid hierarchy.

The details of the multi-grid iteration scheme are straightforward, once the grid hierarchy is established. The point relaxation scheme that we use resembles a multiplicative Schwarz algorithm from domain decomposition [11]. On the partial cells, we perform one point-Jacobi iteration, while holding the values in the full cells fixed. For iteration m, this is expressed as

$$\phi^m_{i,j} = \phi^{m-1}_{i,j} - \frac{1}{\mu}\left(\bar{\rho}_{i,j} - L\phi^{m-1}_{i,j}\right), \tag{22}$$

where μ is the diagonal entry of $L_{i,j}$. Note that even though μ is $O(\Lambda^{-1}_{i,j})$, it cancels with the operator's denominator. We then perform one sweep of Gauss-Seidel relaxation on the full cells, with either red or black ordering, while holding the partial-cell values fixed. The partial cells, along

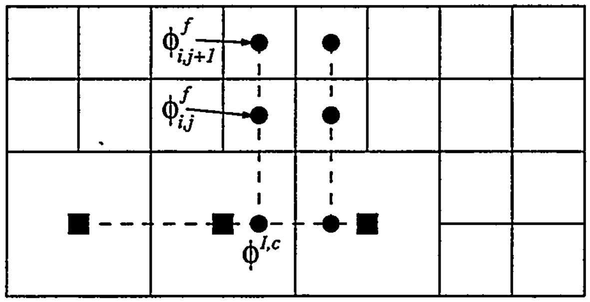


Figure 6: The stencil at the coarse-fine interface is represented. A value is found on a fine-level grid line, from quadratic interpolation on the coarse level. This value, along with two values on the fine grid, is used to calculate a gradient at the coarse-fine interface. Note that the coarse-grid stencil can shift when necessary.

with the full cells used in the stencil for (14), define a region of overlap between the two domains. Although we can provide no convergence analysis for this approach ([11] might provide a good starting point), the convergence rates for the entire multi-grid procedure demonstrate its efficacy.

The residual is restricted to the coarser grid by volume-weighted averaging; the definition of the coarse volume then ensures that a constant $\bar{\rho}$ is coarsened properly. The finite-difference stencil for the gradient on coarser grids is found in the same manner as on the fine grid. On the coarsest grid, we apply the point-relaxation procedure as many times as there are valid points. This is the simplest option, and requires no additional memory or data structures. The coarse correction is then treated as piecewise constant on all cells when interpolating back up the grid hierarchy. This is the least expensive approach, and point-relaxation quickly redistributes coarse corrections locally.

5 Adaptive Mesh Refinement

Oftentimes, the solution provided by a single, uniform discretization of the domain may not be accurate enough. Large gradients in the solution or variation in the domain boundary can require a finer grid spacing than is available with limited computer resources. An adaptive mesh hierarchy enables one to increase grid resolution where necessary; such an approach can greatly reduce the memory required to obtain a given level of accuracy. Our algorithm uses block-grid refinement, based on the work of Berger and Oliger [9]. This permits us to use regular computational data structures, instead of a linked, quad- or oct-tree object (for example, as used in [16]). The algorithm is implemented in a hybrid C++ and Fortran code, where complex organizational tasks are accomplished in C++ data structure library, *BoxLib* [29]. Single-block calculations are performed in Fortran. An excellent discussion of the software issues that have been dealt with in *BoxLib*, can be found in a paper by Crutchfield and Welcome [14].

Our adaptive algorithm is based mostly on work and source code from Cartwright and Martin [23], for adaptive solution of Poisson's equation on rectangular domains. In particular, (14) is used to discretize (1) in every cell, on all levels. Again, the burden of accuracy falls on the discretization of edge gradients. Interfaces between fine and coarse levels have fluxes that are defined by the sum of the more-accurate fine level fluxes (Figure 6). These are calculated using one-sided difference stencils on the fine level, along with a value interpolated from nearby coarse-level cells. This is required to maintain the accuracy of the gradient calculation on the fine grid. As is implied in Figure 6, a quadratic polynomial is fitted to the values in three coarse-grid cells lying beside the fine grid. Then, a second parabola is fitted to the values normal to the boundary, using two fine grid points and the value interpolated from coarse-grid cells. The gradient is then evaluated at the coarse-fine interface. This procedure results in second-order accurate fluxes at the coarse-fine interfaces, which in turn means the discretization of (1) has a first-order truncation error at the coarse-fine interface. However, the coarse-fine interface is a one-dimensional set, so we expect the error in the solution to be second order in the mesh spacing.

A detailed description of this algorithm, and the multi-grid iteration scheme used to solve the resulting linear system, can be found in found in [23] or [26]. The changes required to extend this algorithm to our embedded-boundary method are straightforward. The level relaxation scheme is that described in the previous section, suitably modified to account for the coarse-fine boundary conditions. The averaging and interpolation operators that transmit information between AMR levels are also taken from the that multi-grid algorithm. Also, we use a simplified refinement criterion, which forces refinement at all partial cells, along with suitably chosen buffer so that the values required for the boundary interpolation stencil can be obtained using data from the same grid level. All refined cells are grouped into block grids, for computational efficiency.

6 Numerical Results

We have chosen three simple problems to demonstrate the algorithm, and verify both the single-grid and adaptive algorithms. For the first two problems, the domain is defined by the set

$$\Omega = \{(r,\theta) : r \leq 0.30 + 0.15\cos 6\theta\}\,.$$

The domain is sufficiently complicated to test the algorithm, without compromising the requirements of the finite-difference stencil mentioned in Section 3.

