A Cartesian Grid Embedded Boundary Method for the Heat Equation and Poisson's Equation in Three Dimensions [1,2,3]

Peter Schwartz [*,1,3], Michael Barad [2], Phillip Colella [1,3], Terry Ligocki [3]

Applied Numerical Algorithms Group, Lawrence Berkeley National Laboratory, Berkeley, California 94720 [3]

Department of Civil and Environmental Engineering, University of California, Davis, California 95616 [2]

Abstract

We present an algorithm for solving Poisson's equation and the heat equation on irregular domains in three dimensions. Our work uses the Cartesian grid embedded boundary algorithm for 2D problems of Johansen and Colella (1998, J. Comput. Phys. **147**(2):60–85) and extends work of McCorquodale, Colella and Johansen (2001, J. Comput. Phys. **173**(2):60–85). Our method is based on a finite-volume discretization of the operator, on the control volumes formed by intersecting the Cartesian grid cells with the domain, combined with a second-order accurate discretization of the fluxes. The resulting method provides uniformly second-order accurate solutions and gradients and is amenable to geometric multigrid solvers.

Key words: Poisson Equation, Heat Equation, Multigrid Methods
PACS: 02.60.Lj, 02.70.Bf, 41.05.+e, 41.20.Cv

[*] Corresponding author
Email address: `poschwartz@lbl.gov` (Peter Schwartz).
[1] Supported by the DARPA BioComp program.
[2] Supported by the Computational Science Graduate Fellowship program of the Department of Energy, under grant number DE-FG02-97ER25308.
[3] Supported at the Lawrence Berkeley National Laboratory by the U.S Department of Energy: Director, Office of Science, Office of Advanced Scientific Computing, Mathematical, Information, and Computing Sciences Division under Contract DE-AC03-76SF00098.

1 Introduction

In this paper, we present Cartesian grid embedded boundary methods for solving Poisson's equation and the heat equation on irregular domains in three dimensions. In this approach, the irregular domain is discretized as a collection of control volumes formed by the intersection of the domain with cubic Cartesian grid cells. The primary unknowns are defined at the centers of the Cartesian cells, while the Laplacian is approximated by a finite volume discretization on each of the regular or irregular control volumes. This approach was successfully used in [JC98] and [MCJ01] to solve these problems in two dimensions. The present work is a generalization of that work to 3D. The principal new features of the algorithm include the use of bilinear interpolation to approximate the fluxes on irregular faces and a more general treatment of the moving boundaries. In addition, we introduce an alternate lower-order stencil for computing the flux on a Dirichlet boundary at locations where the boundary geometry is underresolved on the grid. The latter feature permits the use of multigrid coarsening of the grid to much coarser levels than in [JC98]. The resulting method is second-order accurate in space for Poisson's equation, and second-order accurate in space and time for the heat equation.

2 Embedded Boundary Discretization of the Laplacian in 3D

The underlying discretization of space is given by rectangular control volumes on a Cartesian grid: $\Upsilon_{\boldsymbol{i}} = [\boldsymbol{i}h, (\boldsymbol{i} + \boldsymbol{u})h]$, $\boldsymbol{i} \in \mathbb{Z}^D$, where D is the dimensionality of the problem, h is the mesh spacing, and $\boldsymbol{u}$ is the vector whose entries are all ones. In the case of a fixed, irregular domain Ω, the geometry is represented by the intersection of Ω with the Cartesian grid. 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 $\partial V_{\boldsymbol{i}}$ with the coordinate planes $\{\boldsymbol{x} : x_d = (i_d \pm \frac{1}{2})h\}$. Here $\boldsymbol{e}_d$ is the unit vector in the d direction. We also define $A^B_{\boldsymbol{i}}$ to be the intersection of the boundary of the irregular domain with the Cartesian control volume: $A^B_{\boldsymbol{i}} = \partial\Omega \cap \Upsilon_{\boldsymbol{i}}$. We will assume here that there is a one-to-one correspondence between the control volumes and faces and the corresponding geometric entities on the underlying Cartesian grid. The description can be generalized to allow for boundaries whose width is less than the mesh spacing or boundaries with sharp trailing edges.

In order to construct finite difference methods, we will need only a small number of real-valued quantities that are derived from these geometric objects.

- Areas and volumes are expressed in dimensionless terms: volume fractions $\kappa_{\boldsymbol{i}} = |V_{\boldsymbol{i}}|h^{-D}$, face apertures $\alpha_{\boldsymbol{i} \pm \frac{1}{2}\boldsymbol{e}_d} = |A_{\boldsymbol{i} \pm \frac{1}{2}\boldsymbol{e}_d}|h^{-(D-1)}$ and boundary apertures $\alpha^B_{\boldsymbol{i}} =$

$|A^B_{\boldsymbol{i}}|h^{-(D-1)}$. We assume that we can compute estimates of the dimensionless quantities that are accurate to $O(h^2)$.

- The locations of centroids, and the average outward normal to the boundary are given exactly by:

$$\begin{aligned}
\text{Face centroid: } \boldsymbol{x}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d} &= \frac{1}{|A_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}|}\int_{A_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}} \boldsymbol{x}dA \\
\text{Boundary face centroid: } \boldsymbol{x}^B_{\boldsymbol{i}} &= \frac{1}{|A^B_{\boldsymbol{i}}|}\int_{A^B_{\boldsymbol{i}}} \boldsymbol{x}dA \\
\text{Outward normal: } \boldsymbol{n}^B_{\boldsymbol{i}} &= \frac{1}{|A^B_{\boldsymbol{i}}|}\int_{A^B_{\boldsymbol{i}}} \boldsymbol{n}^B dA
\end{aligned}$$

  where $\boldsymbol{n}^B$ is the outward normal to $\partial\Omega$, defined for each point on $\partial\Omega$. Again, we assume that we can compute estimates of these quantities that are accurate to $O(h^2)$.

Using just these quantities, we can define conservative discretizations for the divergence operator. Let $\vec{F} = (F^1 \dots F^D)$ be a function of $\boldsymbol{x}$. Then

$$\begin{aligned}
\nabla \cdot \vec{F} &\approx \frac{1}{|V_{\boldsymbol{i}}|}\int_{V_i} \nabla \cdot \vec{F} dV = \frac{1}{|V_{\boldsymbol{i}}|}\int_{\partial V_i} \vec{F} \cdot \boldsymbol{n} dA \\
&\approx \frac{1}{\kappa_{\boldsymbol{i}} h}\Big[\Big(\sum_{\pm=+,-}\sum_{d=1}^{D} \pm\alpha_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d} F^d(\boldsymbol{x}_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_d})\Big) + \alpha^B_{\boldsymbol{i}}\boldsymbol{n}^B_{\boldsymbol{i}} \cdot \vec{F}(\boldsymbol{x}^B_{\boldsymbol{i}})\Big]
\end{aligned} \tag{1}$$

where (1) is obtained by replacing the integrals of the normal components of the vector field $\vec{F}$ with the values at the face centroids.

We first consider Poisson's equation on an irregular domain Ω.

$$\begin{aligned}
\Delta\psi &= \rho \text{ on } \Omega \\
\frac{\partial\psi}{\partial n} &= g_N \text{ on } \partial\Omega \\
&\text{or} \\
\psi &= g_D \text{ on } \partial\Omega
\end{aligned} \tag{2}$$

We define a discrete variable ϕ, $\phi_{\boldsymbol{i}} \approx \psi(\boldsymbol{i}h)$. Using the discretization of the divergence

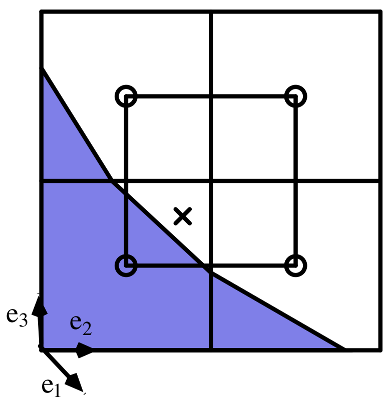

Fig. 1. 3d bilinear flux

defined in (1), we can define a discretization of Poisson's equation as follows.

$$(\Delta^h \phi)_{\boldsymbol{i}} = \rho_{\boldsymbol{i}} \tag{3}$$

$$(\Delta^h \phi)_{\boldsymbol{i}} = \frac{1}{\kappa_{\boldsymbol{i}} h} \Big[\Big(\sum_{\pm = +,-} \sum_{d=1}^{D} \pm \alpha_{\boldsymbol{i} \pm \frac{1}{2} \boldsymbol{e}_d} F^d_{\boldsymbol{i} \pm \frac{1}{2} \boldsymbol{e}_d} \Big) + \alpha^B_{\boldsymbol{i}} F^B \Big] \tag{4}$$

where $\rho_{\boldsymbol{i}} = \rho(\boldsymbol{i}h)$, and the fluxes F^d and F^B are linear combinations of $\phi_{\boldsymbol{i}}$ and the boundary values. In practice, we avoid problems arising from arbitrarily small values of $\kappa_{\boldsymbol{i}}$ in the denominator by solving:

$$\kappa_{\boldsymbol{i}} (\Delta^h \phi)_{\boldsymbol{i}} = \kappa_{\boldsymbol{i}} \rho_{\boldsymbol{i}} \tag{5}$$

