ACADEMIC PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 184 (2003) 215–243

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

Three-dimensional elliptic solvers for interface problems and applications

Shaozhong Deng[a], Kazufumi Ito[b], Zhilin Li[b,*]

[a] *Department of Mathematics, University of North Carolina at Charlotte, Charlotte, NC 28223, USA*
[b] *Center for Research in Scientific Computation and Department of Mathematics, North Carolina State University, Raleigh, NC 27695, USA*

Received 1 March 2002; received in revised form 2 October 2002; accepted 9 October 2002

Abstract

Second-order accurate elliptic solvers using Cartesian grids are presented for three-dimensional interface problems in which the coefficients, the source term, the solution and its normal flux may be discontinuous across an interface. One of our methods is designed for general interface problems with variable but discontinuous coefficient. The scheme preserves the discrete maximum principle using constrained optimization techniques. An algebraic multigrid solver is applied to solve the discrete system. The second method is designed for interface problems with piecewise constant coefficient. The method is based on the fast immersed interface method and a fast 3D Poisson solver. The second method has been modified to solve Helmholtz/Poisson equations on irregular domains. An application of our method to an inverse interface problem of shape identification is also presented. In this application, the level set method is applied to find the unknown surface iteratively.

Keywords: 3D elliptic interface problem; Irregular domain; Discontinuous coefficient; Discrete maximum preserving scheme; Quadratic optimization; Algebraic multigrid; Shape identification; Level set method

1. Introduction

In this paper, we develop two finite difference methods for three-dimensional interface problems using Cartesian grids. Let Ω be a domain in $\mathbb{R}^3$ and Γ be an arbitrary piecewise smooth surface in Ω. The interface Γ divides Ω into two sub-domains Ω^+ and Ω^- and therefore $\Omega = \Omega^+ \cup \Omega^- \cup \Gamma$. We consider the elliptic equation of the form

$$\nabla \cdot (\beta(x,y,z)\nabla u(x,y,z)) + \lambda(x,y,z)u(x,y,z) = f(x,y,z), \quad (x,y,z) \in \Omega - \Gamma, \tag{1}$$

with a boundary condition on $\partial\Omega$. The coefficients β, λ, and the source term f may be discontinuous across the interface Γ.

* Corresponding author.
E-mail addresses: shaodeng@uncc.edu (S. Deng), kito@math.ncsu.edu (K. Ito), zhilin@math.ncsu.edu (Z. Li).

0021-9991/02/$ - see front matter
PII: S0021-9991(02)00028-1

Due to the discontinuity in the coefficient β, or/and the source/dipole distributions along the interface Γ, the solution and the normal flux may be discontinuous across the interface Γ and can be written as

$$[u] = w(s_1, s_2), \quad [\beta u_n] = v(s_1, s_2), \tag{2}$$

where w and v are two known functions defined only on the interface Γ, $u_n = \partial u/\partial n = \nabla u \cdot \mathbf{n}$ is the limiting normal derivative of u, $\mathbf{n}$ is the unit normal vector pointing to Ω^+ side. The interface Γ is expressed as a parametric form $(x(s_1, s_2), y(s_1, s_2), z(s_1, s_2))$, the jump $[u]$ is defined as the difference of the limiting values of u from Ω^+ and Ω^- sides. We refer the readers to [7–9] for more discussions on the jump conditions.

The problem can be solved by body fitted finite element methods, see [2], for one example; the ghost fluid method (GFM) [12] (which is first-order accurate in the infinity norm but has a symmetric linear system); fast solvers based on integral equations (assuming β is a piecewise constant), see [4,13], for example; the immersed interface method (IIM) reported in [7,17] for example; and possibly some others. These methods have been described in details for two-dimensional problems. Despite the fact that the extension of these methods to three-dimensional (3D) problems may be straightforward, the implementation of these methods in 3D can be very different and few have appeared in the literature.

In this paper, we first develop the maximum principle preserving scheme for the interface problems with variable but discontinuous coefficient by requiring the resulting finite difference matrix be an M-matrix using constrained optimization techniques. The M-matrix condition guarantees the convergence of the algebraic multigrid solver [15] when it is applied to solve the linear system of equations.

When the coefficient β is a piecewise constant, we propose a fast solver by transforming the original problem (1) to a Poisson equation with an *unknown jump* in the normal derivative across the interface Γ. We use a GMRES method to determine the unknown jump so that the original jump in the flux is satisfied and thus the solution to the Poisson equation is also the solution to the original problem (1). There are several advantages of this approach: (1) the computed solution is second-order accurate in the infinity norm; (2) the number of iterations in the GMRES method is almost independent of the mesh sizes; (3) the computed normal derivatives are observed to be second-order accurate as well; (4) with slight changes, the methods can be, and have been applied to Helmholtz/Poisson equations defined on irregular domains.

We also present an application of the second method to an electrical impedance tomography problem in identifying an unknown interface in a 3D domain. The inverse problem is solved iteratively by coupling the level set method [14] with the fast Poisson solver on irregular domains developed in this paper.

The paper is organized as the following. In Section 2, we introduce the interface relations of the problem which will be used in the derivation of our methods. The computational frame is established in Section 3. We propose the maximum principle preserving scheme for general coefficient in Section 4 and provide some numerical examples. The fast Poisson solver for piecewise constant coefficient is proposed in Section 5 with some numerical examples. In Section 6, we present an application to an inverse problem of shape identification. We conclude the paper in Section 7.

2. Theoretical aspects

We hope to develop accurate finite difference methods based on Cartesian grids. To this end, we present a complete set of interface relations up to the second-order derivatives by differentiating the jumps along the interface Γ, and making use of the original partial differential equation (PDE) (1). Since the flux jump condition $[\beta u_n]$ is given in the normal direction of the interface, it is convenient to use a local coordinate system in the normal and tangential directions.

2.1. A local coordinate system

Given a point (x^*, y^*, z^*) on the interface Γ, let $\boldsymbol{\xi}$ be the normal direction of Γ, $\boldsymbol{\eta}$ and $\boldsymbol{\tau}$ be two orthogonal directions tangential to Γ. Then the local coordinates transformation is given by

$$\begin{aligned}
\xi &= (x - x^*)\alpha_{x\xi} + (y - y^*)\alpha_{y\xi} + (z - z^*)\ \alpha_{z\xi},\\
\eta &= (x - x^*)\alpha_{x\eta} + (y - y^*)\alpha_{y\eta} + (z - z^*)\alpha_{z\eta},\\
\tau &= (x - x^*)\alpha_{x\tau} + (y - y^*)\alpha_{y\tau} + (z - z^*)\alpha_{z\tau},
\end{aligned} \tag{3}$$

where $\alpha_{x\xi}$ represents the directional cosine between x-axis and $\boldsymbol{\xi}$, and so forth, see Fig. 1 for an illustration.

The three-dimensional coordinates transformation above can also be written in a matrix–vector form. Define the local transformation matrix as

$$A = \begin{pmatrix} \alpha_{x\xi} & \alpha_{y\xi} & \alpha_{z\xi} \\ \alpha_{x\eta} & \alpha_{y\eta} & \alpha_{z\eta} \\ \alpha_{x\tau} & \alpha_{y\tau} & \alpha_{z\tau} \end{pmatrix}, \tag{4}$$

then we have

$$\begin{pmatrix} \xi \\ \eta \\ \tau \end{pmatrix} = A \begin{pmatrix} x - x^* \\ y - y^* \\ z - z^* \end{pmatrix}. \tag{5}$$

Also, for any differentiable function $p(x, y, z)$, we have

$$\begin{pmatrix} \bar{p}_\xi \\ \bar{p}_\eta \\ \bar{p}_\tau \end{pmatrix} = A \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}, \tag{6}$$

where $\bar{p}(\xi, \eta, \tau) = p(x, y, z)$, and

$$\begin{pmatrix} \bar{p}_{\xi\xi} & \bar{p}_{\xi\eta} & \bar{p}_{\xi\tau} \\ \bar{p}_{\eta\xi} & \bar{p}_{\eta\eta} & \bar{p}_{\eta\tau} \\ \bar{p}_{\tau\xi} & \bar{p}_{\tau\eta} & \bar{p}_{\tau\tau} \end{pmatrix} = A \begin{pmatrix} p_{xx} & p_{xy} & p_{xz} \\ p_{yx} & p_{yy} & p_{yz} \\ p_{zx} & p_{zy} & p_{zz} \end{pmatrix} A^{\mathrm{T}}, \tag{7}$$

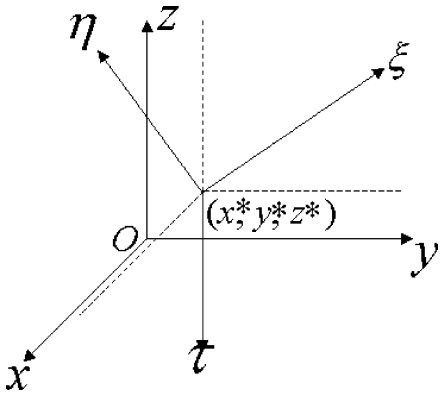

Fig. 1. A sketch of a three-dimensional local coordinates transformation.

where A^{T} is the transpose of A. It is easy to verify that $A^{\mathrm{T}}A = I$, where I is the identity matrix. For two-dimensional problems, the matrix–vector relations under the local coordinates can be found in [3]. Note that under the local coordinates transformation (3), the PDE (1) is invariant. Therefore, we will drop the bars for simplicity.

2.2. Interface relations

We use the superscripts + or − to denote the limiting values of a function from Ω^+ side and Ω^- side of the interface, respectively. Under the local coordinates, the limiting differential equation from the negative side, for example, can be written as

$$\beta^-(u^-_{\xi\xi} + u^-_{\eta\eta} + u^-_{\tau\tau}) + \beta^-_\xi u^-_\xi + \beta^-_\eta u^-_\eta + \beta^-_\tau u^-_\tau + \lambda^- u^- - f^- = 0. \tag{8}$$

Also under the local ξ–η–τ coordinate system, the interface can be expressed as

$$\xi = \chi(\eta,\tau), \quad \text{with } \chi(0,0) = 0,\ \chi_\eta(0,0) = 0,\ \chi_\tau(0,0) = 0. \tag{9}$$

We will use the jump condition (2) and the original differential equation to get more interface relations in this section.

Let us first differentiate the first jump condition $[u] = w$ in (2) with respect to η and τ, respectively, to get

$$[u_\xi]\chi_\eta + [u_\eta] = w_\eta, \tag{10}$$

$$[u_\xi]\chi_\tau + [u_\tau] = w_\tau. \tag{11}$$

Differentiating (10) with respect to τ yields

$$\chi_\eta \frac{\partial}{\partial\tau}[u_\xi] + \chi_{\eta\tau}[u_\xi] + [u_{\eta\xi}]\chi_\tau + [u_{\eta\tau}] = w_{\eta\tau}. \tag{12}$$

Differentiating (10) with respect to η and differentiating (11) with respect to τ, respectively, we obtain

$$\chi_\eta \frac{\partial}{\partial\eta}[u_\xi] + \chi_{\eta\eta}[u_\xi] + \chi_\eta[u_{\eta\xi}] + [u_{\eta\eta}] = w_{\eta\eta}, \tag{13}$$

$$\chi_\tau \frac{\partial}{\partial\tau}[u_\xi] + \chi_{\tau\tau}[u_\xi] + \chi_\tau[u_{\tau\xi}] + [u_{\tau\tau}] = w_{\tau\tau}. \tag{14}$$

Before differentiating the jump of the normal derivative $[\beta u_n] = v$ in (2), we first express the unit normal vector of the interface Γ as

$$\mathbf{n} = \frac{(1, -\chi_\eta, -\chi_\tau)}{\sqrt{1 + \chi_\eta^2 + \chi_\tau^2}}. \tag{15}$$

So the second interface condition $[\beta u_n] = v$ in (2) can be written as

$$[\beta(u_\xi - u_\eta\chi_\eta - u_\tau\chi_\tau)] = v(\eta,\tau)\sqrt{1 + \chi_\eta^2 + \chi_\tau^2}. \tag{16}$$