The boundary $\partial\Omega$ is discretized by first representing it as a parameterization, $r(\theta)$; we then choose points in this parameterization, which are no more than $\alpha\Delta x$ apart (the following calculations use $\alpha = 0.3$). By connecting these points, we form a piecewise-linear representation of the interface. The intersections of this representation, with the finest level's grid lines, define the apertures; these then define the area of the front from (13). We also include two modifications to this grid generation algorithm. The first is related to the grid requirements described above and in [27]; if the boundary representation enters and leaves a cell through the same edge, the intersections with that edge are ignored. In effect, this "clips" the boundary where it crosses the same cell edge twice. The second modification, which occurs rarely in practice, is to adjust the boundary to remove cells with volumes less than $10^{-6}\Delta x\Delta y$. This amounts to shifting the boundary points by at most 0.1% of the *grid spacing*, and incurs negligible error relative to that of the numerical algorithm.

The norms used in the following analysis warrant some additional explanation. We must first separate the computational domain into the adaptive levels, Ω^l, where l is the number of cells per unit length, in each grid direction. For example, Ω^{80} designates the set of all valid cells with a grid spacing of $\Delta x = \frac{1}{80}$, that are not covered by a finer level (*i.e.*, Figure 10(a)). The entire computational domain is then defined by $\Omega = \bigcup \Omega^l$. We note again that the domain boundary is represented only on the finest level, which can be divided into two sets of cells: Ω^l_I, which consists of full cells; and Ω^l_P, consisting of partial cells. The set of cells on the finest level is then just $\Omega^l = \Omega^l_I \cup \Omega^l_P$. Unless otherwise noted, cell counts for a level Ω^l refer to the number of uncovered full or partial cells in Ω^l; this excludes both dummy values outside the domain, and values covered by finer levels.

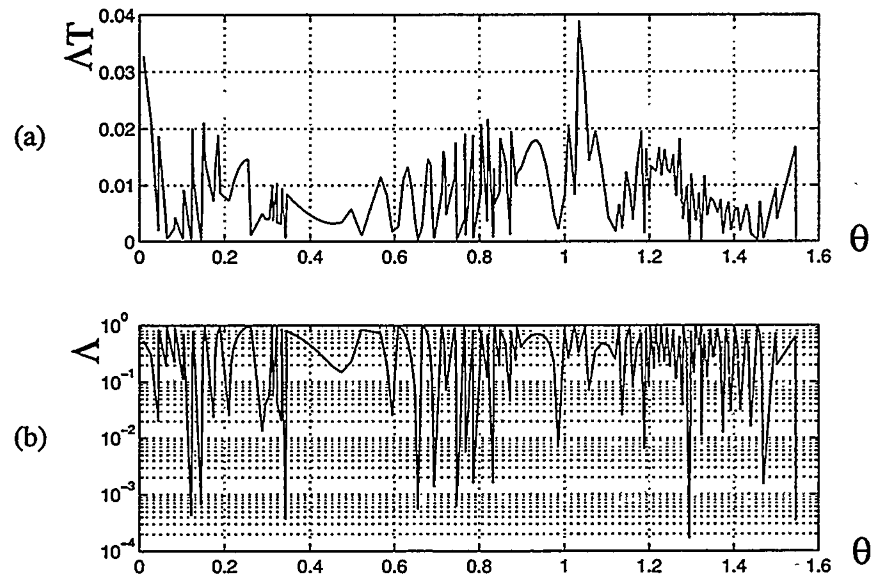


Figure 7: For Problem 1, we plot (a) the magnitude of the volume-weighted truncation error and (b) the partial volume, on Ω_P versus θ. Note that the former is bounded, even for arbitrarily small volumes.

We can now define a volume-weighted norm of a variable e on some set of cells, Ω_k:

$$\|e\|_p^{\Lambda,\Omega_k} = \left(\sum_{(i,j)\in\Omega_k} |e_{i,j}|^p \, \Lambda_{i,j} V^l \Big/ \sum_{(i,j)\in\Omega_k} \Lambda_{i,j} V^l \right)^{\frac{1}{p}}, \tag{23}$$

where $V^l = \Delta x \Delta y$ is the full cell volume on a given level. An unweighted norm, $\|\cdot\|_p^{\Omega_k}$, merely removes $\Lambda_{i,j}$ from equation (23). Any ∞-norm, $\|\cdot\|_\infty^{\Omega_k}$ is just the maximum value over the cells in Ω_k. We can now define the rate of convergence between two norms, e_1 and e_2, with two different grid spacings h_1 and h_2, as

$$r = \log\left(\frac{e_1}{e_2}\right) \Big/ \log\left(\frac{h_1}{h_2}\right) .$$

Thus with $h_1 < h_2$, a rate of $r = 1$ for the two errors e_1 and e_2 indicates a first-order accurate method.

Problem 1. We first set $\beta = 1$, to demonstrate several results for Poisson's equation. The values of the right-hand side are given by the exact Laplacian of the solution,

$$\Delta\varphi = 7r^2 \cos 3\theta,$$

evaluated at the centroid of each finite-difference cell. This is done to best represent the average value in (14). Note that fourth-order derivatives of φ are discontinuous at the origin, and higher-order derivatives are singular. Dirichlet boundary conditions on $\partial\Omega$ are specified by the exact solution:

$$\varphi(r,\theta) = r^4 \cos 3\theta,$$

which has a maximum value of about ± 0.041 at $r = 0.45$ on $\partial\Omega$. The exact solution φ enters into the discretization by taking its value at the midpoint of the front in each cell, and using this as the

N	$\|\Lambda\tau\|_\infty^{\Omega_P}$	r	$\|\Lambda\tau\|_1^{\Omega_P}$	r	N^P	$\|\tau\|_\infty^{\Omega_I}$	r	N^I
40	$1.20 \cdot 10^{-1}$		$2.63 \cdot 10^{-2}$		208	$1.66 \cdot 10^{-3}$		400
80	$7.71 \cdot 10^{-2}$	0.64	$1.45 \cdot 10^{-2}$	0.86	420	$4.15 \cdot 10^{-4}$	2.0	1824
160	$4.20 \cdot 10^{-2}$	0.88	$7.35 \cdot 10^{-3}$	0.98	856	$1.04 \cdot 10^{-4}$	2.0	7712
320	$2.18 \cdot 10^{-2}$	0.95	$3.73 \cdot 10^{-3}$	0.98	1716	$2.59 \cdot 10^{-5}$	2.0	31716
640	$1.11 \cdot 10^{-2}$	0.98	$1.89 \cdot 10^{-3}$	0.98	3416	$6.49 \cdot 10^{-6}$	2.0	128604

Table I: The norms and convergence rates of the partial-volume-weighted truncation error are presented for Problem 1. The values in partial cells are first-order in the grid spacing, while values in the interior are second-order.