The fluxes are given by bilinear interpolation of centered differences. Explicitly, bilinear interpolation of fluxes can be written as an iteration of linear interpolation of fluxes in the two directions that are not normal to the face. For example, given the face with outward normal $\boldsymbol{e}_1$, with centroid $\boldsymbol{x}$, define the linearly interpolated flux in the d $(d \neq 1)$ direction by

$$\begin{aligned} F^d_{\boldsymbol{i} + \frac{1}{2} \boldsymbol{e}_1} &= \eta \frac{(\phi_{\boldsymbol{i} + \boldsymbol{e}_1} - \phi_{\boldsymbol{i}})}{h} + (1 - \eta) \frac{(\phi_{\boldsymbol{i} + \boldsymbol{e}_1 \pm \boldsymbol{e}_d} - \phi_{\boldsymbol{i} \pm \boldsymbol{e}_d})}{h} \\ \eta &= 1 - \frac{|\boldsymbol{x} \cdot \boldsymbol{e}_d|}{h} \\ \pm &= \begin{cases} + & \boldsymbol{x} \cdot \boldsymbol{e}_d > 0 \\ - & \boldsymbol{x} \cdot \boldsymbol{e}_d \leq 0. \end{cases} \end{aligned} \tag{6}$$

The bilinear interpolation of the flux for the face with normal $\boldsymbol{e}_1$ can be written

$$\begin{aligned} F_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_1} &= \eta F^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_1} + (1-\eta) F^d_{\boldsymbol{i}\pm\boldsymbol{e}_{d'}+\frac{1}{2}\boldsymbol{e}_1} \\ \eta &= 1 - \frac{|\boldsymbol{x}\cdot\boldsymbol{e}_{d'}|}{h} \\ \pm &= \begin{cases} + & \boldsymbol{x}\cdot\boldsymbol{e}_{d'} > 0 \\ - & \boldsymbol{x}\cdot\boldsymbol{e}_{d'} \le 0 \end{cases} \end{aligned} \tag{7}$$

where $d' \neq d$, $d' \neq 1$. See figure 1. We note that the particular choice of bilinear interpolation for computing the fluxes is a nontrivial one for obtaining a stable method. In particular, we also tried using simple linear interpolation based on three of the faces in figure 1, omitting the face offset along the diagonal. We found that such a method is unstable for some configurations of adjacent small control volumes, in the sense that point Jacobi fails to converge for any value of the relaxation parameter. No such instability was observed for the bilinear scheme. For that reason, we have chosen to reduce order to a piecewise constant interpolant of the fluxes if all four faces required for a bilinear interpolant are unavailable.

2.1 Boundary Conditions

For Neumann boundary conditions the flux on the boundary is specified. For Dirichlet boundary conditions further calculations are necessary. Our primary method, for use when the geometry is well resolved, is a generalization of the methods described in [JC98]. Figure 2 shows how this generalizes to 3D.

$$\frac{\partial\phi}{\partial n} \approx \frac{1}{d_2 - d_1}\left(\frac{d_2}{d_1}(\phi^B - \phi^I_1) - \frac{d_1}{d_2}(\phi^B - \phi^I_2)\right) \tag{8}$$

Here we have used ϕ^B for the value of ϕ on the boundary, which is given by the Dirichlet boundary condition. Interpolation from cell centered values determines ϕ^I_1 and ϕ^I_2 at distance d_1 and d_2 away from the boundary, respectively.

If all 18 cells are available in figure 2, we make an order $O(h^2)$ estimate $\nabla\phi$ as follows. Depending on the orientation of the normal, two planes are chosen, P_1 and P_2. Using biquadratic interpolation, two values ϕ^I_1 and ϕ^I_2 are calculated, each requiring 9 values.

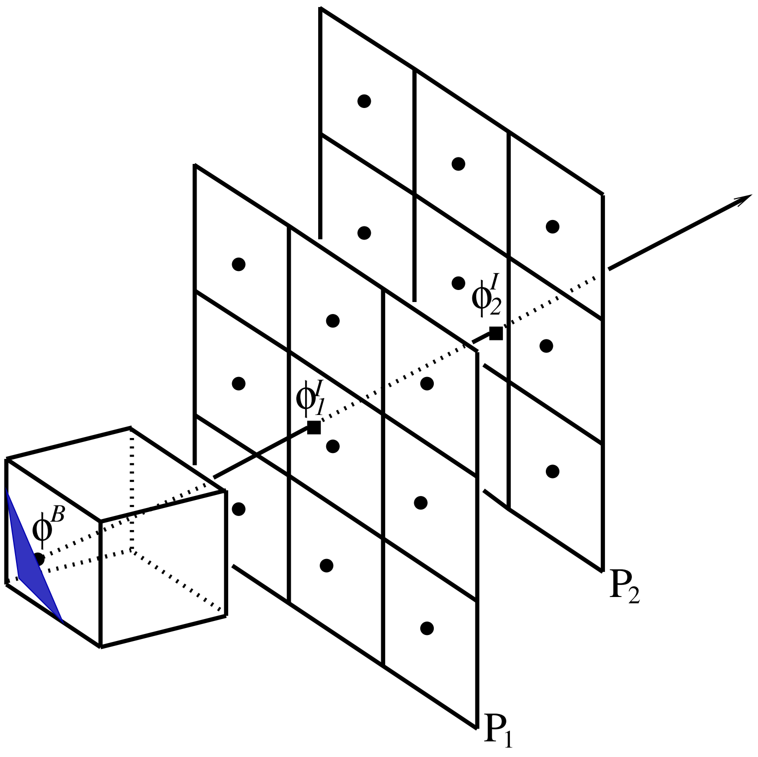

Fig. 2. Diagram of the second order stencil for the gradient normal to the interface.

The gradient is then calculated by fitting a quadratic to the interpolated values and the value at the interface. We chose the planes P_1, P_2 to be perpendicular to $\boldsymbol{e}_d$, where d is given by

$$\{d : n_d^B \geq n_\ell^B,\ \ell = 1, 2, 3\}. \tag{9}$$

In cases where the requisite eighteen cells are not available, we employ a lower order stencil to estimate the flux to $O(h)$. In 3D this lower order stencil contains at most eight points including the centroid of the embedded boundary. These eight points are chosen as follows. We associate each one of the eight possible configurations of plus or minus signs of the components of the normal vector with one of the octants in the the coordinate system with origin at the centroid. The stencil consists of the cell-centers of the seven nearest cells in this octant, excluding the cell-center of the control volume containing the boundary centroid itself.

From these points we create an overdetermined linear system to estimate $\nabla\phi$ as follows:

$$\mathcal{A} \cdot \nabla\phi = \delta\phi \tag{10}$$

where

$$\begin{aligned}\mathcal{A} &= (\delta\boldsymbol{x}_1, \delta\boldsymbol{x}_2, ..., \delta\boldsymbol{x}_7)^T \\ \delta\phi &= (\delta\phi_1, \delta\phi_2, ..., \delta\phi_7)^T \\ \delta\boldsymbol{x}_m &= \boldsymbol{x}_m - \boldsymbol{x}^B_{\boldsymbol{i}} \\ \delta\phi_m &= \phi_m - \phi_B.\end{aligned}$$

We determine $\nabla\phi$ using least squares by solving the normal equations:

$$\mathcal{A}^T\mathcal{A} \cdot \nabla\phi = \mathcal{A}^T\delta\phi.$$

Provided that $\mathcal{A}$ contains three linearly independent rows, $\mathcal{A}^T\mathcal{A}$ is invertible. This is always the case provided the set $\{\boldsymbol{x}_m\}$ contains all points of the form $(\boldsymbol{i} + \boldsymbol{e}^d)h$, plus at least one other point. If that is not the case, we set $\nabla\phi \equiv 0$. These methods lead to a condition number for Δ^h that is bounded independent of κ, and comparable to that of the uniform grid algorithm.

2.2 Truncation and Solution Error

We define the truncation error in the usual fashion: $\tau_{\boldsymbol{i}} = \rho_{\boldsymbol{i}} - \Delta^h\phi^{exact}_{\boldsymbol{i}}$, where $\phi^{exact}_{\boldsymbol{i}} = \psi(\boldsymbol{i}h)$. We then have the following asymptotic error estimates for the truncation error. At regular cells

$$\tau_{\boldsymbol{i}} = O(h^2). \tag{11}$$

If $\boldsymbol{i}$ is an irregular cell, and the flux on the boundary is second order accurate, as in (8), then

$$\tau_{\boldsymbol{i}} = O\Big(\frac{h}{\kappa_{\boldsymbol{i}}}\Big). \tag{12}$$

If we use the flux computation given by (10), the flux on the boundary is first order accurate, and we have

$$\tau_{\boldsymbol{i}} = O\Big(\frac{1}{\kappa_{\boldsymbol{i}}}\Big). \tag{13}$$

We refer to methods that satisfy truncation error estimates of the form (12) on the irregular control volumes as being *formally consistent.* We also define the solution error $\epsilon_{\boldsymbol{i}} = \phi_{\boldsymbol{i}} - \phi_{\boldsymbol{i}}^{exact}$.

There is one apparent problem with this truncation error estimate: it is only first order accurate at the boundary. Nonetheless, we observe robust second-order convergence of the solution in max norm. These two facts can be reconciled using a modified equation analysis [JC98]. Both methods of calculating the gradient for Dirichlet boundary conditions, (8) and (10), lead to a second order solution error. This is because, for Dirichlet boundary conditions, solution error is two orders of accuracy more than the truncation error on the boundary. On the other hand, in the next section we show that it is necessary to use (8) to obtain second-order accurate values for $\nabla\phi$ at the boundary.