Differentiating this with respect to η gives

$$[(\beta_\xi\chi_\eta+\beta_\eta)(u_\xi-u_\eta\chi_\eta-u_\tau\chi_\tau)]+\left[\beta\left(u_{\xi\xi}\chi_\eta+u_{\xi\eta}-\chi_\eta\frac{\partial}{\partial\eta}u_\eta-\chi_\tau\frac{\partial}{\partial\eta}u_\tau-u_\eta\chi_{\eta\eta}-u_\tau\chi_{\eta\tau}\right)\right]$$
$$=v_\eta\sqrt{1+\chi_\eta^2+\chi_\tau^2}+v\frac{\chi_\eta\chi_{\eta\eta}}{\sqrt{1+\chi_\eta^2+\chi_\tau^2}}. \tag{17}$$

Similarly, differentiating (16) with respect to τ gives

$$[(\beta_\xi\chi_\tau+\beta_\tau)(u_\xi-u_\eta\chi_\eta-u_\tau\chi_\tau)]+\left[\beta\left(u_{\xi\xi}\chi_\tau+u_{\xi\tau}-\chi_\eta\frac{\partial}{\partial\tau}u_\eta-\chi_\tau\frac{\partial}{\partial\tau}u_\tau-u_\eta\chi_{\eta\tau}-u_\tau\chi_{\tau\tau}\right)\right]$$
$$=v_\tau\sqrt{1+\chi_\eta^2+\chi_\tau^2}+v\frac{\chi_\tau\chi_{\tau\tau}}{\sqrt{1+\chi_\eta^2+\chi_\tau^2}}. \tag{18}$$

At the origin, $\chi_\eta(0,0)=\chi_\tau(0,0)=0$, and from (10) to (18) we can conclude that

$$\begin{aligned}
&u^+=u^-+w,\\
&u_\xi^+=\frac{\beta^-}{\beta^+}u_\xi^-+\frac{v}{\beta^+},\\
&u_\eta^+=u_\eta^-+w_\eta,\\
&u_\tau^+=u_\tau^-+w_\tau,\\
&u_{\eta\tau}^+=u_{\eta\tau}^-+(u_\xi^--u_\xi^+)\chi_{\eta\tau}+w_{\eta\tau},\\
&u_{\eta\eta}^+=u_{\eta\eta}^-+(u_\xi^--u_\xi^+)\chi_{\eta\eta}+w_{\eta\eta},\\
&u_{\tau\tau}^+=u_{\tau\tau}^-+(u_\xi^--u_\xi^+)\chi_{\tau\tau}+w_{\tau\tau},\\
&u_{\xi\eta}^+=\frac{\beta^-}{\beta^+}u_{\xi\eta}^-+\left(u_\eta^+-\frac{\beta^-}{\beta^+}u_\eta^-\right)\chi_{\eta\eta}+\left(u_\tau^+-\frac{\beta^-}{\beta^+}u_\tau^-\right)\chi_{\eta\tau}+\frac{\beta_\eta^-}{\beta^+}u_\xi^--\frac{\beta_\eta^+}{\beta^+}u_\xi^++\frac{v_\eta}{\beta^+},\\
&u_{\xi\tau}^+=\frac{\beta^-}{\beta^+}u_{\xi\tau}^-+\left(u_\eta^+-\frac{\beta^-}{\beta^+}u_\eta^-\right)\chi_{\eta\tau}+\left(u_\tau^+-\frac{\beta^-}{\beta^+}u_\tau^-\right)\chi_{\tau\tau}+\frac{\beta_\tau^-}{\beta^+}u_\xi^--\frac{\beta_\tau^+}{\beta^+}u_\xi^++\frac{v_\tau}{\beta^+}.
\end{aligned} \tag{19}$$

To get the relation for $u_{\xi\xi}^+$ we need to use the differential equation (1) itself from which we can write

$$[\beta(u_{\xi\xi}+u_{\eta\eta}+u_{\tau\tau})+\beta_\xi u_\xi+\beta_\eta u_\eta+\beta_\tau u_\tau+\lambda u]=[f]. \tag{20}$$

Notice that

$$\lambda^-u^--\lambda^+u^+=\lambda^-u^--\lambda^+u^-+\lambda^+u^--\lambda^+u^+=-[\lambda]u^--\lambda^+[u]. \tag{21}$$

Rearranging Eq. (20) and using Eq. (21) above we get

$$\beta^+(u_{\xi\xi}^++u_{\eta\eta}^++u_{\tau\tau}^+)+\beta_\xi^+u_\xi^++\beta_\eta^+u_\eta^++\beta_\tau^+u_\tau^+=\beta^-(u_{\xi\xi}^-+u_{\eta\eta}^-+u_{\tau\tau}^-)+\beta_\xi^-u_\xi^-+\beta_\eta^-u_\eta^-+\beta_\tau^-u_\tau^-+[f]+\lambda^-u^--\lambda^+u^+. \tag{22}$$

Plugging the sixth and seventh equations of (19) in (22) and collecting terms finally we have

$$\begin{aligned}
u_{\xi\xi}^+=&\frac{\beta^-}{\beta^+}u_{\xi\xi}^-+\left(\frac{\beta^-}{\beta^+}-1\right)u_{\eta\eta}^-+\left(\frac{\beta^-}{\beta^+}-1\right)u_{\tau\tau}^-+u_\xi^+\left(\chi_{\eta\eta}+\chi_{\tau\tau}-\frac{\beta_\xi^+}{\beta^+}\right)-u_\xi^-\left(\chi_{\eta\eta}+\chi_{\tau\tau}-\frac{\beta_\xi^-}{\beta^+}\right)\\
&+\frac{1}{\beta^+}(\beta_\eta^-u_\eta^--\beta_\eta^+u_\eta^+)+\frac{1}{\beta^+}(\beta_\tau^-u_\tau^--\beta_\tau^+u_\tau^+)-\frac{1}{\beta^+}([\lambda]u^-+\lambda^+[u])+\frac{[f]}{\beta^+}-w_{\eta\eta}-w_{\tau\tau}.
\end{aligned} \tag{23}$$

3. The computational frame

For simplicity, we assume that the domain Ω is a solid cube, say $[a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]$. We wish to solve the problem using a finite difference method and a uniform Cartesian grid with

$$x_i = a_1 + ih, \quad y_j = a_2 + jh, \quad z_k = a_3 + kh, \quad 0 \leqslant i \leqslant L, \ 0 \leqslant j \leqslant M, \quad 0 \leqslant k \leqslant N.$$

We also assume that $h = (b_1 - a_1)/L = (b_2 - a_2)/M = (b_3 - a_3)/N$ to make many expressions simple.

We use the zero level surface of a three-dimensional function $\varphi(x, y, z)$ to express the interface, that is,

$$\begin{aligned} &\varphi(x, y, z) < 0 \quad \text{if } (x, y, z) \in \Omega^-, \\ &\varphi(x, y, z) = 0 \quad \text{if } (x, y, z) \in \Gamma, \\ &\varphi(x, y, z) > 0 \quad \text{if } (x, y, z) \in \Omega^+. \end{aligned} \tag{24}$$

We assume that the level set function is Lipschitz continuous and $\varphi(x, y, z) \in C^2$ in the small neighborhood of the zero level set $\varphi = 0$ that represents the interface Γ. At a grid point $\mathbf{x}_{ijk}$, let φ_{ijk}^{min} and φ_{ijk}^{max} be the minimum and maximum values of the level set function φ at $\varphi_{i\pm1,j,k}$, $\varphi_{i,j\pm1,k}$, $\varphi_{i,j,k\pm1}$, and φ_{ijk}. We define $\mathbf{x}_{ijk}$ as an *irregular grid point* if

$$\varphi_{ijk}^{\max} \varphi_{ijk}^{\min} \leqslant 0. \tag{25}$$

Otherwise $\mathbf{x}_{ijk}$ is called a *regular grid point*.

3.1. Setting-up a local coordinate system using the level set function

Given a point $\mathbf{X}^* = (x^*, y^*, z^*)$ on the interface, we choose the $\boldsymbol{\xi}$ direction as the normal direction of the interface

$$\boldsymbol{\xi} = \frac{\nabla\varphi}{|\nabla\varphi|} = (\varphi_x, \ \varphi_y, \ \varphi_z)^{\mathrm{T}} / \sqrt{\varphi_x^2 + \varphi_y^2 + \varphi_z^2},$$

where the unit normal direction is evaluated at (x^*, y^*, z^*). The η- and τ-axes are in the tangent plane passing through (x^*, y^*, z^*). We choose the first tangential direction as

$$\boldsymbol{\eta} = (\varphi_y, \ -\varphi_x, \ 0)^{\mathrm{T}} / \sqrt{\varphi_x^2 + \varphi_y^2}$$

if $\varphi_x^2 + \varphi_y^2 \neq 0$. Otherwise, we choose

$$\boldsymbol{\eta} = (\varphi_z, \ 0, \ -\varphi_x)^{\mathrm{T}} / \sqrt{\varphi_x^2 + \varphi_z^2}.$$

The corresponding second tangential direction is

$$\boldsymbol{\tau} = \frac{\mathbf{s}}{|\mathbf{s}|}, \quad \text{where } \mathbf{s} = (\varphi_x\varphi_z, \ \varphi_y\varphi_z, \ -\varphi_x^2 - \varphi_y^2)^{\mathrm{T}}$$

if $\varphi_x^2 + \varphi_y^2 \neq 0$. Otherwise, we choose

$$\boldsymbol{\tau} = \frac{\mathbf{t}}{|\mathbf{t}|}, \quad \text{where } \mathbf{t} = (-\varphi_x\varphi_y, \ \varphi_x^2 + \varphi_z^2, \ -\varphi_y\varphi_z)^{\mathrm{T}}.$$

3.2. Computing the projections

For each irregular grid point $\mathbf{x} = (x_i, y_j, z_k)$, we select a point $\mathbf{X}^* = (x_i^*, y_j^*, z_k^*)$ on the interface. Although not necessarily, we take this point as the projection of (x_i, y_j, z_k) on the interface. The projection $\mathbf{X}^*$ is the closest point on the interface to the grid point (x_i, y_j, z_k) with $\varphi(\mathbf{X}^*) = 0$. In practice, we can only compute the projection approximately. Since the interface Γ is represented as the zero level surface $\varphi = 0$, and the level set function φ has the fastest rate of increase/decrease in the normal direction of the level surfaces, we write the projection as

$$\mathbf{X}^* = \mathbf{x} + \alpha \mathbf{p},$$

where $\mathbf{p} = \nabla\varphi/|\nabla\varphi|$, and $\alpha \sim h$ is an approximation of the signed distance from the grid point $\mathbf{x}$ to the projection $\mathbf{X}^*$. Neglecting $O(\alpha^3)$ and higher-order terms in the Taylor expansion of $\varphi(\mathbf{X}^*) = 0$, we get a quadratic equation for α

$$\varphi(\mathbf{x}) + |\nabla\varphi|\alpha + \frac{1}{2}(\mathbf{p}^{\mathrm{T}}\,\mathrm{He}(\varphi)\,\mathbf{p})\alpha^2 = 0,$$

where $\mathrm{He}(\varphi)$ is the Hessian matrix of φ

$$\mathrm{He}(\varphi) = \begin{pmatrix} \varphi_{xx} & \varphi_{xy} & \varphi_{xz} \\ \varphi_{yx} & \varphi_{yy} & \varphi_{yz} \\ \varphi_{zx} & \varphi_{zy} & \varphi_{zz} \end{pmatrix}.$$

The values of φ, φ_x, φ_y, φ_z, φ_{xx}, φ_{xy}, φ_{xz}, φ_{yy}, φ_{yz}, and φ_{zz} are all computed at the grid point $\mathbf{x} = (x_i, y_j, z_k)$ using the standard central finite difference scheme. Since the truncation error of the above quadratic expansion is of $O(\alpha^3) \sim O(h^3)$, the central finite difference schemes are second-order accurate, and the quantities φ_x, φ_y, and φ_z appear in the linear and quadratic terms of α and the second-order derivatives $\varphi_{xx}, \ldots, \varphi_{zz}$ appear in the quadratic term of α, the computed projections are third-order accurate.