N	$\|\xi_P\|_\infty^{\Omega}$	$\|\xi\|_\infty^{\Omega}$	r	$\|\xi_P\|_1^{\Omega}$	r	$\|\xi_I\|_1^{\Omega}$	r	$\|\xi\|_1^{\Omega}$	r
40	$5.89 \cdot 10^{-5}$	$5.85 \cdot 10^{-5}$		$9.33 \cdot 10^{-6}$		$1.38 \cdot 10^{-6}$		$8.34 \cdot 10^{-6}$	
80	$7.35 \cdot 10^{-6}$	$7.36 \cdot 10^{-6}$	3.0	$1.33 \cdot 10^{-6}$	2.8	$3.97 \cdot 10^{-7}$	1.8	$1.02 \cdot 10^{-6}$	3.0
160	$1.17 \cdot 10^{-6}$	$1.17 \cdot 10^{-6}$	2.6	$1.76 \cdot 10^{-7}$	2.9	$1.05 \cdot 10^{-7}$	1.9	$1.07 \cdot 10^{-7}$	3.2
320	$1.67 \cdot 10^{-7}$	$1.68 \cdot 10^{-7}$	2.8	$2.27 \cdot 10^{-8}$	3.0	$2.70 \cdot 10^{-8}$	2.0	$1.80 \cdot 10^{-8}$	2.6
640	$2.26 \cdot 10^{-8}$	$2.27 \cdot 10^{-8}$	2.9	$2.86 \cdot 10^{-9}$	3.0	$6.84 \cdot 10^{-9}$	2.0	$5.02 \cdot 10^{-9}$	1.8

Table II: We present the two components of the solution error in Problem 1, and their corresponding convergence rates. The error due to truncation error in the partial cells converges at a substantially higher rate than that due to the interior truncation error.

value of Φ^f in (17).

We will first analyze the algorithm with uniform grid spacing over the domain. We compute the truncation error τ, defined in (19), for this solution. In Table I, we can still see that $\Lambda\tau$ is $O(\Delta x)$ on Ω_P, consistent with the error estimate (20). In the same table we see that the interior truncation error (τ on Ω_I), which is due to the standard five-point difference scheme, is $O(\Delta x^2)$. Because the domain is star-shaped, we can use θ as an independent variable as we walk along the interface. In Figure 7(a), we plot $\Lambda\tau$ versus the angle θ, in Ω_P only (partial cells); obviously, it is certainly not a smooth function, and contains a substantial high-wavenumber component. This is due to the error being dependent on many non-smooth factors, such as the apertures and distance to interpolation lines. In Figure 7(b), we plot the volume fraction Λ as a function of θ. We see that there is no "blow up" in $\Lambda\tau$ for small Λ.

We can also measure the error in the discrete solution, ξ, defined in equation (21). However, to elucidate the resulting behavior, we have also computed solutions to two subproblems. Specifically, we solve

$$\begin{aligned} L\xi_P &= \tau_P \\ L\xi_I &= \tau_I \end{aligned} \tag{24}$$

where τ_P, τ_I are, respectively, equal to the truncation error on the partial and full cells, and zero elsewhere. In that case, $\xi = \xi_P + \xi_I$, and ξ_P, and ξ_I represent the contributions of the error from the interior and the irregular boundary, respectively. In Table II we see that ξ_P converges at a

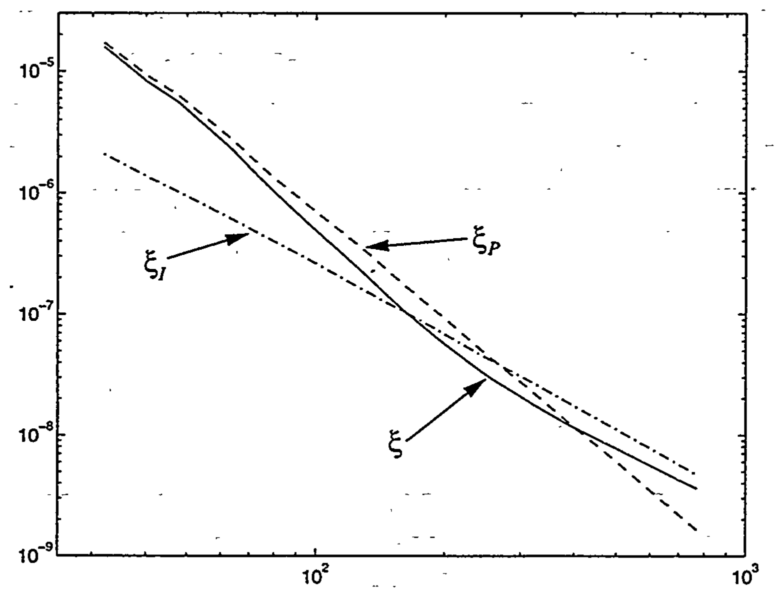


Figure 8: Plot of the one-norm of the error in the discrete solution, ξ, and its two components: ξ_P, from the $O(\Delta x)$ truncation error on Ω_P; and ξ_I, from the $O(\Delta x^2)$ truncation error on Ω_I. Note that ξ_P converges like $O(\Delta x^3)$, while ξ_I converges as $O(\Delta x^2)$.