3 Discretizing the Heat Equation

We now consider the heat equation on a moving domain. $\psi : R^3 \times [0,\infty] \rightarrow R$ is the unknown and $f : R^3 \times [0,\infty] \rightarrow R$ is the source term. On a time dependent domain, we solve

$$\begin{aligned}
\frac{\partial \psi}{\partial t} &= \Delta\psi + f \\
\frac{\partial \psi}{\partial n} &= g_N(\boldsymbol{x}, t),\ \boldsymbol{x} \in \partial\Omega(t) \\
&\text{or} \\
\psi &= g_D(\boldsymbol{x}, t),\ \boldsymbol{x} \in \partial\Omega(t).
\end{aligned} \tag{14}$$

3.1 Time Discretization for a Fixed Domain

First, we consider the case of a fixed domain, i.e. $\Omega(t) = \Omega$ independent of time. We define discrete variables, $\phi_{\boldsymbol{i}}(t), f_{\boldsymbol{i}}(t)$

$$\phi_{\boldsymbol{i}}(t) \approx \psi(\boldsymbol{i}h, t), \quad f_{\boldsymbol{i}}(t) \approx f(\boldsymbol{i}h, t). \tag{15}$$

This leads to a semi-discrete system of ODEs

$$\frac{d\phi_{\boldsymbol{i}}}{dt} = \Delta^h \phi_{\boldsymbol{i}} + f_{\boldsymbol{i}} \tag{16}$$

We discretize this system in time using the L_0-stable method [TGA96], which was also described in [MCJ01], as outlined below.

Denote by I the identity operator and by Δ^h_I and Δ^h_H the discrete Laplacian with inhomogeneous and homogeneous boundary conditions, respectively. We split the timestep Δt such that

$$\begin{aligned} \mu_1 + \mu_2 + \mu_3 &= \Delta t \\ \mu_1 + \mu_2 + \mu_4 &= \Delta t/2. \end{aligned}$$

The update at step n uses the boundary values at the old and new times and also at an intermediate time t_{int}:

$$\begin{aligned} \phi^{n+1} = &(I - \mu_1 \Delta^h_I(t_{new}))^{-1}(I - \mu_2 \Delta^h_I(t_{int}))^{-1} \cdot \\ &[(I + \mu_3 \Delta^h_I(t_{old}))\phi^n + (I + \mu_4 \Delta^h_H) f(t_{avg})\Delta t] \end{aligned} \tag{17}$$

where

$$\begin{aligned} t_{old} &= n\Delta t \\ t_{new} &= (n+1)\Delta t = t_{old} + \mu_1 + \mu_2 + \mu_3 \\ t_{int} &= t_{new} - \mu_1 = t_{old} + \mu_2 + \mu_3 \\ t_{avg} &= (t_{old} + t_{new})/2 = t_{old} + \mu_1 + \mu_2 + \mu_4. \end{aligned}$$

For a second-order L_0-stable method, following [TGA96], we pick $a > 1/2$ and

$$\begin{aligned} \mu_1 &= \frac{a - \sqrt{a^2 - 4a + 2}}{2}\Delta t \\ \mu_2 &= \frac{a + \sqrt{a^2 - 4a + 2}}{2}\Delta t \\ \mu_3 &= (1 - a)\Delta t \\ \mu_4 &= (\frac{1}{2} - a)\Delta t. \end{aligned}$$

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

It was shown in [MCJ01] for Dirichlet boundary conditions that Crank-Nicolson time discretization exhibited oscillatory behavior and furthermore was unstable to some types of forcing at a moving boundary. In [Joh97] this behavior was attributed to the combination of the neutral stability of Crank-Nicolson at high wave numbers and the presence of eigenvalues of Δ^h with non-trivial imaginary parts, corresponding to eigenvalues with oscillatory behavior near the boundary.

3.2 Time Discretization for a Moving Domain

In the moving case, the domain Ω is now a function of time, $\Omega = \Omega(t)$, and the various geometric quantities can also be computed in a time-dependent way: $\kappa_{\boldsymbol{i}}(t)$, $\alpha_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_d}(t)$, and so on.

The timestep is assumed to satisfy a CFL condition with respect to the normal velocity $v = \frac{d\boldsymbol{x}}{dt} \cdot \mathbf{n}$:

$$|v|\frac{\Delta t}{h} < 1.$$

In the present approach we solve the moving-boundary problem by defining an equivalent fixed-boundary problem for each timestep. Specifically, we solve at each time step the discretization of the following fixed-boundary problem:

$$\frac{\partial \psi^{fixed}}{\partial t}(\boldsymbol{x}, t) = D\Delta\psi^{fixed}(\boldsymbol{x}, t) + f(\boldsymbol{x}, t) \tag{18}$$

where

$$\begin{aligned} \psi^{fixed} &= \psi^{fixed}(\boldsymbol{x}, t),\ \boldsymbol{x} \in \Omega(t_{\text{new}}),\ t_{\text{old}} \le t \le t_{\text{new}} \\ \frac{\partial \psi^{fixed}}{\partial n} &= g_N^{extrap}(\boldsymbol{x}, t),\ \boldsymbol{x} \in \partial\Omega(t_{\text{new}}) \\ &\text{or} \\ \psi^{fixed} &= g_D^{extrap}(\boldsymbol{x}, t),\ \boldsymbol{x} \in \partial\Omega(t_{\text{new}}). \end{aligned} \tag{19}$$

The boundary conditions $g_{N,D}^{extrap}$ on the fixed boundary are computed by extrapolating values from the moving boundary to the points on the fixed boundary $\partial\Omega(t_{\text{new}})$ at

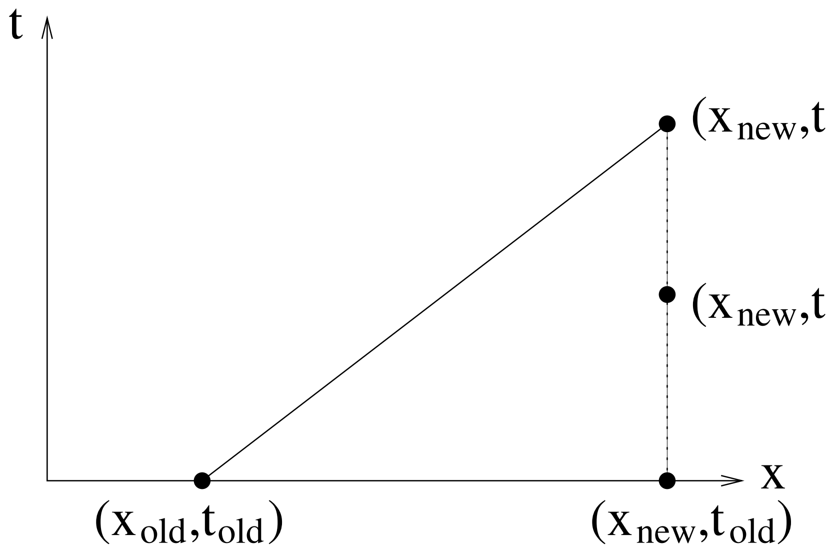

Fig. 3. Boundary conditions for the equivalent problem are extrapolated for $(x_{\text{new}}, t_{\text{int}})$ and $(x_{\text{new}}, t_{\text{old}})$.

times t_{old} and t_{int}, as in (20). Figure 3 shows a one dimensional schematic with time and figure 4 shows the boundary at two different times.

The steps required in setting up the fixed-boundary problem (19) are:

(1) Extend the domain of ϕ^n to $\Omega(t_{\text{new}})$, and define the newly uncovered values by extrapolation.
(2) Compute boundary values for ϕ^{n+1} at $(\boldsymbol{x}^B_{\boldsymbol{i}}(t_{\text{new}}), t_{\text{old}})$ and $(\boldsymbol{x}^B_{\boldsymbol{i}}(t_{\text{new}}), t_{\text{int}})$.

In Step 1, to estimate the value of ϕ^n at the center $\boldsymbol{x}_{\boldsymbol{i}}(t_{\text{new}})$ of a newly uncovered cell in $\Omega(t_{\text{new}}) - \Omega(t_{\text{old}})$, we use a quadratic extrapolant from three other cells in $\Omega(t_{\text{old}})$, such that the centers of these cells form a line with $\boldsymbol{x}_{\boldsymbol{i}}(t_{\text{new}})$. We choose whichever line passing through the centers of the new cell and one of its immediate neighbors has a direction closest to that of the normal $\boldsymbol{n}^B_{\boldsymbol{i}}(t_{\text{new}})$. See figure 4.

In Step 2, for each $V_{\boldsymbol{i}} \in \Omega(t_{\text{new}}) - \Omega(t_{\text{old}})$, we choose the $\boldsymbol{j}$ in the intersection of the $3 \times 3 \times 3$ block of cells with center $\boldsymbol{i}$ and $\Omega(t_{\text{old}})$ that has the largest boundary area. For our extrapolation, we define

$$\delta(\boldsymbol{i}, \boldsymbol{j}) = (\boldsymbol{x}^B_{\boldsymbol{i}}(t_{\text{new}}), t_{\text{old}}) - (\boldsymbol{x}^B_{\boldsymbol{j}}(t_{\text{old}}), t_{\text{old}})$$

We also need approximations to $\nabla\psi$ and $\nabla\nabla\psi$, which we estimate as follows.