4. The maximum principle preserving scheme

We now derive a finite difference equation of the form

$$\sum_{m}^{n_s} \gamma_m u_{i+i_m, j+j_m, k+k_m} + \lambda_{ijk} u_{ijk} = f_{ijk} + C_{ijk} \tag{26}$$

at every grid point (x_i, y_j, z_k) to approximate (1), where i_m, j_m, k_m take values from $0, \pm 1, \pm 2, \ldots$, meaning that the summation is taken over the neighboring grid points centered at (x_i, y_j, z_k). Note that we have omitted the dependency of m on i, j, and k, for simplicity. At a regular grid point, we use the standard seven point finite difference scheme

$$\begin{aligned}
&\gamma_{i-1,jk} = \frac{\beta_{i-1/2,j,k}}{h^2}, \quad \gamma_{i+1,jk} = \frac{\beta_{i+1/2,j,k}}{h^2}, \quad \gamma_{i,j-1,k} = \frac{\beta_{i,j-1/2,k}}{h^2}, \\
&\gamma_{i,j+1,k} = \frac{\beta_{i,j+1/2,k}}{h^2}, \quad \gamma_{i,j,k-1} = \frac{\beta_{i,j,k-1/2}}{h^2}, \quad \gamma_{i,j,k+1} = \frac{\beta_{i,j,k+1/2}}{h^2}, \\
&\gamma_{i,j,k} = -(\gamma_{i-1,jk} + \gamma_{i+1,jk} + \gamma_{i,j-1,k} + \gamma_{i,j+1,k} + \gamma_{i,j,k-1} + \gamma_{i,j,k+1}) \quad C_{ijk} = 0,
\end{aligned} \tag{27}$$

where $\beta_{i-1/2,j,k} = \beta(x_i - h/2, y_j, z_k)$ and so forth. The local truncation errors are $O(h^2)$.

4.1. Setting-up the finite difference equations at irregular grid points

At irregular grid points, we use the method of un-determined coefficients to find the coefficients of the finite difference scheme. Let $\mathbf{X}^*$ be the projection of (x_i, y_j, z_k) on the interface. Using the Taylor expansion from both sides of the interface at $\mathbf{X}^*$, we can write

$$\begin{aligned}\sum_m^{n_s} \gamma_m u(x_{i+i_m}, y_{j+j_m}, z_{k+k_m}) + \lambda_{ijk} u(x_i, y_j, z_k) = \sum_m^{n_s} \gamma_m \Big(& u^{\pm} + u^{\pm}_{\xi}\xi_m + u^{\pm}_{\eta}\eta_m + u^{\pm}_{\tau}\tau_m + \frac{\xi_m^2}{2} u^{\pm}_{\xi\xi} + \frac{\eta_m^2}{2} u^{\pm}_{\eta\eta} \\ & + \frac{\tau_m^2}{2} u^{\pm}_{\tau\tau} + \xi_m\eta_m u^{\pm}_{\xi\eta} + \xi_m\tau_m u^{\pm}_{\xi\tau} + \eta_m\tau_m u^{\pm}_{\eta\tau} + \mathrm{O}(h^3) \Big) \\ & + \lambda_{ijk}(u_{ijk} + \mathrm{O}(h)),\end{aligned} \tag{28}$$

where the function values and the derivatives are defined as the limiting value at $\mathbf{X}^*$ from the side where the grid point $(i + i_m, j + j_m, k + k_m)$ is in. Using the interface relations (19) and (23), the expression above can be written as

$$\begin{aligned}\sum_m^{n_s} \gamma_m u(x_{i+i_m}, y_{j+j_m}, z_{k+k_m}) + \lambda_{ijk} u(x_i, y_j, z_k) \approx & a_1 u^- + a_2 u^+ + a_3 u^-_{\xi} + a_4 u^+_{\xi} + a_5 u^-_{\eta} + a_6 u^+_{\eta} + a_7 u^-_{\tau} \\ & + a_8 u^+_{\tau} + a_9 u^-_{\xi\xi} + a_{10} u^+_{\xi\xi} + a_{11} u^-_{\eta\eta} + a_{12} u^+_{\eta\eta} + a_{13} u^-_{\tau\tau} \\ & + a_{14} u^+_{\tau\tau} + a_{15} u^-_{\xi\eta} + a_{16} u^+_{\xi\eta} + a_{17} u^-_{\xi\tau} + a_{18} u^+_{\xi\tau} + a_{19} u^-_{\eta\tau} \\ & + a_{20} u^+_{\eta\tau} + \lambda^- u^-,\end{aligned} \tag{29}$$

with the higher-order terms being neglected, where the coefficients a_i's depend only on the position of the stencil relative to the interface. They are independent of the functions u, β, λ, and f. If we define the index sets K^+ and K^- by

$$K^{\pm} = \{m : (\xi_m, \eta_m, \tau_m) \text{ is on the } \pm \text{ side of } \Gamma\}, \tag{30}$$

then the a_i's with odd subscript are given by

$$\begin{aligned} & a_1 = \sum_{m\in K^-} \gamma_m, \quad a_3 = \sum_{m\in K^-} \gamma_m \xi_m, \quad a_5 = \sum_{m\in K^-} \gamma_m \eta_m, \quad a_7 = \sum_{m\in K^-} \gamma_m \tau_m, \\ & a_9 = \frac{1}{2}\sum_{m\in K^-} \gamma_m \xi_m^2, \quad a_{11} = \frac{1}{2}\sum_{m\in K^-} \gamma_m \eta_m^2, \quad a_{13} = \frac{1}{2}\sum_{m\in K^-} \gamma_m \tau_m^2, \\ & a_{15} = \sum_{m\in K^-} \gamma_m \xi_m \eta_m, \quad a_{17} = \sum_{m\in K^-} \gamma_m \xi_m \tau_m, \quad a_{19} = \sum_{m\in K^-} \gamma_m \eta_m \tau_m. \end{aligned} \tag{31}$$

The a_i's with even subscript are exactly the same as above except the summation is from the subset K^+. Substituting the interface relations (19) and (23), we express all the quantities from the positive side in terms of the quantities from the negative side. Thus the right-hand side of (29) is represented by the linear combination of the quantities from the negative side. After some manipulations, (29) is arranged as follows:

$$\begin{aligned}\sum_m^{n_s} \gamma_m u(x_{i+i_m}, y_{j+j_m}, z_{k+k_m}) + \lambda_{ijk} u(x_i, y_j, z_k) = & (\quad) u^- + (\quad) u^-_{\xi} + (\quad) u^-_{\eta} + (\quad) u^-_{\tau} + (\quad) u^-_{\xi\xi} + (\quad) u^-_{\eta\eta} \\ & + (\quad) u^-_{\tau\tau} + (\quad) u^-_{\xi\eta} + (\quad) u^-_{\xi\tau} + (\quad) u^-_{\eta\tau} + C_{ijk}.\end{aligned} \tag{32}$$

The contents in the parentheses are the corresponding terms in the left-hand side of (33)–(42). The last term C_{ijk} is a linear function of the jumps in the solution and the flux and is given in (43). We want the finite

difference scheme to be second-order accurate to the differential equation. Therefore at $\mathbf{X}^*$ we match (29) with the PDE (8) from the negative side, i.e., we equate (32) to $\beta^-(u^-_{\xi\xi}+u^-_{\eta\eta}+u^-_{\tau\tau})+\beta^-_\xi u^-_\xi+\beta^-_\eta u^-_\eta+\beta^-_\tau u^-_\tau$. Hence, we obtain the linear system of equations for the coefficients γ_i's below:

$$a_1 - a_{10}\frac{[\lambda]}{\beta^+} + a_2 = 0, \tag{33}$$

$$\begin{aligned} &a_3 - a_{10}\left(\chi_{\eta\eta}+\chi_{\tau\tau}-\frac{\beta^-_\xi}{\beta^+}\right) + a_{12}\chi_{\eta\eta} + a_{14}\chi_{\tau\tau} + a_{16}\frac{\beta^-_\eta}{\beta^+} + a_{18}\frac{\beta^-_\tau}{\beta^+} + a_{20}\chi_{\eta\tau} \\ &\quad + \frac{\beta^-}{\beta^+}\left\{a_4 + a_{10}\left(\chi_{\eta\eta}+\chi_{\tau\tau}-\frac{\beta^+_\xi}{\beta^+}\right) - a_{12}\chi_{\eta\eta} - a_{14}\chi_{\tau\tau} - a_{16}\frac{\beta^+_\eta}{\beta^+} - a_{18}\frac{\beta^+_\tau}{\beta^+} - a_{20}\chi_{\eta\tau}\right\} = \beta^-_\xi, \end{aligned} \tag{34}$$

$$a_5 + a_{10}\frac{\beta^-_\eta}{\beta^+} - a_{16}\frac{\beta^-}{\beta^+}\chi_{\eta\eta} - a_{18}\frac{\beta^-}{\beta^+}\chi_{\eta\tau} + a_6 - a_{10}\frac{\beta^+_\eta}{\beta^+} + a_{16}\chi_{\eta\eta} + a_{18}\chi_{\eta\tau} = \beta^-_\eta, \tag{35}$$

$$a_7 + a_{10}\frac{\beta^-_\tau}{\beta^+} - a_{16}\frac{\beta^-}{\beta^+}\chi_{\eta\tau} - a_{18}\frac{\beta^-}{\beta^+}\chi_{\tau\tau} + a_8 - a_{10}\frac{\beta^+_\tau}{\beta^+} + a_{16}\chi_{\eta\tau} + a_{18}\chi_{\tau\tau} = \beta^-_\tau, \tag{36}$$

$$a_9 + a_{10}\frac{\beta^-}{\beta^+} = \beta^-, \tag{37}$$

$$a_{11} + a_{12} + a_{10}\left(\frac{\beta^-}{\beta^+}-1\right) = \beta^-, \tag{38}$$

$$a_{13} + a_{14} + a_{10}\left(\frac{\beta^-}{\beta^+}-1\right) = \beta^-, \tag{39}$$

$$a_{15} + a_{16}\frac{\beta^-}{\beta^+} = 0, \tag{40}$$

$$a_{17} + a_{18}\frac{\beta^-}{\beta^+} = 0, \tag{41}$$

$$a_{19} + a_{20} = 0. \tag{42}$$

If we can solve this linear system of equations to get γ_i's, then by collecting the remaining terms in (29), we can determine the correction term which is given by

$$\begin{aligned} C_{ijk} = {} & a_{10}\left(\frac{[f]}{\beta^+} - \frac{\lambda^+ w}{\beta^+} - w_{\eta\eta} - w_{\tau\tau}\right) + a_{12}w_{\eta\eta} + a_{14}w_{\tau\tau} + a_{16}\frac{v_\eta}{\beta^+} + a_{18}\frac{v_\tau}{\beta^+} + a_{20}w_{\eta\tau} + a_2 w \\ & + \frac{1}{\beta^+}\left\{a_4 + a_{10}\left(\chi_{\eta\eta}+\chi_{\tau\tau}-\frac{\beta^+_\xi}{\beta^+}\right) - a_{12}\chi_{\eta\eta} - a_{14}\chi_{\tau\tau} - a_{16}\frac{\beta^+_\eta}{\beta^+} - a_{18}\frac{\beta^+_\tau}{\beta^+} - a_{20}\chi_{\eta\tau}\right\} v \\ & + \left(a_6 - a_{10}\frac{\beta^+_\eta}{\beta^+} + a_{16}\chi_{\eta\eta} + a_{18}\chi_{\eta\tau}\right) w_\eta + \left(a_8 - a_{10}\frac{\beta^+_\tau}{\beta^+} + a_{16}\chi_{\eta\tau} + a_{18}\chi_{\tau\tau}\right) w_\tau. \end{aligned} \tag{43}$$