rate $r \approx 3$ in the ∞-norm. Our explanation of this behavior is the potential-theoretic model for the error on the partial cells described at the end of Section 2. The partial cells induce a dipole distribution on the boundary, due to the homogeneous Dirichlet boundary condition for the error equation. The field induced by this dipole distribution is $O(\Delta x^3)$, uniformly in the range of values taken on by the Λ's. However, ξ_I is is strictly second-order accurate. Table II demonstrates that the overall solution error converges at a rate of $r \approx 3$ for coarser grids, and is then only second-order accurate for finer grids. Figure 8 demonstrates this for the one-norm. Essentially, the error on coarser grids is dominated by the effect of the truncation error in partial cells; for finer grids, the effect of the interior truncation error begins to dominate. Both ξ_P and ξ_I, converge to zero at the stated asymptotic rates; however, their sum does not settle down to its asymptotic rate until $\xi_I >> \xi_P$. This leads to some anomalous behavior in the convergence rate for ξ. For example, the rate of convergence for $||\xi||_1^\Omega$ appears to be less than second order, even though both summands are converging at rates greater than or equal to second order. The reason for this is easily seen in Figure 8. At the grid spacing where $||\xi_P||$ and $||\xi_I||$ are comparable, there is partial cancellation between the two components of the error. At the finer grid spacings, as that cancellation diminishes because of the more rapid convergence of ξ_P, the convergence rate of ξ decreases slightly as it asymptotes to ξ_I.

With this in mind, we can also demonstrate some benefits of adaptive mesh refinement for this problem, even though the right-hand side is evenly distributed over the whole domain. Table III shows three cases using adaptive mesh refinement:

- Case 1: Two levels of refinement, with coarsest level Ω^{80}.
- Case 2: Two levels of refinement, with coarsest level Ω^{160}.

	Level	Ω^{80}	Ω^{160}	Ω^{320}	Ω^{640}	Overall
Case 1	N in Ω^l	896	4984			5880
	$\|\xi\|_\infty^{\Omega^l}$	$1.05 \cdot 10^{-6}$	$1.36 \cdot 10^{-6}$			$1.36 \cdot 10^{-6}$
	$\|\xi\|_1^{\Omega^l}$	$2.87 \cdot 10^{-7}$	$2.24 \cdot 10^{-7}$			$2.51 \cdot 10^{-7}$
Case 2	N in Ω^l		5568	11160		16728
	$\|\xi\|_\infty^{\Omega^l}$		$2.36 \cdot 10^{-7}$	$2.58 \cdot 10^{-7}$		$2.58 \cdot 10^{-7}$
	$\|\xi\|_1^{\Omega^l}$		$9.39 \cdot 10^{-8}$	$4.49 \cdot 10^{-8}$		$7.75 \cdot 10^{-8}$
Case 3	N in Ω^l		4480	9664	21684	35828
	$\|\xi\|_\infty^{\Omega^l}$		$2.79 \cdot 10^{-7}$	$3.20 \cdot 10^{-7}$	$1.50 \cdot 10^{-7}$	$3.20 \cdot 10^{-7}$
	$\|\xi\|_1^{\Omega^l}$		$1.02 \cdot 10^{-7}$	$8.40 \cdot 10^{-8}$	$2.35 \cdot 10^{-8}$	$8.40 \cdot 10^{-8}$
Uniform	$N \in \Omega^l$	2244	8568	33432	132020	$\Uparrow$
	$\|\xi\|_\infty^{\Omega^l}$	$7.36 \cdot 10^{-6}$	$1.17 \cdot 10^{-6}$	$1.68 \cdot 10^{-7}$	$2.27 \cdot 10^{-8}$	$\Leftarrow$
	$\|\xi\|_1^{\Omega^l}$	$1.02 \cdot 10^{-6}$	$1.07 \cdot 10^{-7}$	$1.80 \cdot 10^{-8}$	$5.02 \cdot 10^{-9}$	

Table III: Adaptive mesh refinement errors for the solution in Problem 1. Each case uses different levels of refinement. We also give the results for uniform grid spacing; arrows indicate which sets of values to compare.

- Case 3: Three levels of refinement, with coarsest level Ω^{160}.

In each case, we refine only the boundary region, subject to the constraint that each grid block has at least eight points in each direction (similar to Figure 10(b). By refining around the boundary, we can reduce the impact of the larger truncation error there. For example, the error in the adaptive solution for Case 1 and Case 2, is roughly that of the finest grid, yet both require fewer points than a calculation with uniform grid spacing. However, the effect of the interior truncation error is seen again in Case 3; the algorithm is not able to improve the solution significantly without global refinement, since the truncation error in the interior is evenly distributed for this problem.

Problem 2. Here we include variation in the coefficient β,

$$\beta(r, \theta) = 1 - r^2,$$

which is evaluated at the midpoint of the actual edges in the finite-difference cell of Figure 3. The right-hand side is then given by

$$\nabla \cdot \beta \nabla \varphi = (7r^2 - 15r^4) \cos 3\theta,$$

so that the exact solution is the same as in the first problem. Again, the solution is evaluated at cell-centers when calculating the truncation error, and the right-hand side is evaluated at cell centroids. We can see from Table IV that the non-adaptive cases have results similar to those of Problem 1.