Let $\vec{G}_{\boldsymbol{i}} = \nabla\psi(\boldsymbol{x}_{\boldsymbol{i}}(t_{\text{new}}), t_{\text{old}}) + O(h)$. Each component of $G^d_{\boldsymbol{i}}$ is computed separately by differentiating the quadratic interpolant through three points chosen from $\phi_{\boldsymbol{i}}$, $\phi_{\boldsymbol{i}\pm\boldsymbol{e}_d}$,

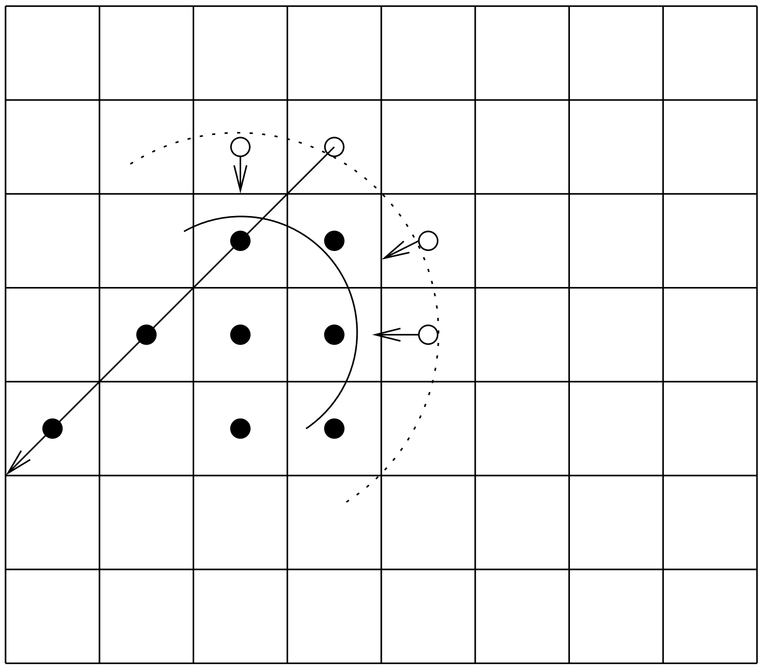

Fig. 4. Centers of cells in $\Omega(t_{\text{old}})$ are shown with solid circles, and centers of cells in $\Omega(t_{\text{new}})$ - $\Omega(t_{\text{old}})$ are shown with unfilled circles. To estimate the value of ϕ^n at one of these latter points, we extrapolate quadratically from values at the centers of three other cells in $\Omega(t_{\text{old}})$ forming a line with the new cell center. We pick whichever such line is closest in direction to the normal at time t_{new}.

and $\phi_{\boldsymbol{i}\pm 2\boldsymbol{e}_d}$, where the sign of $\pm$ is chosen so that all points are in $\Omega(t_{\text{new}})$ and therefore where ϕ^n has been computed.

Explicitly, for the d component,

$$G^d_{\boldsymbol{i}} = \pm\frac{1}{h}(-\frac{3}{2}\phi^n_{\boldsymbol{i}} + 2\phi^n_{\boldsymbol{i}\pm\boldsymbol{e}_d} - \frac{1}{2}\phi^n_{\boldsymbol{i}\pm 2\boldsymbol{e}_d})$$

or

$$G^d_{\boldsymbol{i}} = \frac{1}{h}(\phi^n_{\boldsymbol{i}+\boldsymbol{e}_d} - \phi^n_{\boldsymbol{i}-\boldsymbol{e}_d}).$$

depending on whether $\boldsymbol{i}$ is on the end or in the middle of 3 cells.

We estimate second derivatives as follows. Let $GG_{\boldsymbol{i}} \approx \nabla\nabla\psi(\boldsymbol{x}_{\boldsymbol{i}}(t_{\text{new}}), t_{\text{old}})$. The order of the error in this approximation will vary between one and two, depending on the local geometry. For derivatives of the form $\phi_{d,d}$, we differentiate the quadratic extrapolant through $\phi_{\boldsymbol{i}}$, $\phi_{\boldsymbol{i}\pm\boldsymbol{e}_d}$, and $\phi_{\boldsymbol{i}\pm 2\boldsymbol{e}_d}$ according to the stencil

$$G^d G^d_{\boldsymbol{i}} = \frac{1}{h^2}(\phi^n_{\boldsymbol{i}+\boldsymbol{e}_d} - 2\phi^n_{\boldsymbol{i}} + \phi^n_{\boldsymbol{i}-\boldsymbol{e}_d}).$$

This term is second order if $\boldsymbol{i}$ is in the center of the three point stencil and first order otherwise. For mixed derivatives, ($\phi_{d,d'}$ with $d \neq d'$), we average one, two, three, or four estimates of the derivative each of which is second order accurate at points of the form $\boldsymbol{i} \pm \boldsymbol{e}_d \pm \boldsymbol{e}_{d'}$. For example, assuming that $\boldsymbol{i} + \boldsymbol{e}_d, \boldsymbol{i} + \boldsymbol{e}_{d'}$, and $\boldsymbol{i} + \boldsymbol{e}_d + \boldsymbol{e}_{d'}$ are all in $\Omega(t_{\text{new}})$, we include the following in our average:

$$G^d G^{d'}_{\boldsymbol{i}} = \frac{1}{h^2}(\phi^n_{\boldsymbol{i}+\boldsymbol{e}_d+\boldsymbol{e}_{d'}} - \phi^n_{\boldsymbol{i}+\boldsymbol{e}_d} - \phi^n_{\boldsymbol{i}+\boldsymbol{e}_{d'}} + \phi^n_{\boldsymbol{i}+\boldsymbol{e}_d}).$$

Given these estimates of $\nabla\psi$ and the matrix $\nabla\nabla\psi$ at cell centers, we extrapolate the boundary conditions using:

$$\begin{aligned} g_N^{extrap} &= g_N(\boldsymbol{x}^B_{\boldsymbol{i}}(t_{\text{old}}), t_{\text{old}}) + (\boldsymbol{n}^B_{\boldsymbol{j}} - \boldsymbol{n}^B_{\boldsymbol{i}}) \cdot \vec{G}_{\boldsymbol{i}} + \boldsymbol{n}^B_{\boldsymbol{i}} \cdot [\vec{G}\vec{G}_{\boldsymbol{i}}\delta(\boldsymbol{i},\boldsymbol{j})] \\ &\text{or} \\ g_D^{extrap} &= g_D(\boldsymbol{x}^B_{\boldsymbol{i}}(t_{\text{old}}), t_{\text{old}}) + \vec{G}_{\boldsymbol{i}} \cdot \delta(\boldsymbol{i},\boldsymbol{j}) \end{aligned} \tag{20}$$

where as remarked above, $\boldsymbol{j}$ is chosen from among the neighbors of $\boldsymbol{i}$ as the neighbor with the largest boundary area.

Finally, we linearly interpolate the boundary conditions at $(\boldsymbol{x}^B_{\boldsymbol{i}}(t_{\text{new}}), t_{\text{int}})$ from the boundary conditions at $(\boldsymbol{x}^B_{\boldsymbol{i}}(t_{\text{new}}), t_{\text{old}})$ and $(\boldsymbol{x}^B_{\boldsymbol{i}}(t_{\text{new}}), t_{\text{new}})$. The former is calculated and the latter is given as part of the data of the problem.

4 Numerical Results

For our test problems, we compute the max norm of the solution error and truncation error. For a discrete variable, ξ, the max norm is given by:

$$||\xi||_\infty = \max_{\boldsymbol{i}} |\xi_{\boldsymbol{i}}|,$$

and the p-norm in 3D is given by:

$$||\xi||_p = (\sum_{\boldsymbol{i}} |(\xi_{\boldsymbol{i}})^p h^D \kappa_{\boldsymbol{i}}|)^{\frac{1}{p}}.$$

4.1 Truncation Error for the Laplacian and Solution Error for Poisson's equation

We estimate the truncation error of the discretized Laplacian using the test function

$$f(x,y,z) = sin(x)sin(2y)sin(3z). \tag{21}$$

Here Ω is a sphere of radius $r = 0.392$ centered at the origin. Our discretized Laplacian has inhomogeneous boundary conditions of either Neumann or Dirichlet type. Figures 5 and 6 show that the discretization of the operator has the accuracy anticipated by (11), (12), and (13). If we set $\psi = f$, ψ satisfies

$$\begin{aligned} \Delta\psi &= -14f \text{ on } \Omega \\ \frac{\partial\psi}{\partial n} &= \frac{\partial f}{\partial n} \text{ on } \partial\Omega \\ &\text{or} \\ \psi &= f \text{ on } \partial\Omega. \end{aligned} \tag{22}$$

Figures 13 and 14 show the solution error for Poisson's equation with Dirichlet boundary conditions (using the higher order stencil) and Neumann boundary conditions. Figures 7 and 8 show one of the differences between the higher and lower order Dirichlet stencils. If the lower order stencil is used, $\nabla\phi - \nabla\phi^{exact} = O(h)$, as expected. However, for the higher order stencil $\nabla\phi - \nabla\phi^{exact} = O(h^2)$. In other words, our numerical experiments imply that for these cases, at least, our estimated solution has the asymptotic form $\phi = \phi^{exact} + Ch^2$, where C is a smooth function for the higher order stencil, but not for the lower order stencil. In figure 9 and 10 we display the smoothness solution error.

In figures 11 and 12, we show the effectiveness of the multigrid V cycles and W cycles. We considered mesh sizes of 16, 32, 64, 128, and 256 (only the last three are shown). Using the test problem (22), we reduced the residual eleven orders of magnitude by using V cycles and by using W cycles. Figure 11 shows that for each doubling of the mesh size, four or five more V cycles are required to achieve the same reduction of the residual. By contrast, increasing the mesh size does not increase the number of

W cycles required. On the other hand, W cycles require more work than V cycles and at these resolutions solving this problem with V cycles takes less computational time than solving it with W cycles.

4.2 Solution Error for the Heat Equation

Our test problems for the heat equation (14) have as their solution

$$\psi(x,y,z,t) = \frac{4\ \exp\ (-\frac{x^2+y^2+z^2}{5(t+1)})}{5\pi(t+1)} \tag{23}$$

satisfying

$$\psi_t = \Delta\psi + f, \tag{24}$$

where

$$f(x,y,z,t) = \frac{4(x^2+y^2+z^2+5(t+1))}{125\pi(t+1)^3}\ \exp\ (-\frac{x^2+y^2+z^2}{5(t+1)}).$$

In the case where the boundary is not moving we solve the Neumann problem on a computational domain based on an neutrophil. Beginning with slice-data generated by confocal microscopy, we constructed an implicit function with the following property: the surface of the neutrophil is implicitly defined by the set of points at which the implicit function takes the value zero. From this implicit definition, we construct the volume fractions, apertures, normals, and centroids necessary for Cartesian grid embedded boundary methods.