4.2. Computing the principal curvatures using the level set function

In order to determine the matrix entries of the linear system of equations (33)–(42) for the coefficients γ_i's, we need to compute the second-order tangential derivatives $\chi_{\eta\eta}$, $\chi_{\tau\tau}$, $\chi_{\eta\tau}$ of χ at $\mathbf{X}^*$. We call these quantities *principal curvatures*. These quantities are computed from the level set function. Since $\varphi(\chi(\eta,\tau),\eta,\tau)=0$, it follows from the implicit function theory that

$$\varphi_\eta + \varphi_\xi \chi_\eta = 0, \tag{44}$$

$$\varphi_\tau + \varphi_\xi \chi_\tau = 0 \tag{45}$$

and

$$\begin{aligned}
&\varphi_{\eta\eta} + \varphi_{\eta\xi}\chi_\eta + (\varphi_{\xi\eta} + \varphi_{\xi\xi}\chi_\eta)\chi_\eta + \varphi_\xi \chi_{\eta\eta} = 0,\\
&\varphi_{\eta\tau} + \varphi_{\eta\xi}\chi_\tau + (\varphi_{\xi\tau} + \varphi_{\xi\xi}\chi_\tau)\chi_\eta + \varphi_\xi \chi_{\eta\tau} = 0,\\
&\varphi_{\tau\tau} + \varphi_{\tau\xi}\chi_\tau + (\varphi_{\xi\tau} + \varphi_{\xi\xi}\chi_\tau)\chi_\tau + \varphi_\xi \chi_{\tau\tau} = 0.
\end{aligned}$$

So, we have

$$\begin{aligned}
\chi_{\eta\eta} &= -\varphi_{\eta\eta}/\varphi_\xi,\\
\chi_{\tau\tau} &= -\varphi_{\tau\tau}/\varphi_\xi,\\
\chi_{\eta\tau} &= -\varphi_{\eta\tau}/\varphi_\xi,
\end{aligned} \tag{46}$$

where $(\varphi_\xi, \varphi_\eta, \varphi_\tau)$ and $(\varphi_{\xi\xi}, \varphi_{\eta\eta}, \varphi_{\tau\tau})$ are given in (6) and (7), respectively. Our procedure for evaluating the principal curvatures includes:

- compute the first and second derivatives of φ at the surrounding *grid points* using the standard central difference scheme;
- use the bi-linear interpolation to compute the first- and second-order derivatives $\varphi_x, \varphi_y, \ldots, \varphi_{zz}$ in the original coordinates at the *projection point* $\mathbf{X}^*$;
- use the formulae (6) and (7) to get the first- and second-order derivatives $\varphi_\xi, \varphi_\eta, \varphi_\tau, \ldots, \varphi_{\tau\tau}$ in the local coordinate system;
- use the formula (46) to compute the principal curvatures $\chi_{\eta\eta}$, $\chi_{\eta\tau}$, and $\chi_{\tau\tau}$.

The bi-linear interpolation uses eight grid points. Given any point (x^*, y^*, z^*) on the interface, we can find a cube containing the point with the eight vertices (x_0, y_0, z_0), (x_0, y_0, z_1), (x_0, y_1, z_0), (x_0, y_1, z_1), (x_1, y_0, z_0), (x_1, y_0, z_1), (x_1, y_1, z_0), and (x_1, y_1, z_1). Let Q_{ijk} be the function values at the eight vertices. The eight point bi-linear interpolation is defined as

$$Q(x^*, y^*, z^*) = \frac{1}{8}\sum_{i,j,k=0}^{1} Q_{ijk}\bar{x}_i\bar{y}_j\bar{z}_k, \tag{47}$$

where

$$\begin{aligned}
&\bar{x}_i = 1 + (2i-1)\left(\frac{2(x-x_0)}{h} - 1\right), \quad \bar{y}_j = 1 + (2j-1)\left(\frac{2(y-y_0)}{h} - 1\right),\\
&\bar{z}_k = 1 + (2k-1)\left(\frac{2(z-z_0)}{h} - 1\right).
\end{aligned} \tag{48}$$

4.3. Computing the tangential derivatives of interface quantities

In the evaluation of the correction terms C_{ijk}, we need to compute the tangential derivatives such as w_η, w_τ, $w_{\eta\eta}$, $w_{\tau\tau}$, $w_{\eta\tau}$, v_η, and v_τ, where w and v are only defined along the interface Γ. They are computed using the least squares interpolation.

Let g be a function defined on the interface and therefore we know its values at the projections $\mathbf{X}_p^*$. We explain below how to compute its tangential derivatives at a particular projection $\mathbf{X}^*$ using the projections $\mathbf{X}_p^*$ in the neighborhood of $\mathbf{X}^*$.

In the neighborhood of $\mathbf{X}^*$, the interface quantity g can be written as $g(\eta,\tau)$ using the local coordinates. The least squares interpolation for g_η, g_τ, and $g_{\eta\eta}$ at $\mathbf{X}^*$, for example, can be written as

$$g_\eta(\mathbf{X}^*) = \sum_{|\mathbf{X}^*-\mathbf{X}_p^*|\leqslant R_\epsilon} \alpha_p g_p, \quad g_\tau(\mathbf{X}^*) = \sum_{|\mathbf{X}^*-\mathbf{X}_p^*|\leqslant R_\epsilon} \lambda_p g_p, \quad g_{\eta\eta}(\mathbf{X}^*) = \sum_{|\mathbf{X}^*-\mathbf{X}_p^*|\leqslant R_\epsilon} \sigma_p g_p, \tag{49}$$

where $g_p = g(\mathbf{X}_p^*)$ are the function values at the projections $\mathbf{X}_p^*$, R_ϵ is a pre-chosen parameter between $5.1h$ and $6.1h$. We should choose R_ϵ such that at least 10 points are involved. We explain how to compute the coefficients α_p for $g_\eta(\mathbf{X}^*)$ as an illustration. It is based on the Taylor expansion and the singular value decomposition (SVD) to solve an under-determined system of equations.

Using the Taylor expansion at $\mathbf{X}^*$, we have

$$\begin{aligned}\sum_{|\mathbf{X}_p^*-\mathbf{X}^*|\leqslant R_\epsilon} \alpha_p g_p &= \sum_{|\mathbf{X}_p^*-\mathbf{X}^*|\leqslant R_\epsilon} \alpha_p \left(g^* + \nabla g^* \cdot (\mathbf{X}_p^* - \mathbf{X}^*) + \frac{1}{2}(\mathbf{X}_p^* - \mathbf{X}^*)^{\mathrm{T}} H(g^*)(\mathbf{X}_p^* - \mathbf{X}^*) + \cdots \right) \\ &= (\quad) g^* + (\quad) g_\eta^* + (\quad) g_\tau^* + (\quad) g_{\eta\eta}^* + (\quad) g_{\eta\tau}^* + (\quad) g_{\tau\tau}^* + \text{h.o.t},\end{aligned}$$

where $H(g)$ is the Hessian matrix of g in terms of the variables η and τ under the local coordinates ξ–η–τ centered at the projection $\mathbf{X}^*$, h.o.t stands for the high-order terms of $|\mathbf{X}_p^* - \mathbf{X}^*|$, and the contents in the parentheses are the corresponding right-hand side in the system of equations below. Using the method of the under-determined coefficient, we set

$$\begin{aligned}&\sum_p \alpha_p = 0, \quad \sum_p \alpha_p \eta_p = 1, \quad \sum_p \alpha_p \tau_p = 0, \\ &\sum_p \alpha_p \eta_p^2 = 0, \quad \sum_p \alpha_p \tau_p^2 = 0, \quad \sum_p \alpha_p \eta_p \tau_p = 0,\end{aligned} \tag{50}$$

where (η_p, τ_p)'s are the coordinates of the projections $\mathbf{X}_p^*$ on the interface in the parametric form $\xi = \chi(\eta,\tau)$ centered at $\mathbf{X}^*$. The under-determined system is solved by the SVD subroutine form LAPACK/LINPACK which is available from the Netlib. The other tangential derivatives can be computed in the same way with different right-hand side. Since the coefficients matrix is the same, we just need to compute the SVD once.

4.4. An optimization approach

After the preparations from previous sections, we are ready to determine the coefficients γ_m's and n_s in the finite difference equation in (26) for all irregular grid points. It seems that we can take $n_s = 10$ because there are 10 Eqs. (33)–(42). The problem is that we cannot guarantee that the system (33)–(42) has a solution and the stability of the resulting system of the finite difference equations.

We propose the maximum principle preserving scheme by choosing $n_s > 10$ and adding the sign constraints

$$\begin{aligned} &\gamma_m < 0 \quad \text{if } (i_m, j_m, k_m) = (0,0,0), \\ &\gamma_m \geqslant 0 \quad \text{if } (i_m, j_m, k_m) \neq (0,0,0), \end{aligned} \tag{51}$$

in addition to the linear system of equations (33)–(42). The sign constraints will guarantee the coefficient matrix of the system of the finite difference equations be an M-matrix.

To solve the linear system of equations (33)–(42) with the inequality constraints (51), we construct a quadratic optimization problem

$$\min_{\gamma} \frac{1}{2} \sum_{m} (\gamma_m - d_m)^2, \tag{52}$$

s.t.

$$\begin{aligned} &B\gamma = \mathbf{b}, \quad \text{the system of (33)–(42)}, \\ &\gamma_m < 0 \quad \text{if } (i_m, j_m, k_m) = (0,0,0), \\ &\gamma_m \geqslant 0 \quad \text{if } (i_m, j_m, k_m) \neq (0,0,0), \end{aligned} \tag{53}$$

where $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_{n_s})^{\mathrm{T}}$ is the vector composed of the coefficients of the finite difference scheme and $B\gamma = \mathbf{b}$ denotes the equality constraints specified by (33)–(42). We also want to choose γ_m's in such a way that the finite difference scheme (26) reduces to the standard central finite difference scheme if there is no interface or the coefficient β of the PDE is continuous across the interface. This can be done by choosing

$$\begin{aligned} d_m &= \frac{\beta_{i+i_m/2, j+j_m/2, k+k_m/2}}{h^2} \quad \text{if } (i_m, j_m, k_m) \text{ is one of the six neighbors of } (0,0,0), \\ d_0 &= -\frac{1}{h^2} \sum_{m, m \neq 0} \beta_{i+i_m/2, j+j_m/2, k+k_m/2}, \\ d_m &= 0, \quad \text{otherwise}, \end{aligned} \tag{54}$$

where the summation is over the six neighbors of the grid point (x_i, y_j, z_k) and $\beta_{i-1/2,j,k} = \beta(x_i - h/2, y_j, z_k)$ and so forth.

There are several commercial and educational software packages that are designed to solve constrained quadratic optimization problems, such as **QP** in Matlab and **QL** by Schittkowski [16].

What the minimum n_s should be that can guarantee the existence of the solution to the optimization problem is still an open theoretical problem. In our implementation, we take all the grid points in the cube centered at (x_i, y_j, z_k), that is, $n_s = 27$. We have not experienced any numerical failure for our testing problems. In [11], the authors numerically showed the existence of the solution to the optimization problem in two space dimensions. We believe that the conclusions are also true in three dimensions.

If the optimization problem has a solution at all irregular grid points, then it is shown in [11] that the solution to the finite difference scheme has second-order accuracy globally in the infinity norm. We omit the proof here since it is essentially the same as in two space dimensions.

In case that the optimization solver fails to return a feasible solution at an irregular grid point, we can switch to a lower-order scheme such as the ghost fluid method [12] or the standard central finite difference scheme. Since such points are few, if there are any, the global accuracy will not be affected.

We summarize our algorithm for the elliptic interface problem with variable but discontinuous coefficient below.

- Set-up a Cartesian grid.
- Label the grid points as regular, irregular.
- Find the projections of irregular grid points on the interface as described in Section 3.2.

- At each irregular grid point, calculate the local coordinates (ξ_m, η_m, τ_m) for 27 neighboring grid points. Calculate the matrix B and vectors $\mathbf{b}$ from the system (33)–(42) and d_i from (54). Solve the quadratic programming (52) and (53) for γ_m's and then compute the correction term C_{ijk} using (43).
- Use the formula (27) as the finite difference scheme at regular grid points.
- Form the system of the finite difference equations (26).
- Solve the system of the finite difference equations (26) to get an approximate solution to the PDE.