We can also analyze the effectiveness of the multi-grid algorithm for this problem. Each multi-grid iteration applies the point-relaxation scheme four times (*i.e.*, four full sweeps of Gauss-Seidel relaxation), before and after the coarse-grid correction is applied. Figure 9 plots the norm of the residual,

$$\|\Lambda \, (L\phi^m - \bar{\rho})\|_\infty^\Omega,$$

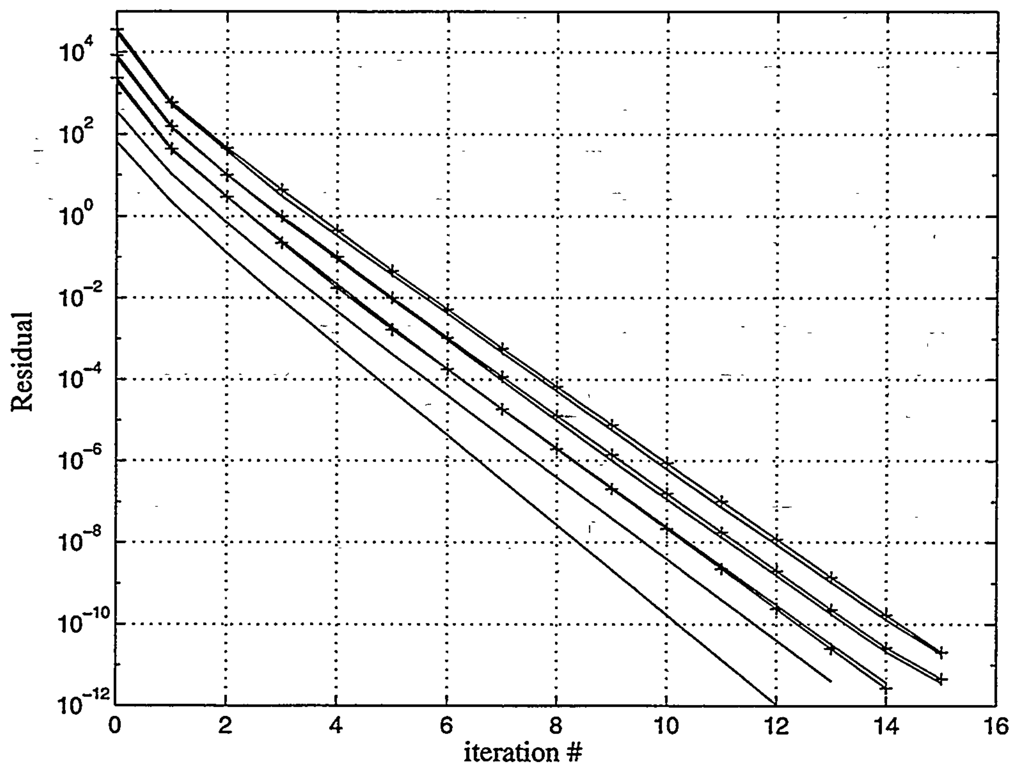


Figure 9: Plot of the ∞-norm of the partial volume-weighted residual, versus multi-grid iteration m, for the single-grid (solid) and adaptive-grid (solid lines with plus signs) solutions of Problem 2.

versus iteration number, m, for all the calculations on a uniform grid. The solution ϕ is initialized to zero, so that the initial residual grows as $O(\Delta x^{-2})$, due to the inhomogeneous boundary conditions. Our multi-grid algorithm reduces the residual by about an order of magnitude per iteration, even as its norm approaches the cutoff of 10^{-11}. There is a slight decrease in performance as the grid spacing is reduced, so that the reduction rates are around 8.5 for the finest grid. The adaptive cases are shown in Figure 9, also. Even with the coarse-fine interface relations, we are able to obtain nearly an order of magnitude reduction in the residual per iteration, despite the unsophisticated interpolation operator.

Problem 3. We also wish to test the algorithm for problems without analytic solutions, to demonstrate that values centered outside the domain do not cause problems. We solve Laplace's equation on $\Omega = \Upsilon_1 \cap \Upsilon_2$, where

$$\Upsilon_1 = \{(r,\theta) : r \geq 0.25 + 0.05\cos 6\theta\}$$

and Υ_2 is the unit square centered at the origin. The Dirichlet boundary conditions are $\phi = 1$ on $\partial\Upsilon_1$, and $\phi = 0$ on $\partial\Upsilon_2$.

A plot of the solution is given in Figure 11; we can see that it extends smoothly outside of $\partial\Upsilon_1$, and overshoots the boundary condition as a result. For equal grid spacing in both directions, the maximum principal for Laplace's equation dictates that there should be no overshoot for values of $\Lambda < \frac{1}{2}$, while values in cells with $\Lambda > \frac{1}{2}$ should be greater than one. Table V shows the number

N	$\|\Lambda\tau\|_\infty^\Omega$	r	$\|\xi\|_\infty^\Omega$	r	$\|\xi\|_1^\Omega$	r
40	$9.60 \cdot 10^{-2}$		$5.86 \cdot 10^{-5}$		$8.16 \cdot 10^{-6}$	
80	$6.17 \cdot 10^{-2}$	0.64	$7.30 \cdot 10^{-6}$	3.0	$9.57 \cdot 10^{-7}$	3.1
160	$3.36 \cdot 10^{-2}$	0.88	$1.17 \cdot 10^{-6}$	2.6	$9.58 \cdot 10^{-8}$	3.3
320	$1.75 \cdot 10^{-2}$	0.94	$1.68 \cdot 10^{-7}$	2.8	$2.00 \cdot 10^{-8}$	2.3
640	$8.84 \cdot 10^{-3}$	0.99	$2.28 \cdot 10^{-8}$	2.9	$6.00 \cdot 10^{-9}$	1.7

Table IV: We list errors and convergence rates for Problem 2. The results are very similar to those obtained in Problem 1.