We display the solution error in figure 18, the solution in figure 19, and a convergence study in figure 15.

In the case of a moving boundary, we numerically solve the Neumann and Dirichlet problems on a spherical domain, with boundary conditions of Neumann or Dirichlet type computed using the exact solution.

The radius of the sphere changes with time, increasing at a prescribed speed. We advance the solution in time from $t = 0$ to $t = 0.1875$ using a mesh spacing h and corresponding timestep Δt such that $\Delta t/h = 0.5$. The number of timesteps equals $\frac{0.1875}{\Delta t}$ and $h = 2^{-n}$ where $n = 4, ..., 8$. The solution error is shown in figures 16 and 17.

5 Conclusion

We have described a Cartesian grid embedded boundary algorithm for solving Poisson's equation and the heat equation on irregular domains in three dimensions. The resulting method provides uniformly second-order accurate solutions and gradients and is amenable to geometric multigrid solvers. In the future, we plan on extending this approach in a variety of directions: to generalize this approach to free boundary value problems, in which the discretizations used here are used on both sides of the boundary, combined with jump relations at the boundary, to define a finite-volume discretization; and to combine this approach for elliptic and parabolic problems with the approach described in [CGKM] to solve a variety of problems in low-Mach number flows with fixed and free boundaries. To carry this program out, it will be useful to extend the approach described here to the case of adaptively refined meshes, for which the only outstanding issue is the question of discretizing the operator when the embedded boundary crosses a boundary between refinement levels. Finally, it would be interesting to attempt to extend the method described here to higher order, e.g. fourth-order accuracy. A starting point for this could be the finite-volume formulation of fourth-order Mehrstellen methods in [BC].

Acknowledgments

We would like to thank Adam Arkin, Matt Onsum, and David Adalsteinsson for help with generating the neutrophil geometry. Dan Graves, Dan Martin, Ted Sternberg, and Brian Van Straalen contributed helpful ideas as well as information about the Chombo software library.

References

[BC] M. Barad and P. Colella. A fourth order accurate adaptive mesh refinement method for Poisson's equation. Submitted for publication, 2004.

[CGKM] P. Colella, D.T. Graves, B.J. Keen, and D. Modiano. A Cartesian grid embedded boundary method for hyperbolic conservation laws. Submitted for publication, 2004.

[JC98] H. Johansen and P. Colella. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *J. Comput. Phys.*, 147(2):60–85, December 1998.

[Joh97] Hans Svend Johansen. *Cartesian Grid Embedded Boundary Finite Difference Methods for Elliptic and Parabolic Partial Differential Equations on Irregular Domains.* PhD thesis, University of California, Berkeley, 1997.

[MCJ01] P. McCorquodale, P. Colella, and H. Johansen. A Cartesian grid embedded boundary method for the heat equation on irregular domains. *J. Comput. Phys.*, 173:620–635, November 2001.

[TGA96] E.H. Twizell, A.B. Gumel, and M.A. Arigu. Second-order, l_0-stable methods for the heat equation with time-dependent boundary conditions. *Advances in Computational Mathmatics*, 6:333–352, 1996.

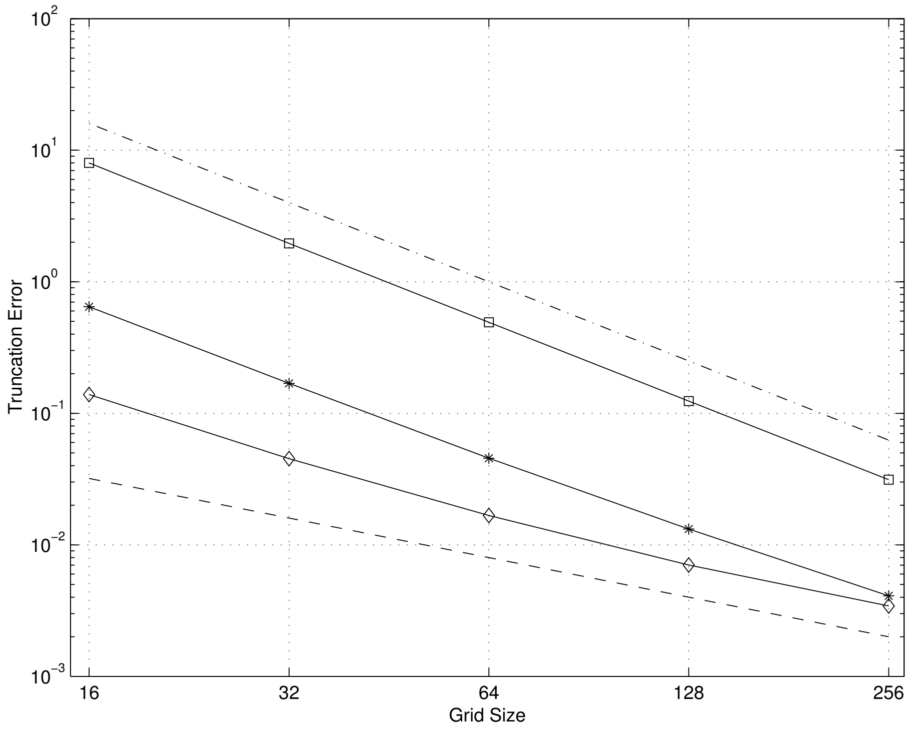

Fig. 5. Truncation error for Neumann boundary conditions. The symbol $\square$ denotes the L^1 norm. The $*$ denotes L^2, and the $\diamond$ denotes the max norm. The reference line for first order is given by $--$, and the reference line for second order is shown with $\cdot - \cdot$.

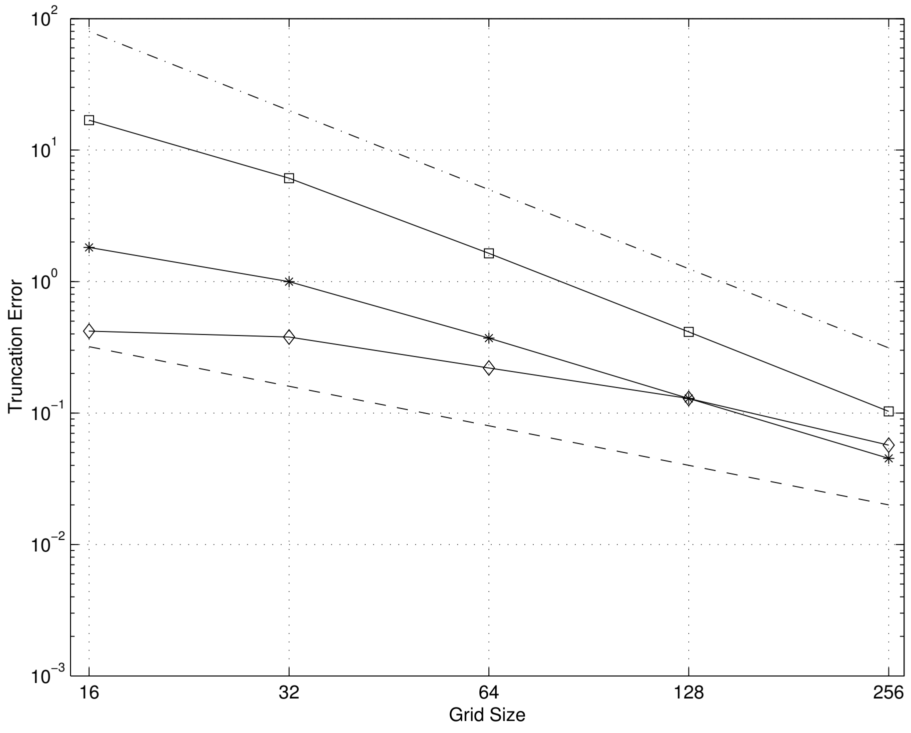

Fig. 6. Truncation error for Dirichlet boundary conditions.The symbol $\square$ denotes the L^1 norm. The $*$ denotes L^2, and the $\diamond$ denotes the max norm. The reference line for first order is given by $--$, and the reference line for second order is shown with $\cdot - \cdot$.

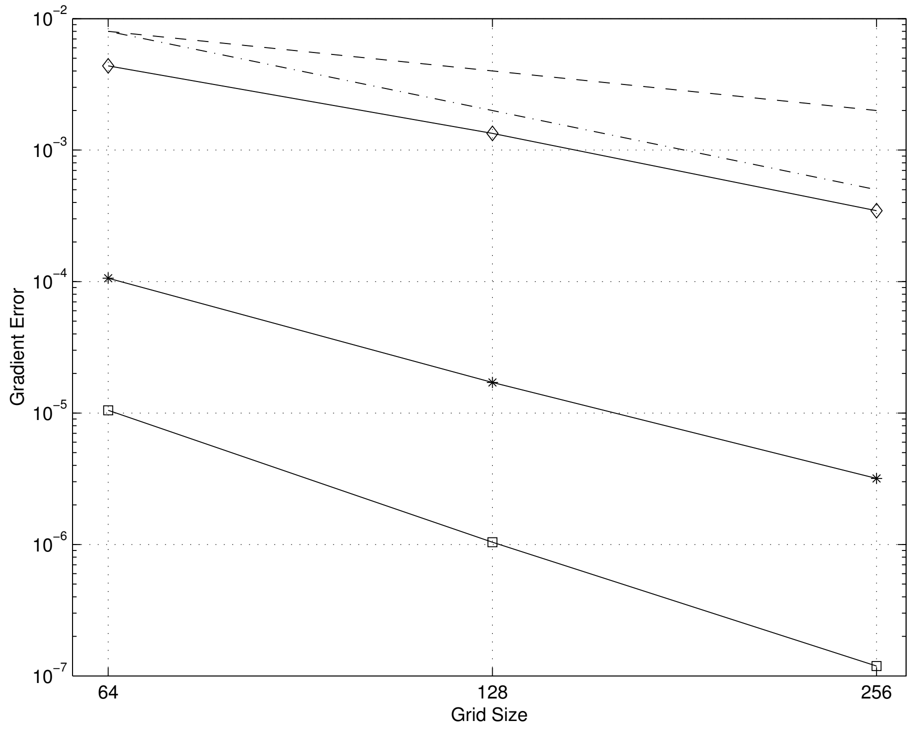

Fig. 7. Convergence of the first component of the gradient of the solution to Poisson's equation with Dirichlet boundary conditions, using (8) to compute the flux. Notation as in figure 6.