4.5. An algebraic multigrid solver

If the coefficient β not only has a jump across the interface, but also is a function of location (x, y, z), there are almost no fast elliptic solvers that can be used to solve the linear system of equations obtained from our maximum principle preserving scheme. The Gauss–Seidel or SOR method is too slow in convergence. We use the algebraic multigrid method (AMG) developed by the German National Research Center for Information Technology (GMD) which is available on the Internet, see also [15]. The AMG has been shown to be a robust and efficient solver for a linear system of equations $Qu = F$ with certain properties. The AMG is guaranteed to converge if the coefficient matrix Q satisfies one of the following conditions:

- Q is positive/negative definite or semi-positive/negative definite with ROWSUM $= 0$ for each row, where ROWSUM denotes the sum of the entries in each row.
- Q is "essentially" positive type, i.e.,
  - The diagonal entries of Q must be positive/negative.
  - Most of the off-diagonal entries of Q are non-positive/non-negative.
  - For each row, the ROWSUM should be non-negative/non-positive.

The linear system of equations derived from the maximum principle preserving scheme is "essentially" negative definite matrix (an M-matrix) and the algebraic multigrid solver can be applied. Our numerical experiments showed that the AMG method generally converged faster than the SOR method. The speed-up increases as the number of grid lines gets larger.

4.6. Numerical results for the maximum principle preserving scheme

We have done a number of numerical experiments which confirm second-order accuracy of the maximum principle preserving scheme. The numerical tests are done using Sun Ultra 10 workstations or the CRAY T916 supercomputer at the North Carolina Supercomputing Center (NCSC). The computational domain is $[-1, 1] \times [-1, 1] \times [-1, 1]$. In all examples, $L = M = N$, and $n_s = 27$ (i.e., all 27 grid points involved in the usual 27 point stencil) unless otherwise specified. The convergence tolerance for the algebraic multigrid method is 10^{-5} for the test results presented here.[1]

Example 4.1. We present an example with a variable and discontinuous coefficient β. The interface is a sphere $x^2 + y^2 + z^2 = 1/4$. The differential equation is

$$(\beta u_x)_x + (\beta u_y)_y + (\beta u_z)_z = f,$$

[1] We have also tried smaller tolerance 10^{-8}. The accuracy and the order of convergence remain almost the same except that the CPU time for the linear solver increases slightly. With smaller tolerance, the computed solution gives better approximation to the linear system of the finite difference equations, but not necessarily a better approximation to the original partial differential equation because the local truncation errors are generally larger than the tolerance. We believe that 10^{-5} is a reasonable choice of the tolerance for our test problems.

where

$$\beta(x,y,z) = \begin{cases} r^2+1 & \text{if } r < \frac{1}{2}, \\ b & \text{if } r \geqslant \frac{1}{2}, \end{cases}$$

$$f(x,y,z) = 10r^2 + 6,$$

and b is a constant and $r = \sqrt{x^2+y^2+z^2}$. The Dirichlet boundary condition is determined from the exact solution

$$u(x,y,z) = \begin{cases} r^2 & \text{if } r < \frac{1}{2}, \\ \left(\frac{r^4}{2} + r^2\right)\Big/ b - \left(\frac{r_0^4}{2} + r_0^2\right)\Big/ b + r_0^2 & \text{if } r \geqslant \frac{1}{2}, \end{cases} \tag{55}$$

where $r_0 = 1/2$.

It can be calculated that $[u] = 0$ and $[\beta u_n] = 0$ in this example. However, the normal derivative u_n is discontinuous due to the discontinuity in the coefficient β.

The jump in the coefficient β depends on the choice of the constant b. We tested three different cases, $b = 1$, $b = 10$ (small jump), and $b = 1000$ (large jump). Table 1 shows the results of the grid refinement analysis. The maximum relative error is defined as

$$\|E_N\|_\infty = \frac{\max_{i,j,k} |u(x_i,y_j,z_k) - u_{ijk}|}{\max_{i,j,k} |u(x_i,y_j,z_k)|}, \tag{56}$$

where u_{ijk} is the computed approximation to the exact one $u(x_i,y_j,z_k)$. We also display the ratio of two successive errors

$$\text{ratio} = \frac{\|E_N\|_\infty}{\|E_{2N}\|_\infty}. \tag{57}$$

In Table 1, we see that the *average ratio* approaches number 4 indicating second-order accuracy of the maximum principle preserving scheme.

Example 4.2. Now we present an example of multi-connected domain and discontinuous solution. The level set function is

$$\varphi(x,y,z) = S_1(x,y,z)S_2(x,y,z),$$

where

$$S_1(x,y,z) = (x-0.2)^2 + 2(y-0.2)^2 + z^2 - 0.01,$$
$$S_2(x,y,z) = 3(x+0.2)^2 + (y+0.2)^2 + z^2 - 0.01.$$

Table 1
The grid refinement analysis for Example 4.1

$L \times M \times N$	$b = 1$		$b = 10$		$b = 1000$	
	$\|E_N\|_\infty$	Ratio	$\|E_N\|_\infty$	Ratio	$\|E_N\|_\infty$	Ratio
$26 \times 26 \times 26$	1.247×10^{-3}		1.525×10^{-3}		3.485×10^{-3}	
$52 \times 52 \times 52$	3.979×10^{-4}	3.134	5.240×10^{-4}	2.910	1.111×10^{-3}	3.137
$104 \times 104 \times 104$	9.592×10^{-5}	4.148	1.010×10^{-4}	5.188	1.605×10^{-4}	6.922

The interface is the zero level set of $\varphi(x,y,z)$ that satisfies $\varphi(x,y,z)=0$. The coefficient β is piecewise constant β^+ in Ω^+ and β^- in Ω^-. The source term is

$$f(x,y,z)=\begin{cases} 6\beta^- e^{x+2y+z} & \text{in } \Omega^-, \\ -\beta^+(4\pi^2\sin 2\pi x+\pi^2\sin\pi y+16\pi^2\sin 4\pi z) & \text{in } \Omega^+. \end{cases}$$

The Dirichlet boundary condition is determined from the exact solution

$$u(x,y,z)=\begin{cases} e^{x+2y+z} & \text{in } \Omega^-, \\ \sin 2\pi x+\sin\pi y+\sin 4\pi z & \text{in } \Omega^+. \end{cases} \tag{58}$$

The jump conditions are

$$[u]=\sin 2\pi x+\sin\pi y+\sin 4\pi z-e^{x+2y+z},$$

and

$$\begin{pmatrix}[\beta u_\xi]\\ [\beta u_\eta]\\ [\beta u_\tau]\end{pmatrix}=A\begin{pmatrix}2\pi\beta^+\cos 2\pi x-\beta^- e^{x+2y+z}\\ \pi\beta^+\cos\pi y-2\beta^- e^{x+2y+z}\\ 4\pi\beta^+\cos 4\pi z-\beta^- e^{x+2y+z}\end{pmatrix},$$

and A is the local coordinates transformation matrix defined in (4).

We tested two different cases, $\beta^-=\beta^+=1$, and $\beta^-=1,\ \beta^+=1000$. Fig. 2 shows a slice of the computed solution for the bigger jump case and Table 2 shows the results of the grid refinement analysis. Again second-order accuracy is observed.

In Table 3, we show the comparison of the CPU time (in seconds) of the SOR method and the algebraic multigrid solver for the maximum principle preserving scheme on Sun Ultra 10. We see the algebraic multigrid solver is much faster than the SOR method when L, M, and N are large. For small problems, the algebraic multigrid solver may be slower due to the set-up time in the algebraic multigrid solver. More examples can be found in [5].

A natural concern about the maximum principle preserving scheme is the computational cost in dealing with the interface, or irregular grid points. The cost includes indexing the grid points, finding the or-

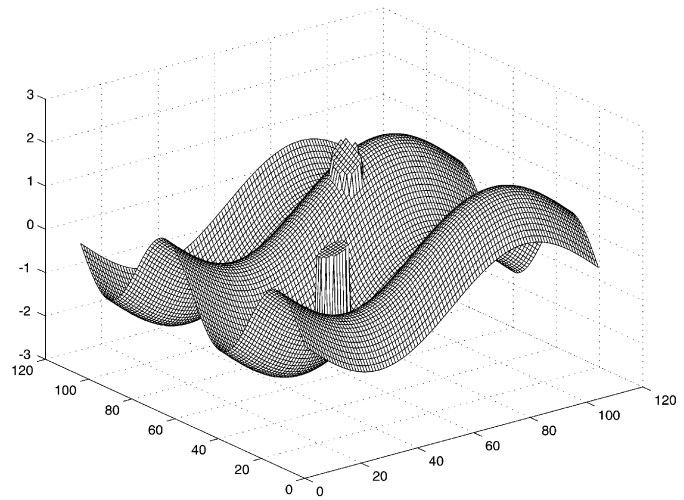

Fig. 2. A slice of the computed solution $u(x,y,0)$ for Example 4.2. The parameters are: $\beta^+=1000,\ \beta^-=1$ and $L=M=N=104$.

Table 2
The grid refinement analysis for Example 4.2

$L \times M \times N$	$\beta^+ = 1,\ \beta^- = 1$		$\beta^+ = 1000,\ \beta^- = 1$	
	$\|E_N\|_\infty$	Ratio	$\|E_N\|_\infty$	Ratio
$52 \times 52 \times 52$	3.108×10^{-2}		2.032×10^{-2}	
$104 \times 104 \times 104$	6.758×10^{-3}	4.599	4.771×10^{-3}	4.259

Table 3
Comparison of CPU time (in seconds) of the SOR method and the AMG method on Sun Ultra 10

$L \times M \times N$	Example 4.1 ($b = 10^7$)		Example 4.2 ($\beta^+ = 10^4,\ \beta^- = 1$)	
	SOR	AMG	SOR	AMG
$20 \times 20 \times 20$	0.21	1.57	0.06	0.83
$40 \times 40 \times 40$	27.51	25.56	29.62	13.89
$80 \times 80 \times 80$	1410.54	265.26	1464.57	265.84

thogonal projections, and solving the quadratic optimization problem at each irregular grid point. Note that the extra time is always needed to deal with the interface no matter what a numerical method is used to solve the interface problem.

Since the number of irregular grid points is one-order fewer than the number of regular grid points, the computational cost in dealing with the irregular grid points is negligible if the spatial step size h is small enough. In practice, for three-dimensional problems, the spatial step size h will be limited. The CPU time spent on interface depends on the complexity of the interface, the linear solver used, and the number of grid points n_s in the finite difference stencil. In Table 4, we show the percentage of the CPU time spent on the interface with $n_s = 27$. In Example 4.1, there are more irregular grid points than that in Example 4.2, so the percentage of CPU time spent on the interface is larger. In either case, the percentage of CPU time decreases significantly as we decrease the spatial step size h.

4.7. A special case

When β and λ are continuous but the solution and/or the normal derivative have jumps across the interface, the maximum principle preserving method becomes the standard central finite difference scheme. Furthermore if β is a constant, we have the following theorem.

Table 4
CPU percentage distribution for Examples 4.1 and 4.2

$L \times M \times N$	Example 4.1 ($b = 1000$)				Example 4.2 ($\beta^+ = 1000,\ \beta^- = 1$)			
	n_{irreg}	T_{pre}	T_{total}	Percentage	n_{irreg}	T_{pre}	T_{total}	Percentage
$26 \times 26 \times 26$	920	23.65	31.95	74.02	58	3.53	10.44	33.81
$52 \times 52 \times 52$	3528	92.50	172.79	53.53	218	5.95	95.53	6.23
$104 \times 104 \times 104$	14048	380.22	2196.31	17.31	870	36.74	1430.14	2.57

n_{irreg} is the total number of irregular grid points, T_{pre} is the CPU time spent in dealing with irregular grid points, and T_{total} is the total CPU time. The time unit is in second.

Theorem 4.1. *If $\beta(x,y,z)$ is a constant β and λ is continuous, then the solution of Eqs.* (33)–(42) *are*

$$\gamma_{i-1,jk} = \gamma_{i+1,jk} = \gamma_{i,j-1,k} = \gamma_{i,j+1,k} = \gamma_{ij,k-1} = \gamma_{ij,k+1} = \frac{\beta}{h^2}, \quad \gamma_{ijk} = -\frac{6\beta}{h^2}, \tag{59}$$

and all other coefficients $\gamma_{i+i_m,j+j_m,k+k_m} = 0$. The solution above is also a solution to the constrained optimization problem.