N	$\|\xi\|_\infty^{\Omega_I}$	r	$\|\xi\|_1^{\Omega_I}$	r	k/N^P
40	$1.27 \cdot 10^{-2}$		$2.82 \cdot 10^{-3}$		8/104
80	$4.32 \cdot 10^{-3}$	1.6	$6.28 \cdot 10^{-4}$	2.2	0/208
160	$1.27 \cdot 10^{-3}$	1.8	$1.50 \cdot 10^{-4}$	2.1	4/408
320	$2.45 \cdot 10^{-4}$	2.4	$3.00 \cdot 10^{-5}$	2.3	0/820

Table V: We present results for Problem 3. The error between successive levels is approximately second-order in the grid spacing. In addition, the last column indicates that relatively few cells violate a discrete maximum principal.

of cells, k, in Ω_P that violate this criterion; we see that it is variable, but small, with respect to the total number of points in Ω_P. For the finest grid, $N = 640$ and $k = 0$. We assume that this violation occurs when a partial volume, $\Lambda \approx \frac{1}{2}$, is in a region with significantly under-resolved gradients.

To evaluate the convergence of the algorithm for the single grid case, we compare two solutions, with a factor of two difference in grid spacing. To do this comparison, we must interpolate the solution on the fine level, to the coarse grid cell centers. This is done with bilinear interpolation, denoted with B, between the four fine-level solution values closest to the coarse grid cell center, as in Figure 5. We can then define the error as the difference between the two results:

$$\xi^l = B_l^{l+1}\phi^{l+1} - \phi^l \text{ on } \Omega_I^l \, .$$

We do this only for interior cells on the coarse grid, because the values needed on the fine level, in order to interpolate to the center of a partial cell on the coarse-grid, are not necessarily available. Table V contains the convergence rates for this error, which are roughly second-order in both norms.

To demonstrate the AMR algorithm, we compare the solution on each level, to the solution with the finest uniform grid spacing, i. e., we replace ϕ^{l+1} above with ϕ^{640}, where Ω^{640} is the finest grid level in this case. Although the resulting errors are not appropriate for calculating convergence rates, they are accurate up to the error on the finest grid. In Table VI, we see that the AMR algorithm is able to improve the accuracy of the method substantially, by merely refining around the boundaries. In Case 1, with two levels, we are able to obtain results with the accuracy of the finest grid, with only 35% of the points; in Case 2, with three levels, this holds true with 17%.

	Level	Ω^{80}	Ω^{160}	Ω^{320}	Overall
Case 1	N in Ω_I^l	4288	3096		7384
	$\|\xi\|_\infty^{\Omega^l}$	$6.64 \cdot 10^{-4}$	$1.49 \cdot 10^{-3}$		$1.49 \cdot 10^{-3}$
	$\|\xi\|_1^{\Omega^l}$	$1.48 \cdot 10^{-4}$	$3.71 \cdot 10^{-4}$		$1.82 \cdot 10^{-4}$
Case 2	N in Ω_I^l	3792	3664	6148	13604
	$\|\xi\|_\infty^{\Omega^l}$	$1.58 \cdot 10^{-4}$	$1.43 \cdot 10^{-4}$	$2.46 \cdot 10^{-4}$	$2.46 \cdot 10^{-4}$
	$\|\xi\|_1^{\Omega^l}$	$2.83 \cdot 10^{-5}$	$6.42 \cdot 10^{-5}$	$7.45 \cdot 10^{-5}$	$3.83 \cdot 10^{-5}$
Uniform	N in Ω_I^l	5016	20248	81476	$\Uparrow$
	$\|\xi\|_\infty^{\Omega^l}$	$5.49 \cdot 10^{-3}$	$1.49 \cdot 10^{-3}$	$2.45 \cdot 10^{-4}$	$\Leftarrow$
	$\|\xi\|_1^{\Omega^l}$	$8.06 \cdot 10^{-4}$	$1.80 \cdot 10^{-4}$	$3.00 \cdot 10^{-5}$	

Table VI: Adaptive mesh refinement errors for the solution in Problem 3. Errors are found by comparing the solution to values interpolated from the finest result. The last row contains results for uniform grid spacing; arrows indicate which set of values to compare.

Figure 10(a) demonstrates the valid regions on which norms are computed, for for Case 1; Figure 10(b) plots the solution and block grid structure.

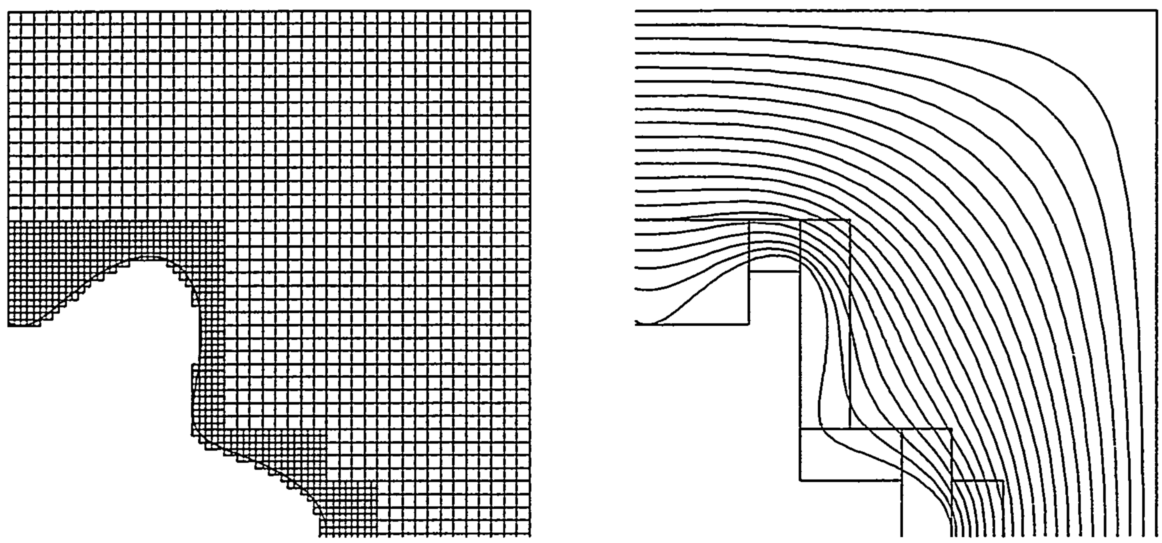
7 Conclusions

The algorithm described in this paper satisfies a number of desirable criteria. The finite-volume formulation uses second-order accurate gradients for calculating surface fluxes. These gradients are calculated from cell-centered quantities, even when those centers are outside the domain. The truncation error for the resulting discretization for Eq. (1) is first-order in the mesh spacing only along the domain boundary, and second-order in the interior. In our three test problems, the solution is found to be second-order accurate, on domains with significant curvature and variation. We also observe numerically that the error induced by the truncation error at the boundary, converges to zero like $O(\Delta x^3)$. We have given a rigorous proof that this is the case in one dimension, and in two dimensions have given a potential-theoretic model for the error induced by the discretization on partial cells that accounts for this behavior.