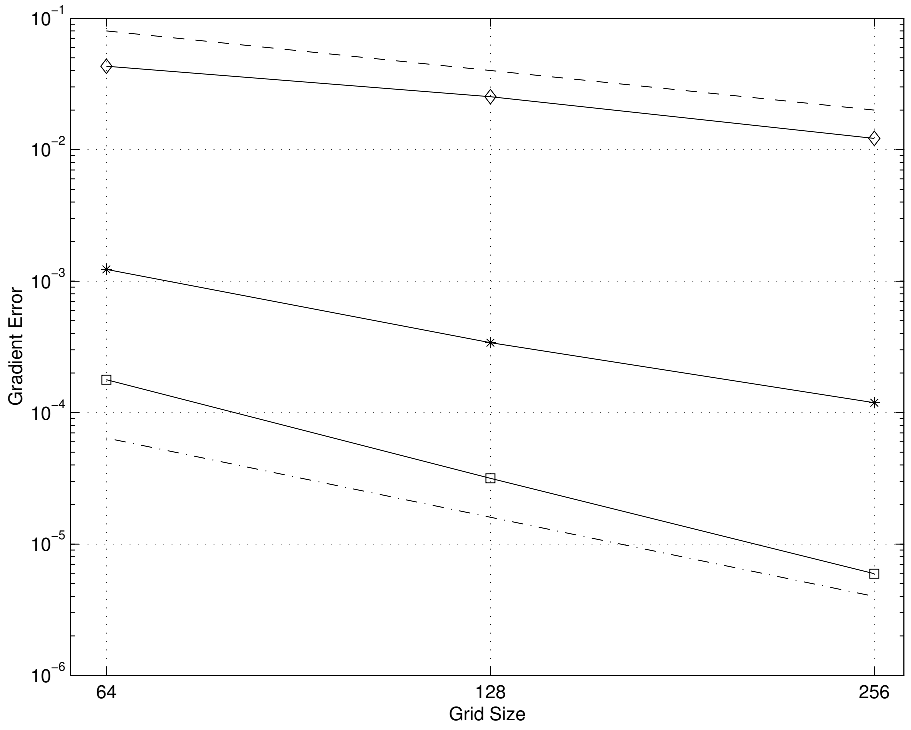

Fig. 8. Convergence of the first component of the gradient of the solution to Poisson's equation conditions, using (10) to compute the flux on the boundary. Notation as in figure 6.

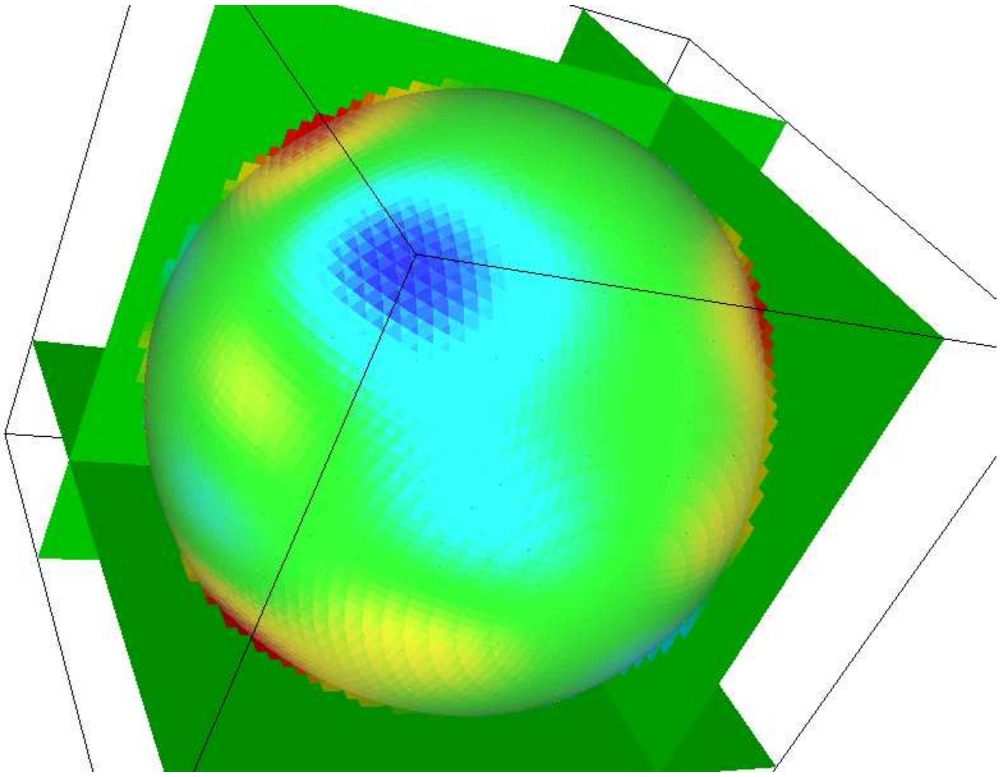

Fig. 9. Solution error for Poisson's problem on a 64^3 grid.

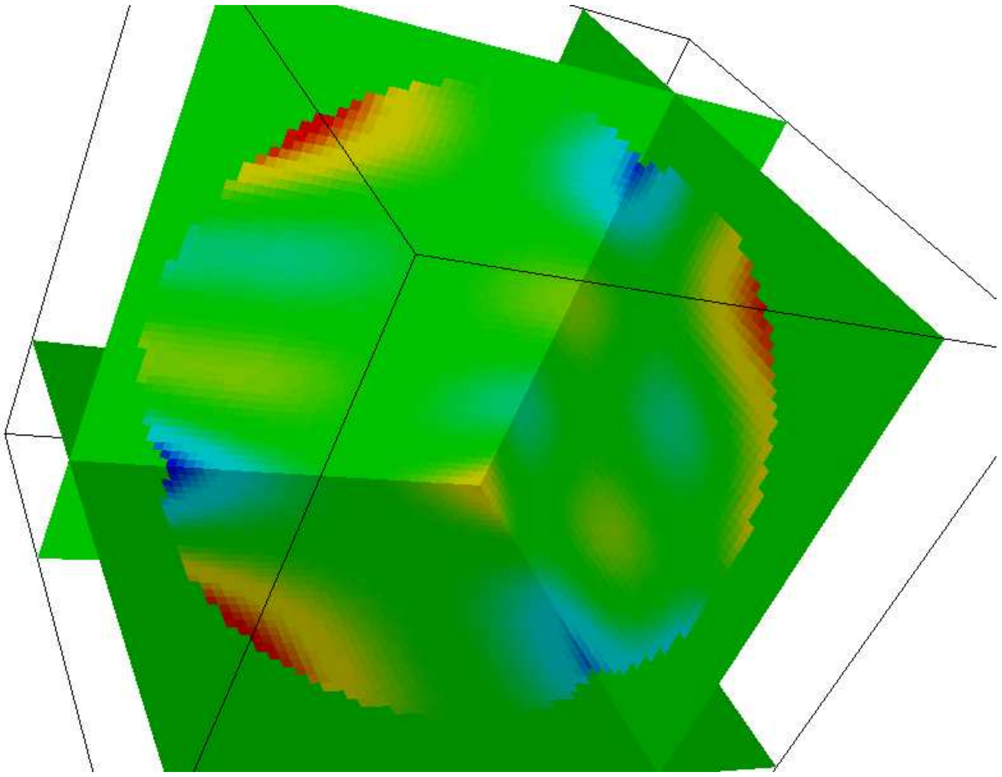
Fig. 10. Solution error for Poisson's problem on a 64^3 grid.

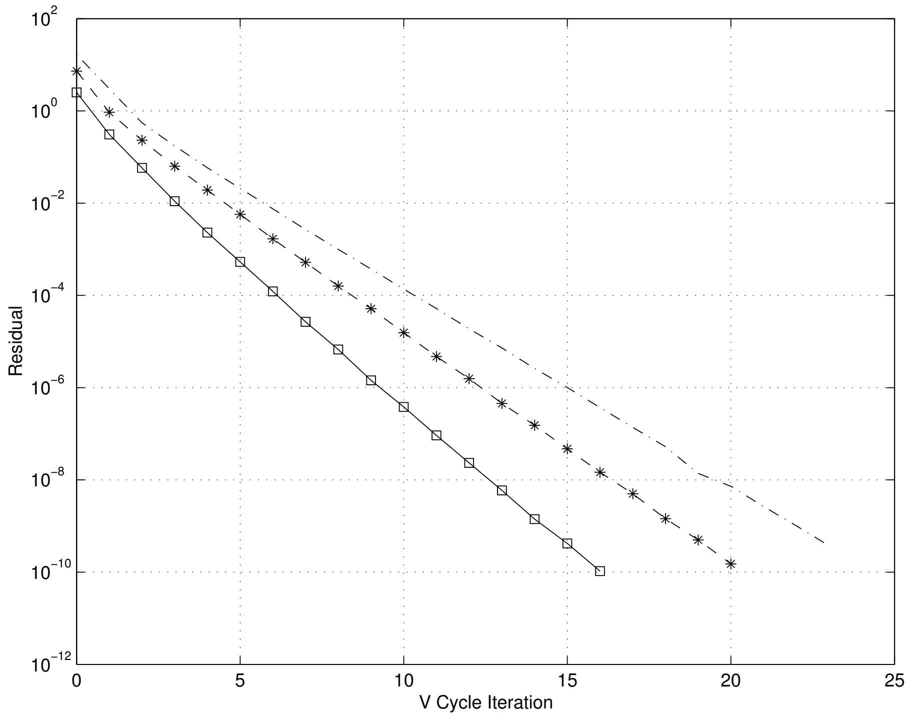

Fig. 11. Plot of the max norm of the partial volume-weighted residual versus V-cycle iteration. The graphs are for meshes of size 64 ($\Box$), 128 ($*$), and 256 ($\cdot - \cdot$).