Proof. We only need to verify that these γ_{ijk}'s satisfy the linear system of equations (33)–(42) at any irregular grid point. Without loss of generality, we assume the irregular grid point (x_i, y_j, z_k) be the origin. The continuity condition of λ means $[\lambda] = 0$, and a constant β means $[\beta] = 0$. We also have $\rho = \beta^-/\beta^+ = 1$, $\beta_\xi = \beta_\eta = \beta_\tau = 0$, etc. Therefore, the linear system of equations (33)–(42) becomes

$$\begin{aligned}
&a_1 + a_2 = 0, \quad a_3 + a_4 = 0,\\
&a_5 + a_6 = 0, \quad a_7 + a_8 = 0,\\
&a_9 + a_{10} = \beta, \quad a_{11} + a_{12} = \beta,\\
&a_{13} + a_{14} = \beta, \quad a_{15} + a_{16} = 0,\\
&a_{17} + a_{18} = 0, \quad a_{19} + a_{20} = 0.
\end{aligned}$$

The first equation

$$a_1 + a_2 = 0, \quad \text{i.e.,} \quad \sum_m \gamma_m = 0$$

is obviously true. Under the transformation (3), let the new coordinates corresponding to $(0,0,0)$, $(-h,0,0)$, $(h,0,0)$, $(0,-h,0)$, $(0,h,0)$, $(0,0,-h)$, and $(0,0,h)$ be (ξ_m, η_m, τ_m), $m = 1, \ldots, 7$. Define

$$\mathbf{Y}_m = \begin{pmatrix} \xi_m \\ \eta_m \\ \tau_m \end{pmatrix}, \quad m = 1, 2, \ldots, 7,$$

and

$$\mathbf{X}_1 = \begin{pmatrix} -x^* \\ -y^* \\ -z^* \end{pmatrix}, \quad \mathbf{X}_2 = \begin{pmatrix} -h - x^* \\ -y^* \\ -z^* \end{pmatrix}, \quad \mathbf{X}_3 = \begin{pmatrix} h - x^* \\ -y^* \\ -z^* \end{pmatrix}, \quad \mathbf{X}_4 = \begin{pmatrix} -x^* \\ -h - y^* \\ -z^* \end{pmatrix},$$

$$\mathbf{X}_5 = \begin{pmatrix} -x^* \\ h - y^* \\ -z^* \end{pmatrix}, \quad \mathbf{X}_6 = \begin{pmatrix} -x^* \\ -y^* \\ -h - z^* \end{pmatrix}, \quad \mathbf{X}_7 = \begin{pmatrix} -x^* \\ -y^* \\ h - z^* \end{pmatrix}.$$

We can verify

$$\mathbf{Y}_m = A\mathbf{X}_m, \quad m = 1, 2, \ldots, 7,$$

and

$$\begin{aligned}
\begin{pmatrix} a_3 + a_4 \\ a_5 + a_6 \\ a_7 + a_8 \end{pmatrix} &= \begin{pmatrix} \sum_m \gamma_m \xi_m \\ \sum_m \gamma_m \eta_m \\ \sum_m \gamma_m \tau_m \end{pmatrix} = \sum_m \gamma_m \mathbf{Y}_m = \sum_m \gamma_m A \mathbf{X}_m \\
&= \frac{\beta}{h^2} A \begin{pmatrix} 6x^* - h - x^* + h - x^* - x^* - x^* - x^* - x^* \\ 6y^* - y^* - y^* - h - y^* + h - y^* - y^* - y^* \\ 6z^* - z^* - z^* - z^* - z^* - h - z^* + h - z^* \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.
\end{aligned}$$

Therefore, the second to the fourth equations (34)–(36) are also satisfied. To prove the rest, we can easily verify

$$\sum_m \gamma_m \mathbf{X}_m \mathbf{X}_m^{\mathrm{T}} = \begin{pmatrix} 2\beta & 0 & 0 \\ 0 & 2\beta & 0 \\ 0 & 0 & 2\beta \end{pmatrix} = 2\beta I.$$

Therefore, we have

$$\sum_m \frac{\gamma_m}{2} \mathbf{Y}_m \mathbf{Y}_m^{\mathrm{T}} = \frac{1}{2} \sum_m \gamma_m A \mathbf{X}_m \mathbf{X}_m^{\mathrm{T}} A^{\mathrm{T}} = \beta I,$$

since $AA^{\mathrm{T}} = I$, where A is defined in (4). Furthermore, we have the following:

$$\sum_m \frac{\gamma_m}{2} \mathbf{Y}_m \mathbf{Y}_m^{\mathrm{T}} = \begin{pmatrix} \sum_m \frac{\gamma_m}{2} \xi_m^2 & \sum_m \frac{\gamma_m}{2} \xi_m \eta_m & \sum_m \frac{\gamma_m}{2} \xi_m \tau_m \\ \sum_m \frac{\gamma_m}{2} \xi_m \eta_m & \sum_m \frac{\gamma_m}{2} \eta_m^2 & \sum_m \frac{\gamma_m}{2} \eta_m \tau_m \\ \sum_m \frac{\gamma_m}{2} \xi_m \tau_m & \sum_m \frac{\gamma_m}{2} \eta_m \tau_m & \sum_m \frac{\gamma_m}{2} \tau_m^2 \end{pmatrix},$$

which implies that

$$a_9 + a_{10} = a_{11} + a_{12} = a_{13} + a_{14} = \beta,$$
$$a_{15} + a_{16} = a_{17} + a_{18} = a_{19} + a_{20} = 0.$$

This concludes the proof. □

The result above tells us that for Poisson equations, the standard central finite difference scheme can be applied directly with the right-hand side being modified by C_{ijk} even if the solution and/or the normal derivative have jumps. Fast Poisson solvers such as the one from the Fishpack [1] can be used to solve the resulting linear system.

5. A fast Poisson solver for piecewise constant coefficient

In this section, we discuss a fast algorithm for solving the Poisson equation (1), (2) when β is piecewise constant in the domain Ω and $\lambda \equiv 0$. Divided by the coefficient in each sub-domain of Ω, the original problem can be written as

$$\begin{aligned} \Delta u &= \frac{f}{\beta^+} \quad \text{if } (x,y,z) \in \Omega^+, \\ \Delta u &= \frac{f}{\beta^-} \quad \text{if } (x,y,z) \in \Omega^-, \end{aligned} \tag{60a}$$

$$[u] = w, \quad [\beta u_n] = v, \tag{60b}$$

$$\text{Given BC} \quad \text{on } \partial\Omega. \tag{60c}$$

The Poisson equation is only valid in the interior of the domain excluding the interface Γ. The Poisson equation can be solved readily with a fast 3D Poisson solver if we know the jump in the solution $[u] = w$ and the jump in the normal derivative $[u_n]$. This is because the system of the finite difference equations (26) for u_{ijk} reduces to the standard seven point discrete Poisson equation with modified right-hand side, see Section 4.7.

However, the second jump condition for the original problem (1) is in the flux $[\beta u_n] = v$ instead of $[u_n]$. We cannot divide β from the flux jump condition because β is discontinuous. In [10], a fast method for two-dimensional problems is proposed. In this section, we describe our algorithm for three-dimensional problems.

As described in [10], the idea is to augment the unknown $[u_n] = g$ and equation $[\beta u_n] = v$ to (60a). That is, we determine an interface function $g(s_1, s_2)$ in such a way that the solution $u(g)$ to (60a) satisfies the jump condition $[\beta u_n(g)] = v$. Since $[u_n]$ is only defined along the interface, it is one-dimensional lower than the dimension of the solution u. We apply the GMRES method to solve the unknown jump g by eliminating u from the augmented system. Numerically, we represent the unknown jump g only at certain projections $\mathbf{X}_c^*$ of the irregular grid points from a particular side of the interface, for example, the side where $\varphi \geq 0$ to avoid possibly clustered points. We call these projections $\mathbf{X}_c^*$ *control points* where we will find the unknown jump $[u_n]$ numerically.

5.1. Setting-up the system of equations for $[u_n]$ and computing the residual

We select the projections from the $\varphi \geqslant 0$ side as a set of control points $\mathbf{X}_1^*, \mathbf{X}_2^*, \ldots, \mathbf{X}_{N_{\text{contr}}}^*$, and the jump in the normal derivative $[u_n]$ as $G_1, G_2, \ldots, G_{N_{\text{contr}}}$ at the control points $\mathbf{X}_c^*$. Denote the resulting discrete linear system of equations for $G = [G_1, G_2, \ldots, G_{N_{\text{contr}}}]^{\mathrm{T}}$ as

$$SG = \bar{\mathbf{b}}, \tag{61}$$

where S is an N_{contr} by N_{contr} matrix. The matrix–vector form above is the Schur complement system of the augmented system

$$\begin{bmatrix} A & B \\ E & D \end{bmatrix} \begin{bmatrix} U \\ G \end{bmatrix} = \begin{bmatrix} \bar{F}_1 \\ \bar{F}_2 \end{bmatrix}, \tag{62}$$

where the system of the first row in the block matrix is the system of finite difference equations of the Poisson equation given $[u] = w$ and $[u_n] = G$ defined at the control points, while the second row is the flux jump condition $[\beta u_n(g)] = v$ in the discrete form. In our implementation, the Schur complement $S = D - EA^{-1}B$ and other matrices are never formed explicitly. Below, we outline our method to compute the residual vector $R(G) = SG - \bar{\mathbf{b}}$:

- *Step 1:* For a given vector G defined at control points $\mathbf{X}_c^*$, we use the least squares interpolation to get the intermediate jump g of the normal derivative and their first-order derivatives along the interface at all projections $\mathbf{X}_p^*$. The scheme is discussed in the next section.
- *Step 2:* Solve the Poisson equation for $u_{ijk}(G)$ with given $[u] = w$ and the interpolated $[u_n] = g$. This step is done using a fast Poisson solver since only the right-hand side of (60a) needs to be modified by the correction term C_{ijk} determined from (43). The main computational cost in this step includes the time to determine the correction terms and solve the Poisson equation.
- *Step 3:* Compute the residual vector

$$R(G) = \beta^+ u_n^+(g) - \beta^- u_n^-(g) - v = SG - \bar{\mathbf{b}}, \tag{63}$$

  which is simply the equation of the flux jump condition at the control points. The normal derivatives $u_n^+(g)$ and $u_n^-(g)$ at the control points $\mathbf{X}_c^*$ are computed by the least squares interpolation which will be explained in Section 5.3.

Note that when we take $[u] = w$ and $[u_n] = 0$, we have $R(\mathbf{0}) = -\bar{\mathbf{b}}$, the right-hand side of the Schur complement. We apply the GMRES method to solve $R(G) = \bar{\mathbf{b}}$ with initial guess $g^0(\mathbf{X}_c^*) = v(\mathbf{X}_c^*)$. When the convergence criteria is met, we not only have the jump in the normal derivatives of the both sides at the control points, but also the solution to the original PDE (1).

Below we outline the entire fast algorithm for 3D elliptic interface problems with piecewise constant coefficient. Some implementation details can either be found in previous sections or will be explained further later in this section.

Outline of the algorithm for Poisson equations with piecewise constant coefficient:

- Set-up a Cartesian grid.
- Label the grid points as regular, irregular.
- Find the projections for irregular grid points.
- Select a set of control points, for example, we choose the projection points from a particular side of the interface as the control points.
- Let $[u] = w$, $[u_n] = 0$. Compute the residual of the Schur complement to get the right-hand side $\bar{\mathbf{b}}$.
- Set $G^0 = v$, call the GMRES method to solve the Schur complement. Once the convergence criteria is met, the method returns an approximate solution $U(G^{k_{final}})$, the normal derivative u_n^+ and u_n^- corresponding to the final step k_{final}.