Our analysis in one dimension demonstrated that our discretization is well-conditioned, even in the presence of arbitrarily small or thin cells. In addition, the multi-grid algorithm uses only a simple point-relaxation scheme, with volume-weighted restriction and piecewise constant prolongation operators, and we obtain nearly the same multi-grid reduction rates for the residual, regardless of grid size or quality. This suggests that we retain a well-conditioned system in more than one dimension.

We have demonstrated that our method is amenable to the introduction of adaptive mesh refinement to improve the accuracy locally. We refine the cells containing portions of the domain boundary; this simultaneously refines the geometry description, while reducing the effect of the larger truncation error. The multi-grid framework attains reduction rates for the adaptive grid hierarchy that are no worse than those with uniform refinement.

There are a number of research directions suggested by this algorithm. More accurate geom-



(a) (b)

Figure 10: We present two plots of the first quadrant of Problem 3, Case 1. (a) represents the valid regions of each level, and (b) gives grid block boundaries, and contours of the solution.

etry specifications, such as the sub-cell information used in [2], along with the ability to handle thin "fingers," will be required to effectively extend the algorithm to arbitrary three-dimensional domains, and problems with internal interfaces. Similarly, it is necessary to extend the adaptive version of the algorithm to allow embedded boundaries to cross refinement levels, a capability that has proven useful in the hyperbolic case [7, 13, 27]. Applications such as viscous fluid flow and heat transfer will require a conservative, time-dependent discretization; we are currently developing such an algorithm for the heat equation. Adding a front-tracking algorithm [7] [12] to the current method will allow us to treat moving boundaries, such as those found in the Stefan problem and in studying fluid-structure interactions. Finally, there are a number of applications of the ideas here in a more general setting. The formal truncation error analysis introduced here for centering the fluxes (16) could be used in the conservative algorithm for elliptic equations with Neumann boundary conditions discussed in [5], with the possibility of improving the rate of convergence in that algorithm from the $O(\Delta x^{1.75})$ reported there. A similar approach could also be applied to the algorithms discussed in [12, 27] for hyperbolic problems. However, in the hyperbolic case, it is not obvious what the tradeoffs are between improved accuracy and robustness in the presence of shocks and other discontinuities. The present work also suggests an approach to deriving conservative boundary conditions for overset grid algorithm of the sort discussed in [17]. In this case, the irregular boundary is not the boundary of the domain, but the boundary given by an overlapping grid.

References

[1] L. Adams, "A Multigrid Algorithm for Immersed Interface Problems," in *Proceedings, 7th Copper Mountain Multigrid Conference, Copper Mountain, CO, April 2-7, 1995.* See

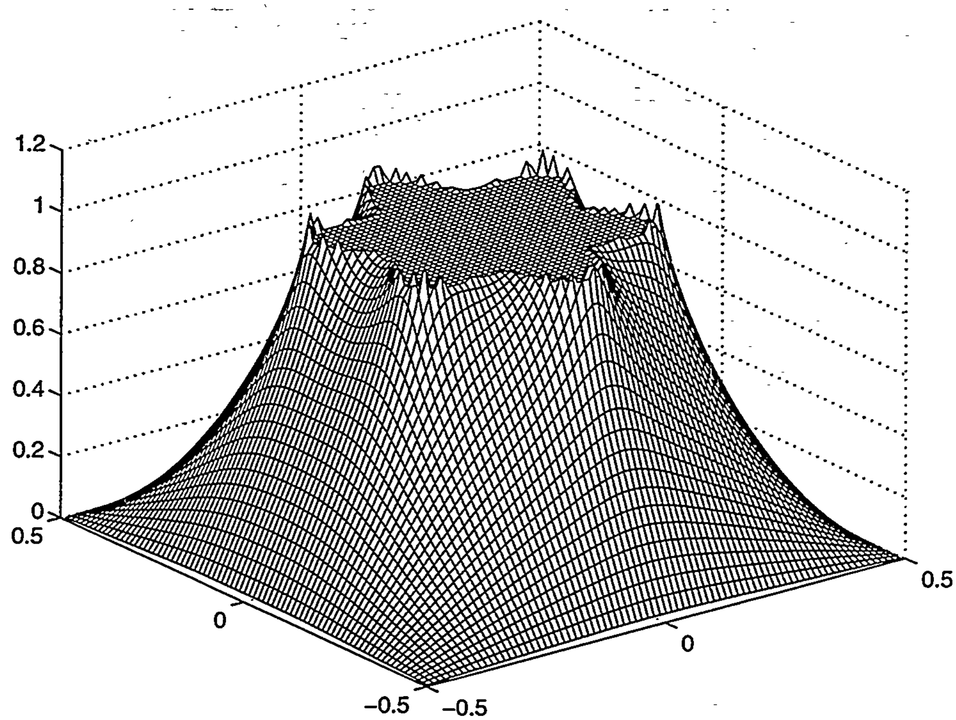


Figure 11: Surface plot of the solution to Problem 3, for $N = 80$.

http://na.cs.yale.edu/mgnet/www/mgnet-ccmm95.html.