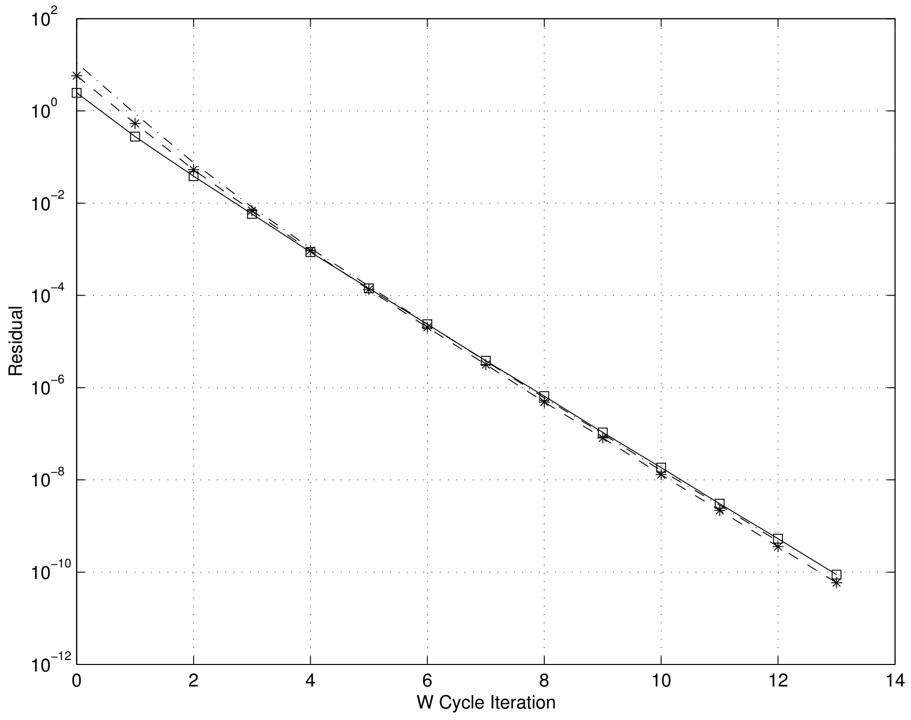

Fig. 12. Plot of the max norm of the partial volume-weighted residual versus W-cycle iteration. The graphs are for meshes of size 64($\square$), 128($*$), and 256($\cdot - \cdot$).

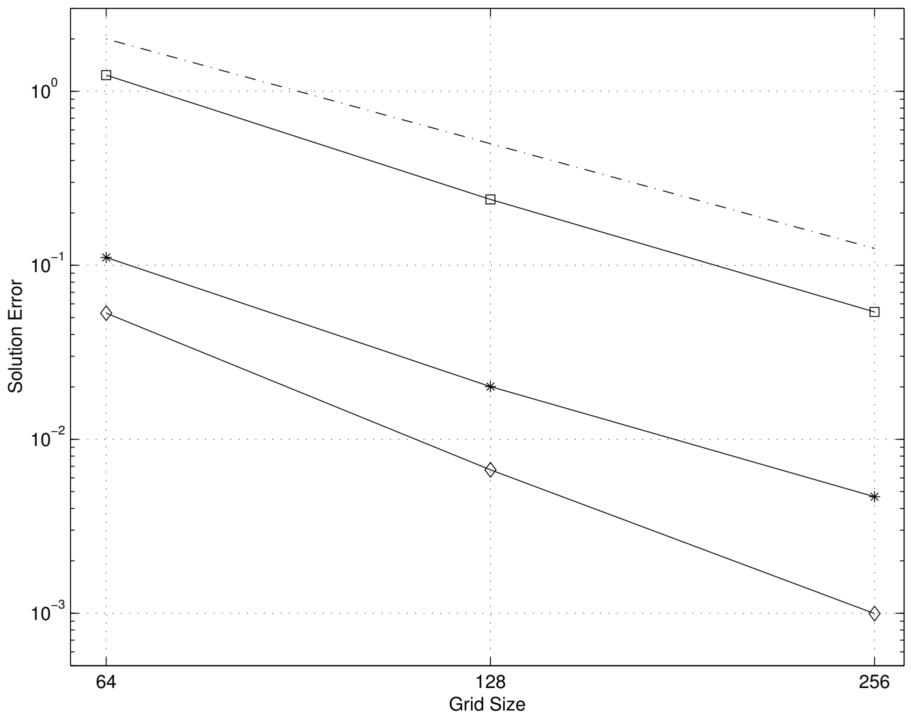

Fig. 13. Solution error for Poisson's equation with Dirichlet boundary conditions. Notation as in figure 6.

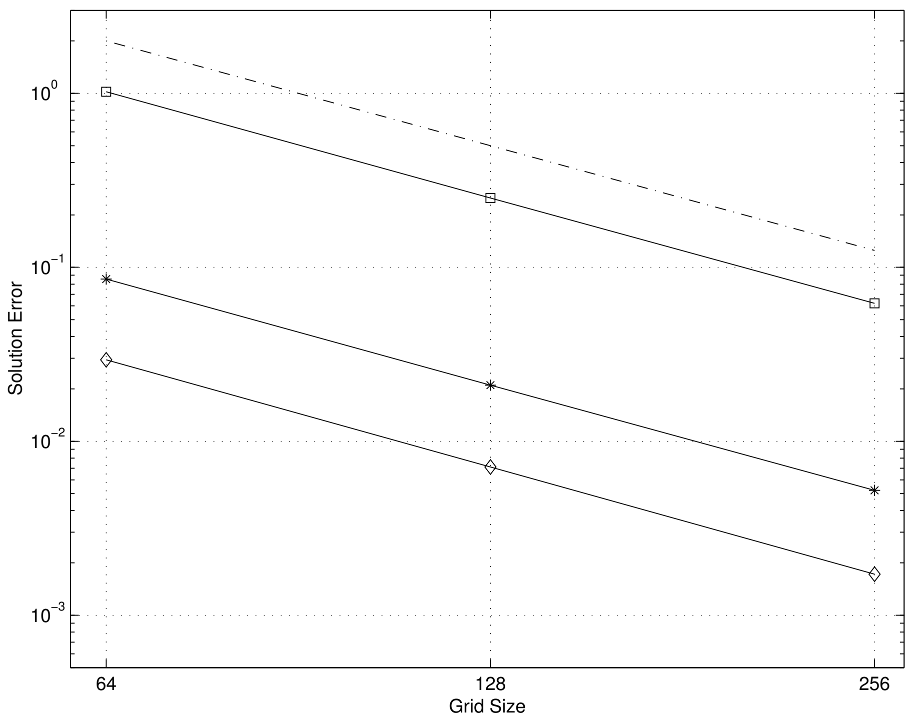

Fig. 14. Solution error for Poisson's equation with Neumann boundary conditions. Notation as in figure 6.

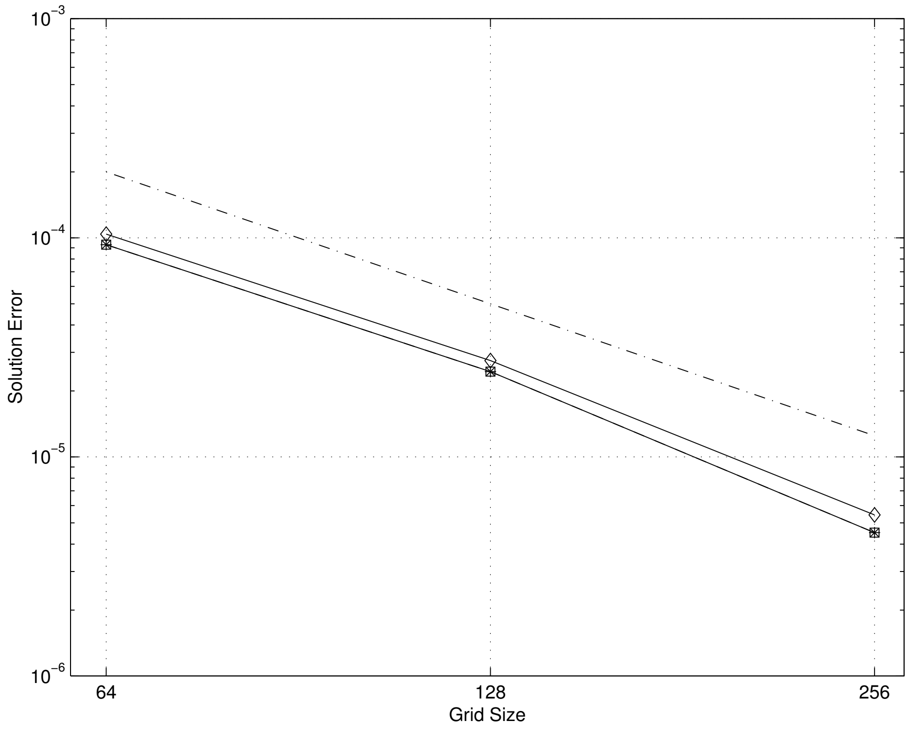

Fig. 15. Solution error for heat equation on the Neutrophil geometry with Neumann boundary conditions. Notation as in figure 6.