5.2. Computing the surface quantities of the jump $[u_n]$ defined only at control points

In Section 4.3, we have discussed the least squares interpolation scheme to compute the surface derivatives of an interface quantity, for example, w_η, w_τ from w, the jump in the solution. The surface derivatives are needed in computing the correction terms for the Poisson equation (60a). However, the intermediate unknown vector $G = [u_n]$ is only defined at the control points $\mathbf{X}_c^*$, which are the projections of a particular side of the interface, say $\varphi \geqslant 0$ in our choice. The least squares interpolation scheme needs to be modified to use only the information from the control points but not all the projections. Given G that are defined at the control points, the interpolation scheme for g, g_η, g_τ at any projection $\mathbf{X}^*$ are

$$g_\eta(\mathbf{X}^*) = \sum_{|\mathbf{X}^* - \mathbf{X}_c^*| \leqslant R_\epsilon} \bar{\alpha}_c G_c, \quad g_\tau(\mathbf{X}^*) = \sum_{|\mathbf{X}^* - \mathbf{X}_c^*| \leqslant R_\epsilon} \bar{\lambda}_c G_c, \quad g(\mathbf{X}^*) = \sum_{|\mathbf{X}^* - \mathbf{X}_c^*| \leqslant R_\epsilon} \bar{\sigma}_c G_c, \tag{64}$$

where G_c are the given values at the control points $\mathbf{X}_c^*$. The procedure to determine the coefficients then is the same as described in Section 4.3.

5.3. Computing the normal derivatives of the solution u_{ijk} at projections

In the GMRES method, given a guess G, we need to carry out the matrix–vector multiplication. As stated before, this comprises three steps: (1) extend G to all projections of irregular grid points to get $g(\mathbf{X}_p^*)$; (2) solve the Poisson equation (60a) with $[u] = w$ and $[u_n] = g$ to get $u(g)$; (3) compute the residual $R(G) = \beta^+ u_n^+(g) - \beta^- u_n^-(g) - v$ at the control points. We have explained how to extend G and how to solve the Poisson equation. We now explain how to calculate $u_n^\pm$ at the control points based on the solution u_{ijk}. The algorithm is based on the least squares interpolation and the given jump condition $u_n^+ - u_n^- = g$. We explain the idea for computing u_n^- at a particular projection $\mathbf{X}_c^*$. Let

$$u_n^-(\mathbf{X}_c^*) \approx \sum_{(i,j,k) \in \mathcal{N}} \gamma_{ijk}\, u_{ijk} - C(\mathbf{X}_c^*), \tag{65}$$

where $\mathcal{N}$ denotes a set of the closest 50 grid points to the projection $\mathbf{X}_c^*$ in the sphere $|\mathbf{x}_{ijk} - \mathbf{X}_c^*| \leqslant R_\epsilon$, and $C(\mathbf{X}_c^*)$ is a correction term which can be determined once γ_{ijk}'s are computed. Note that the coefficients γ_{ijk}'s now have different meaning as the coefficients of the finite difference scheme that we used earlier. In our numerical tests, we take $R_\epsilon = 6.1h$. The interpolation (65) is robust and depends on u_{ijk} continuously. Using

the same idea presented in Section 4, we expand the true solution $u(\mathbf{x}_{ijk})$ at $\mathbf{X}_c^*$ from different sides of the interface and then express the quantities from the positive side in terms of those from the negative side. Have done this and made use of the formulae (19) and (23), we get, after collecting terms:

$$\begin{aligned} u_n^-(\mathbf{X}_c^*) = {} & (a_1+a_2)u^- + (a_3+a_4)u_\xi^- + (a_5+a_6)u_\eta^- + (a_7+a_8)u_\tau^- + (a_9+a_{10})u_{\xi\xi}^- + (a_{11}+a_{12})u_{\eta\eta}^- \\ & + (a_{13}+a_{14})u_{\tau\tau}^- + (a_{15}+a_{16})u_{\xi\eta}^- + (a_{17}+a_{18})u_{\xi\tau}^- + (a_{19}+a_{20})u_{\eta\tau}^- + a_2[u] + a_4[u_\xi] \\ & + a_6[u_\eta] + a_8[u_\tau] + a_{10}[u_{\xi\xi}] + a_{12}[u_{\eta\eta}] + a_{14}[u_{\tau\tau}] + a_{16}[u_{\xi\eta}] + a_{18}[u_{\xi\tau}] + a_{20}[u_{\eta\tau}] - C(\mathbf{X}_c^*) \\ & + \mathrm{O}(h^3 \max|\gamma_{ijk}|). \end{aligned} \tag{66}$$

where the variables a_i's are defined in (31). Thus we determine γ_{ijk}'s by setting

$$a_3 + a_4 = 1, \quad a_{2i-1} + a_{2i} = 0, \quad i = 1, 3, 4, \ldots, 10. \tag{67}$$

Again, the system is under-determined and in general, there are infinite number of solutions. We use the SVD subroutine from LAPACK/LINPACK to solve the system. Once the coefficients γ_{ijk}'s are determined, the correction term $C(\mathbf{X}_c^*)$ is then

$$\begin{aligned} C(\mathbf{X}_c^*) = {} & a_2[u] + a_4[u_\xi] + a_6[u_\eta] + a_8[u_\tau] + a_{10}[u_{\xi\xi}] + a_{12}[u_{\eta\eta}] + a_{14}[u_{\tau\tau}] \\ & + a_{16}[u_{\xi\eta}] + a_{18}[u_{\xi\tau}] + a_{20}[u_{\eta\tau}], \end{aligned} \tag{68}$$

in continuous case. Computationally, it is

$$\begin{aligned} C(\mathbf{X}_c^*) = {} & a_2 w + a_4 g + a_6 w_\eta + a_8 w_\tau + a_{10}\left(g(\chi_{\eta\eta} + \chi_{\tau\tau}) + \left[\frac{f}{\beta}\right] - w_{\eta\eta} - w_{\tau\tau}\right) + a_{12}(w_{\eta\eta} - g\chi_{\eta\eta}) \\ & + a_{14}(w_{\tau\tau} - g\chi_{\tau\tau}) + a_{16}(w_\eta\chi_{\eta\eta} + w_\tau\chi_{\eta\tau} + g_\eta) + a_{18}(w_\eta\chi_{\eta\tau} + w_\tau\chi_{\tau\tau} + g_\tau) + a_{20}(w_{\eta\tau} - g\chi_{\eta\tau}). \end{aligned} \tag{69}$$

The same procedure can be used to compute $u_n^+(\mathbf{X}_c^*)$. The interpolation scheme with under-determined system and the use of the SVD provide a stable and robust interpolation scheme with smooth error distributions.

5.4. The pre-conditioning strategy

Since the flux jump condition involves the normal derivative, some pre-conditioning techniques are crucial to reduce the number of iterations. The pre-conditioning technique that we have implemented is as follows. We use the method described in the previous section to compute one of $u_n^-(\mathbf{X}_\mathbf{c}^*)$ or $u_n^+(\mathbf{X}_\mathbf{c}^*)$, and we use equations

$$u_n^+(\mathbf{X}_c^*) - u_n^-(\mathbf{X}_c^*) = G(\mathbf{X}_c^*),$$

$$\beta^+ u_n^+(\mathbf{X}_c^*) - \beta^- u_n^-(\mathbf{X}_c^*) = v(\mathbf{X}_c^*)$$

to determine the other. Again, $G(\mathbf{X}_c^*)$ is the intermediate jump of the normal derivative at a control point. The formulas are

$$\text{If } \beta^+ < \beta^- : \quad u_n^-(\mathbf{X}_c^*) = \frac{v(\mathbf{X}_c^*) - \beta^+ G(\mathbf{X}_c^*)}{\beta^+ - \beta^-},$$

$$\text{If } \beta^+ > \beta^- : \quad u_n^+(\mathbf{X}_c^*) = \frac{v(\mathbf{X}_c^*) - \beta^- G(\mathbf{X}_c^*)}{\beta^+ - \beta^-}.$$

Consequently, we have the pre-conditioned equation

$$\text{If } \beta^+ < \beta^- : \quad \frac{\beta^-(\beta^+ G(\mathbf{X}_c^*) - v(\mathbf{X}_c^*))}{\beta^+ - \beta^-} + \beta^+ u_n^+(\mathbf{X}_c^*) - v(\mathbf{X}_c^*) = 0,$$

$$\text{If } \beta^+ > \beta^- : \quad \frac{\beta^+(v(\mathbf{X}_c^*) - \beta^- G(\mathbf{X}_c^*))}{\beta^+ - \beta^-} - \beta^- u_n^-(\mathbf{X}_c^*) - v(\mathbf{X}_c^*) = 0.$$

In this way we obtain the better conditioned system with the matrix form $I + K$, where K is a discretization of the Neumann-to-Neumann map for the Poisson equation.

5.5. An application to Helmholtz/Poisson equations on irregular domains

The idea of the fast interface Poisson solver described in the previous section can be used with a little modifications to solve three-dimensional Helmholtz/Poisson equations

$$\begin{aligned} &u_{xx} + u_{yy} + u_{zz} + \lambda u = f, \quad (x,y,z) \in \Omega, \\ &q(u,u_n) = 0, \quad (x,y,z) \in \partial\Omega \end{aligned} \tag{70}$$

defined on an irregular domain Ω (interior or exterior[2]), where $q(u,u_n)$ is a prescribed boundary condition which is a linear function of u and u_n along the boundary $\partial\Omega$. We will demonstrate the idea for interior problems.

We embed Ω into a cube R and extend the definition of the PDE and source term to the entire cube R

$$\Delta u + \lambda u = \begin{cases} f & \text{if } (x,y,z) \in \Omega, \\ 0 & \text{if } (x,y,z) \in R - \Omega, \end{cases} \quad \begin{array}{ll} [u] = g & \text{on } \partial\Omega, \\ [u_n] = 0 & \text{on } \partial\Omega, \\ u = 0 & \text{on } \partial R, \end{array} \quad \text{or} \quad \begin{array}{ll} [u] = 0 & \text{on } \partial\Omega, \\ [u_n] = g & \text{on } \partial\Omega, \\ u = 0 & \text{on } \partial R. \end{array} \tag{71}$$

Again, the solution u is a linear functional of g. We determine $g(s_1,s_2)$ such that the solution $u(g)$ satisfies the boundary condition $q(u(g),u_n(g)) = 0$. This can be solved using the GMRES iteration exactly as we discussed in Section 5. The only difference is the way in computing the residual vector.

5.6. Numerical examples of piecewise constant coefficient and irregular domains

All the simulations in this section are done on Sun Ultra 10 workstations. First we show an example of the interface problem with piecewise constant coefficient.

Example 5.1. The interface is a sphere $x^2 + y^2 + z^2 = 1/4$. The source term is

$$f(x,y,z) = \begin{cases} 6 & \text{if } r < \frac{1}{2}, \\ 20r^2 + \frac{1}{r^2} & \text{if } r \geqslant \frac{1}{2}. \end{cases} \tag{72}$$

Dirichlet boundary conditions and the jump conditions (2) are determined from the exact solution

$$u(x,y,z) = \begin{cases} \frac{r^2}{\beta^-} & \text{if } r < \frac{1}{2}, \\ \frac{r^4 + \log(2r)}{\beta^+} + \frac{(\frac{1}{2})^2}{\beta^-} - \frac{(\frac{1}{2})^4}{\beta^+} & \text{if } r \geqslant \frac{1}{2}. \end{cases} \tag{73}$$

[2] For exterior problems, we assume that the domain is a cube with holes.

We have $[u] = 0$ and $[\beta u_n] = 3/2$, Note that the solution is continuous in this example, but the normal derivative is not.

We tested three different cases, no jump, small jump, and large jump in β. Table 5 shows the results of the grid refinement analysis. An average ratio of 4 confirms the second-order accuracy. In Table 6, we listed the CPU time in seconds for the three cases on a Sun Ultra 10 computer. The second column N_{irreg} in the table is the number of total irregular grid points; the second column N_{contr} is the number of control points; the fifth, seventh, and ninth columns are the number of iterations of the GMRES method, which is also the number of calls to the 3D fast Poisson solvers. We see the numbers are almost independent of the mesh sizes.

We now show two examples in solving Poisson equations on interior and exterior irregular domains, respectively. The problems are not the interface problems and cannot be solved using the maximum principle preserving scheme. More examples can be found in [5].