[2] M. J. Aftosmis, J. E. Melton, and M. J. Berger, "Adaptation and Surface Modelling for Cartesian Mesh Methods," AIAA-95-1725-CP, *AIAA 12th Computational Fluid Dynamics Conference, San Diego, CA, June 19-22, 1995.*

[3] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, "A Conservative Adaptive Projection Method for the Variable Density Incompressible Navier-Stokes Equations," submitted to *J. Sci. Comput.*

[4] A. S. Almgren, J. B. Bell, P. Colella, and L. H. Howell, "An Adaptive Projection Method for the Incompressible Euler Equations," in *Proceedings, AIAA 11th Computational Fluid Dynamics Conference, Orlando, FL, July 6-9, 1993.*

[5] A. Almgren, J. B. Bell, P. Colella, and T. Marthaler, "A Cartesian Grid Projection Method for the Incompressible Euler Equations in Complex Geometries," *SIAM J. Sci. Comput.*, to appear.

[6] "Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations," edited by I. Babuska, *et al.*, (Springer-Verlag, New York, 1995).

[7] J. B. Bell, P. Colella, and M. Welcome, "Conservative Front-Tracking for Inviscid Compressible Flow," in *Proceedings, AIAA 10th Computational Fluid Dynamics Conference, Honolulu, Hawaii, June 24-27, 1991*, p. 814.

[8] M. J. Berger and P. Colella, "Local Adaptive Mesh Refinement for Shock Hydrodynamics," *J. Comput. Phys.* **82**, 64-84 (1989).

[9] M. J. Berger and J. Oliger, "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations," *J. Comput. Phys.* **53**, 484-512 (1984).

[10] W. Briggs, *A Multigrid Tutorial*, (SIAM, Philadelphia, PA, 1987).

[11] T. F. Chan and T. P. Mathew, "Domain Decomposition Algorithms," *Acta Numerica*, p. 41-143 (1994).

[12] I. Chern and P. Colella, "A Convervative Front Tracking Method for Hyperbolic Conservation Laws," UCRL JC-97200, Lawrence Livermore National Laboratory, July 1987 (unpublished).

[13] P. Colella and L. F. Henderson, "The von Neumann Paradox for the Diffraction of Weak Shock Waves," *J. Fluid Mech.* **213**, 71-94 (1990).

[14] W. Y. Crutchfield and M. L. Welcome, "Object-Oriented Implementations of Adaptive Mesh Refinement Algorithms," *Scientific Programming* **2**, 145-156 (1993).

[15] L. Demkowicz, J. T. Oden, W. Rachowicz, and O. Hardy, "Toward a Universal h-p Adaptive finite element strategy," Comp. Meth. in Appl. Mech. and Eng., **77**, p.79 (1989).

[16] L. Greengard and J. Lee, "A Direct Poisson Solver of Arbitrary Order Accuracy," *J. Comput. Phys.* **125**, 415-424 (1996).

[17] G. Chesshire and W. D. Henshaw, "Composite Overlapping Meshes for the Solution of Partial Differential Equations," *J. Comput. Phys.* **90**, 1-64 (1990).

[18] L. H. Howell and J. B. Bell, "An Adaptive-Mesh Projection Method for Viscous Incompressible Flow," *SIAM J. Sci. Comp.*, to appear.

[19] C. Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method,* (Cambridge University Press, New York, 1987).

[20] R. J. LeVeque and Z. Li, "The Immersed Interface Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources," *SIAM J. of Numer. Anal.* **31**, 1019-1044 (1994).

[21] Z. Li, *The Immersed Interface Method – A Numerical Approach for Partial Differential Equations with Interfaces,* Ph. D. thesis, University of Washington, Seattle, WA, 1994.

[22] Z. Li, "A Fast Iterative Method for Elliptic Interface Problems," *SIAM J. of Numer. Anal.*, to appear.

[23] D. F. Martin and K. L. Cartwright, "Solving Poisson's Equation Using Adaptive Mesh Refinement," UCB/ERL M96/66, Electronics Research Laboratory Memorandum, University of California, Berkeley, October 1996 (unpublished). Code and documentation available at `http://barkley.me.berkeley.edu/~martin/public_html/AMRPoisson.html` .

[24] A. Mayo, "The Rapid Evaluation of Volume Integrals of Potential Theory on General Regions," *J. Comput. Phys.* **100**, 236-245 (1992).

[25] A. McKenney, L. Greengard, and A. Mayo, "A Fast Poisson Solver for Complex Geometries," *J. Comput. Phys.* **118**, 348-355 (1995).

[26] M. Minion, "A Projection Method for Locally Refined Grids," *J. Comput. Phys.* **127**, 158-178 (1996).

[27] R. B. Pember, J. B. Bell, P. Colella, W. Y. Crutchfield, and M. Welcome, "An Adaptive Cartesian Grid Method for Unsteady Compressible Flow in Irregular Regions," *J. Comput. Phys.* **120**, 278-304 (1995).

[28] C. S. Peskin and B. F. Printz, "Improved Volume Conservation in the Computation of Flows with Immersed Elastic Boundaries," *J. Comput. Phys.* **105**, 33-46 (1993).

[29] C. Rendleman, "BoxLib Tutorial," source code and documentation available at `http://www.nersc.gov/research/CCSE/software/boxlib.tutorial/boxlib.html` .

[30] A. Schmidt, "Computation of Three Dimensional Dendrites with Finite Elements," *J. Comput. Phys.* **125**, 293-312 (1996).

[31] D. P. Young, R. G. Melvin, M. B. Bieterman, F. T. Johnson, S. S. Samant, and J. E. Bussoletti, "An Locally Refined Rectangular Grid Finite Element Method: Application to Computational Fluid Dynamics and Computational Physics," *J. Comput. Phys.* **62**, 1-66 (1991).