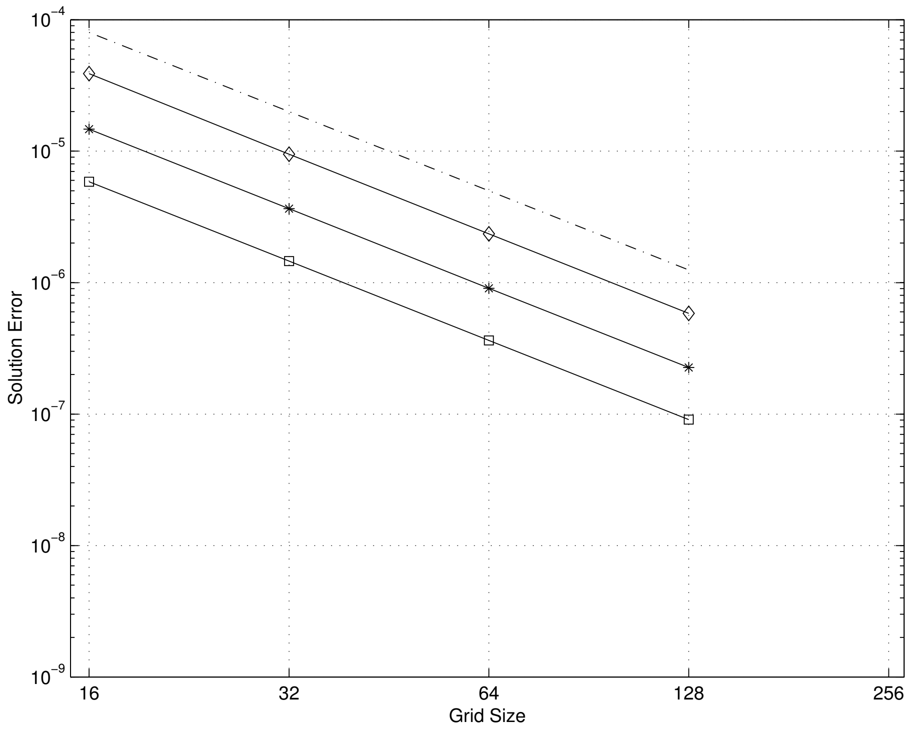

Fig. 16. Solution error for heat equation on a moving domain with Neumann boundary conditions. Notation as in figure 6.

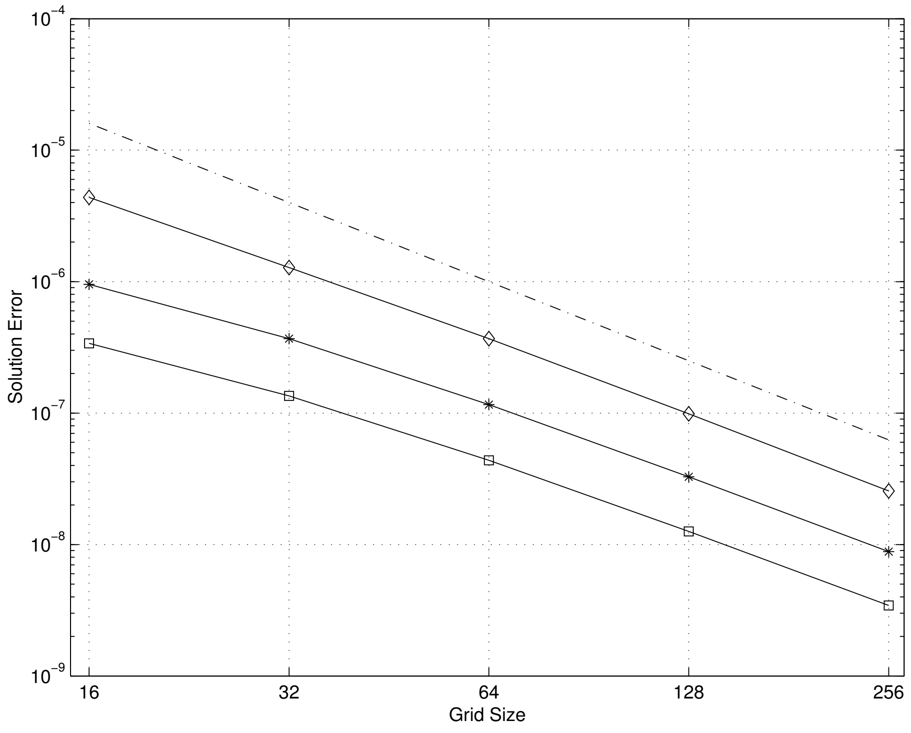

Fig. 17. Solution error for heat equation on a moving domain with Dirichlet boundary conditions. Notation as in figure 6.

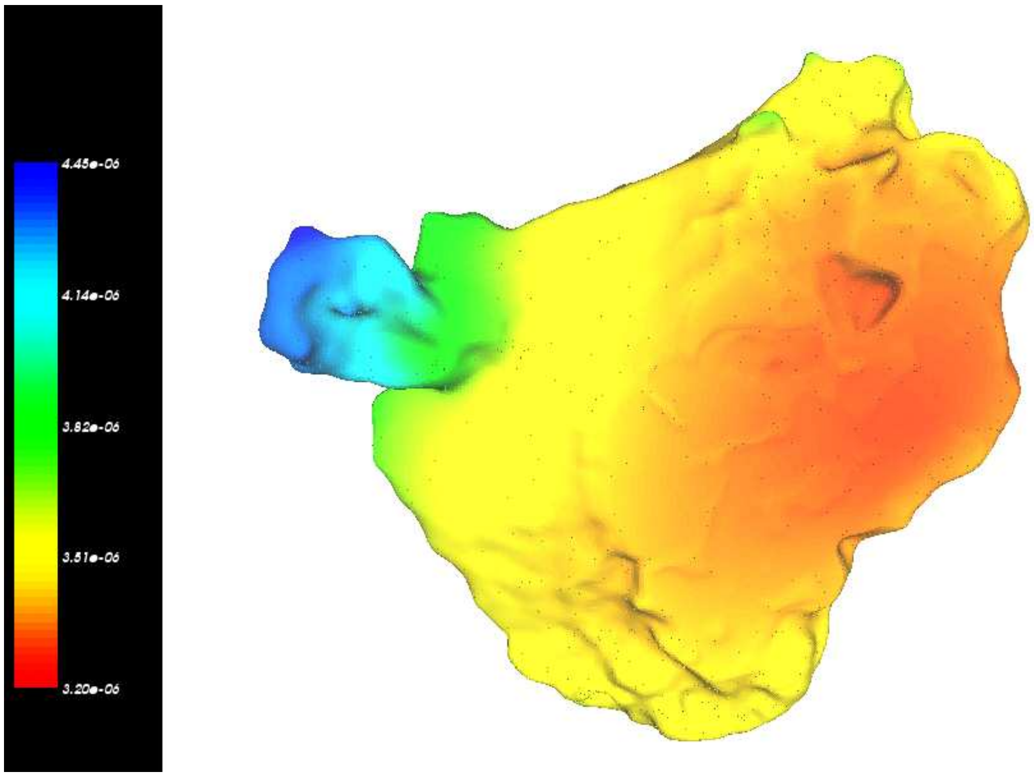

Fig. 18. Solution error for heat equation for the 256^3 Neutrophil Geometry.

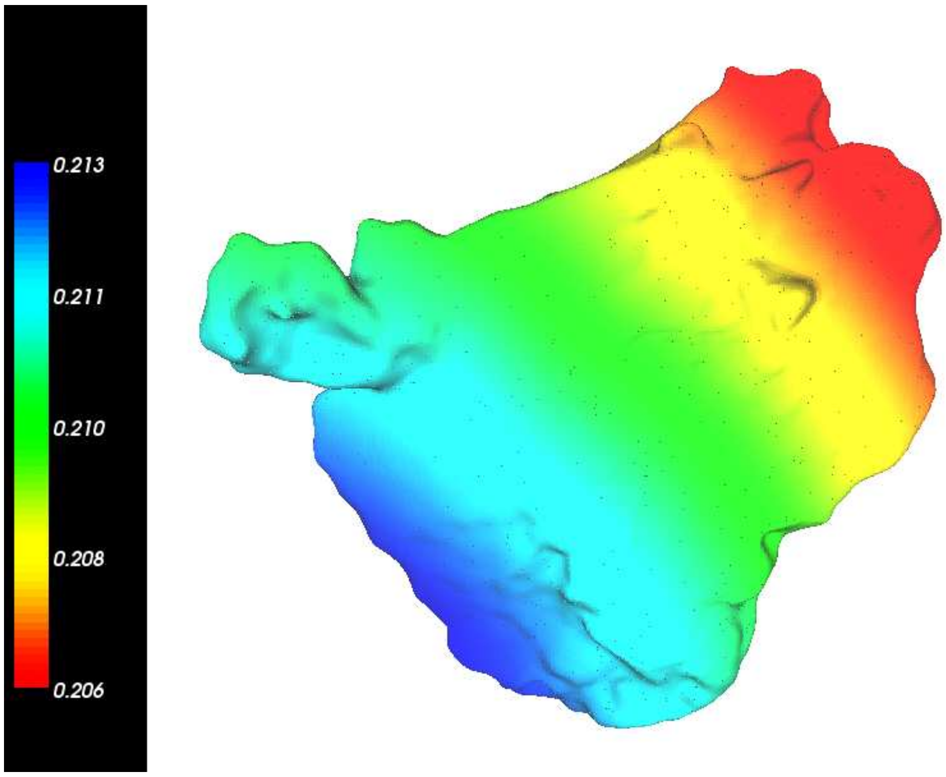

Fig. 19. Solution for heat equation for the 256^3 Neutrophil Geometry.