Example 5.2. In this example, the domain is the exterior of the ellipsoid $x^2 + 2y^2 + z^2 = 1/4$. The differential equation is

$$u_{xx} + u_{yy} + u_{zz} = -3 \sin x \cos y \cos z.$$

The Dirichlet boundary condition is chosen from the following exact solution:

$$u(x, y, z) = \sin x \cos y \cos z.$$

Fig. 3(a) (outside of the ellipse) shows a slice of the computed solution: $-u(x, y, 0)$. The ellipsoid is embedded into a unit cube $[-1, 1] \times [-1, 1] \times [-1, 1]$. Table 7 shows the errors in the infinity norm and other information. In the table, N_{irreg} and N_{contr} are the number of the total irregular grid points and the number of control points, respectively; N_{iter} is the number of iterations of the GMRES method, or the number of calls to the 3D fast Poisson solver. We see the number of iterations is independent of mesh size as in the case of two space dimensions.

Table 5
The grid refinement analysis for Example 5.1 on a Sun Ultra 10 computer

$L \times M \times N$	$\beta^+ = 1$		$\beta^+ = 10$		$\beta^+ = 1000$	
	$\|E_N\|_\infty$	Ratio	$\|E_N\|_\infty$	Ratio	$\|E_N\|_\infty$	Ratio
$26 \times 26 \times 26$	3.931×10^{-4}		6.635×10^{-4}		3.598×10^{-5}	
$52 \times 52 \times 52$	9.732×10^{-5}	4.039	1.816×10^{-4}	3.654	9.787×10^{-6}	3.676
$104 \times 104 \times 104$	2.351×10^{-5}	4.140	4.198×10^{-5}	4.326	2.266×10^{-6}	4.319

The coefficient in Ω^- is $\beta^- = 1$.

Table 6
CPU time (seconds) and the number of iterations for Example 5.1 on a Sun Ultra 10 computer

$L \times M \times N$	N_{irreg}	N_{contr}	$\beta^+ = 1$		$\beta^+ = 10$		$\beta^+ = 1000$	
			CPU	N_{iter}	CPU	N_{iter}	CPU	N_{iter}
$26 \times 26 \times 26$	920	506	52.127	1	75.602	13	89.737	19
$52 \times 52 \times 52$	3528	1828	210.789	1	318.671	13	405.130	21
$104 \times 104 \times 104$	14048	7180	917.517	1	1536.733	14	1987.950	24

The coefficient in Ω^- is $\beta^- = 1$.

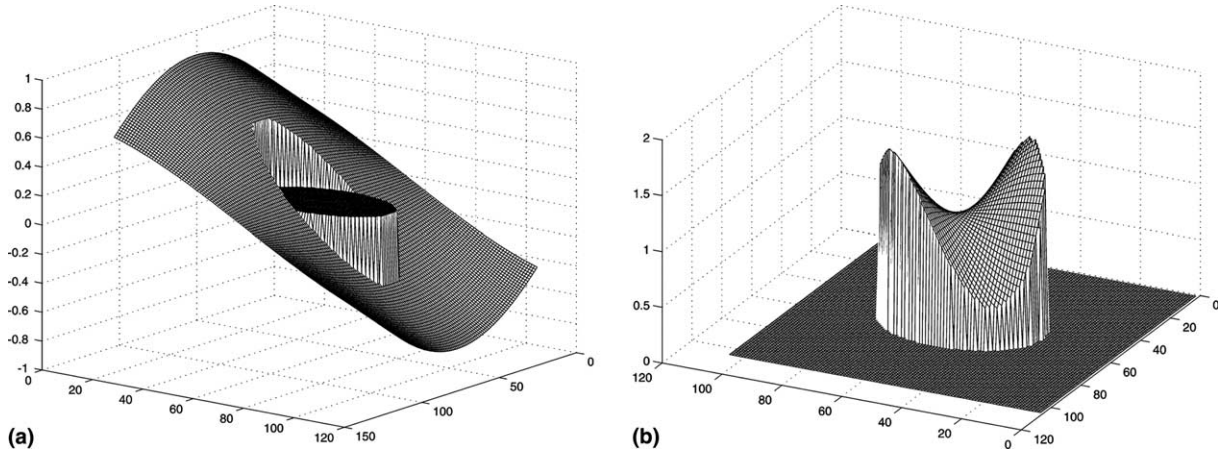

Fig. 3. (a) A slice of the computed solution for Example 5.2: $-u(x, y, 0)$. (b) A slice of the computed solution for Example 5.3. $L = M = N = 104$.

Table 7
The grid refinement analysis for Example 5.2.

$L \times M \times N$	N_{irreg}	N_{contr}	CPU (s)	N_{iter}	$\|E_N\|_\infty$	Ratio
$26 \times 26 \times 26$	720	402	43.347	18	6.464×10^{-3}	
$52 \times 52 \times 52$	2888	1512	196.011	20	7.328×10^{-4}	8.821
$104 \times 104 \times 104$	11516	5861	930.597	19	7.416×10^{-5}	9.822

Example 5.3. The differential equation is

$$u_{xx} + u_{yy} + u_{zz} = -3\pi^2 \sin \pi x \sin \pi y \cos \pi z$$

in the interior of ellipsoid $2x^2 + y^2 + z^2 = 1/4$. The Dirichlet boundary condition is chosen from the following exact solution:

$$u(x, y, z) = \sin \pi x \sin \pi y \cos \pi z + 1.$$

Again the domain is embedded into the unit cube. Fig. 3(b) (inside on the top) shows a slice of the computed solution: $u(x, y, 0)$. Table 8 shows the errors in the infinity norm and other information.

Table 8
The grid refinement analysis for Example 5.3.

$L \times M \times N$	N_{irreg}	N_{contr}	CPU (s)	N_{iter}	$\|E_N\|_\infty$	Ratio
$26 \times 26 \times 26$	720	318	42.994	18	4.382×10^{-3}	
$52 \times 52 \times 52$	2888	1352	178.600	18	1.071×10^{-3}	4.092
$104 \times 104 \times 104$	11520	5562	949.558	22	2.452×10^{-4}	4.368

6. An application to an inverse problem of shape identification

In [6], we proposed a variational model and a numerical method for identifying an unknown shape in a problem motivated by electrical tomography. In this section, we show some three-dimensional simulations using the fast Poisson solver for exterior irregular domains.

The variational form of the problem is

$$\min_{\Gamma} J(\Gamma) = \frac{1}{2}\int\int\int_{\hat{\Omega}} |u(\Gamma) - u_{\mathrm{ob}}|^2 \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z + \epsilon \int\int_{\Gamma} 1 \,\mathrm{d}S, \tag{74}$$

where the given u_{ob} is the observed data in a small tube

$$\hat{\Omega} = \{(x,y,z), -1 \leqslant x \leqslant -1+\delta, 1-\delta \leqslant x \leqslant 1; -1 \leqslant y \leqslant -1+\delta, 1-\delta \leqslant y \leqslant 1; -1 \leqslant z \leqslant -1 + \delta, 1-\delta \leqslant z \leqslant 1\}, \tag{75}$$

where $\delta > 0$ is a parameter, and ϵ is a regularization parameter.

Given a domain Ω, a sub-set $\hat{\Omega}$, and a three-dimensional function u_{ob} defined on $\hat{\Omega}$, the problem is to find the unknown surface(s) Γ (within Ω) that minimizes $J(\Gamma)$.

We use the zero level set of a function $\varphi(x,y,z)$ to express an admissible surface Γ

$$\Gamma = \{\mathbf{x} \in \mathbb{R}^3 : \varphi(\mathbf{x}) = 0\}.$$

Given an admissible surface Γ, the gradient (steepest ascent) direction of J at the surface Γ is given by

$$V(\mathbf{x}) = -\nabla u \cdot \nabla p + \epsilon\kappa \quad \text{on } \Gamma, \tag{76}$$

where κ is the mean curvature of the interface Γ and $u \in H_0^1(\Omega^+)$ satisfies

$$\begin{aligned} -\Delta u &= 0 \quad \text{in } \Omega^+, \\ u &= 0 \quad \text{on } \Gamma, \\ u_n &= g \quad \text{on } \partial\Omega, \end{aligned} \tag{77}$$

and the adjoint function $p \in H^1(\Omega^+)$ satisfies

$$\begin{aligned} -\Delta p &= (u - u_{\mathrm{ob}})\chi_{\hat{\Omega}} \quad \text{in } \Omega^+, \\ p &= 0 \quad \text{on } \Gamma, \\ p_n &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{78}$$

Here, $\chi_{\hat{\Omega}}$ is the characteristic function of the domain $\hat{\Omega}$, we refer the reader to [6] for the derivation.

Since we know the steepest descent direction, we can use the steepest descent and quasi-Newton method to move an admissble Γ closer to its minima. We use the level set method as a tool to find the unknown shape that minimizes $J(\Gamma)$ by moving the surface along the steepest descent direction of $J(\Gamma)$ through the Hamilton–Jacobi equation

$$\varphi_t + V\nabla\varphi = 0 \tag{79}$$

with an artificial time variable t. The algorithm is outlined below.

6.1. Outline of the algorithm

- Select an initial level set function φ whose zero level set $\Gamma_0 = \{(x,y,z) : \varphi(x,y,z) = 0\}$ is within the domain Ω. Let $\Gamma = \Gamma_0$.

- Solve the Laplace equation (77) in the exterior of Γ using the fast solver described in Section 5.5 to get $u = u(\Gamma)$.
- Compute the difference of the computed solution with the observed data $(u(\Gamma) - u_{\text{ob}})\chi_{\hat{\Omega}}$.
- Check convergence. If $\|(u(\Gamma) - u_{\text{ob}})\chi_{\hat{\Omega}}\| \leqslant \epsilon_2$, then stop, where $\epsilon_2 > 0$ is a pre-chosen tolerance. Otherwise, continue.
- Solve the Poisson equation (78) in the exterior of Γ using the fast solver described in Section 5.5 to get $p = p(\Gamma)$.
- Evaluate the normal velocity V using the least squares interpolation scheme, see Section 4.3, to get

$$V = -\nabla u \cdot \nabla p + \epsilon\kappa \quad \text{on } \Gamma, \tag{80}$$

  where κ is the mean curvature of the interface.
- Extend the normal velocity V to a computational tube $|\varphi| \leqslant \delta_2$, where δ_2 is the width of the tube.
- Update the level set function by solving the Hamilton–Jacobi equation $\varphi_t + V\,|\nabla\varphi| = 0$.
- Reinitialize the interface.
- Let Γ be defined by the new level set function. Repeat the process if necessary.

Since the emphasis here is the application of our fast Poisson solver on irregular domains, we omit some of the details due to the space limitation and refer the reader [6] for more details. The time step size is chosen as

$$\Delta t = \min\left(10, \frac{h}{4\ v_{\max}}\right),$$

where $v_{\max}$ is the maximum magnitude of the velocity in the computational tube. Based on the CFL condition for the level set equation, we could use $\Delta t < h/v_{\max}$. However because the problem is non-linear, we take a more conservative approach.

6.2. Numerical simulations of shape identification.

We performed some numerical experiments on Sun Ultra 10 workstations with a $60 \times 60 \times 60$ grid. The computational domain is scaled to the unit cube $[-1,1] \times [-1,1] \times [-1,1]$. As stated in [6] for two-dimensional problems, the algorithm works well for single convex objects or multi-convex objects that are far apart.

We present two examples in which we know the exact solutions. In the first one the exact shape is a sphere $x^2 + y^2 + z^2 = 0.3^2$. In the second example, the exact shape is an ellipsoid $x^2 + 3y^2 + z^2 = 0.3^2$. We started with a large sphere $x^2 + y^2 + z^2 = 0.4^2$ that surrounds the exact shape. The observed data are assumed to have a noise

$$z_{ijk} = u(\Gamma^*)_{ijk} + \bar{\delta}_{ijk},$$

where Γ^* denotes the "true" interface and $\bar{\delta}_{ijk}$ is chosen as uniformly distributed random noise.

In Fig. 4, we show the evolution process of our computation for the sphere case. The parameters are $\epsilon = 0.001$, the relative noise level $\bar{\delta} = \max_{ijk}|\delta_{ijk}|/\max_{ijk}|u_{ijk}|$ is 17%. The stopping criteria is $|J| \leqslant 10^{-5}$. In Fig. 5, we show the evolution process of our computation using the slices of the computed shape for the ellipsoid case. We show the results with $\epsilon = 0.001$. The relative noise level once again is 17%. In both cases, we get satisfactory results. The small difference in the final shape is mainly due to the noise in the observed data.

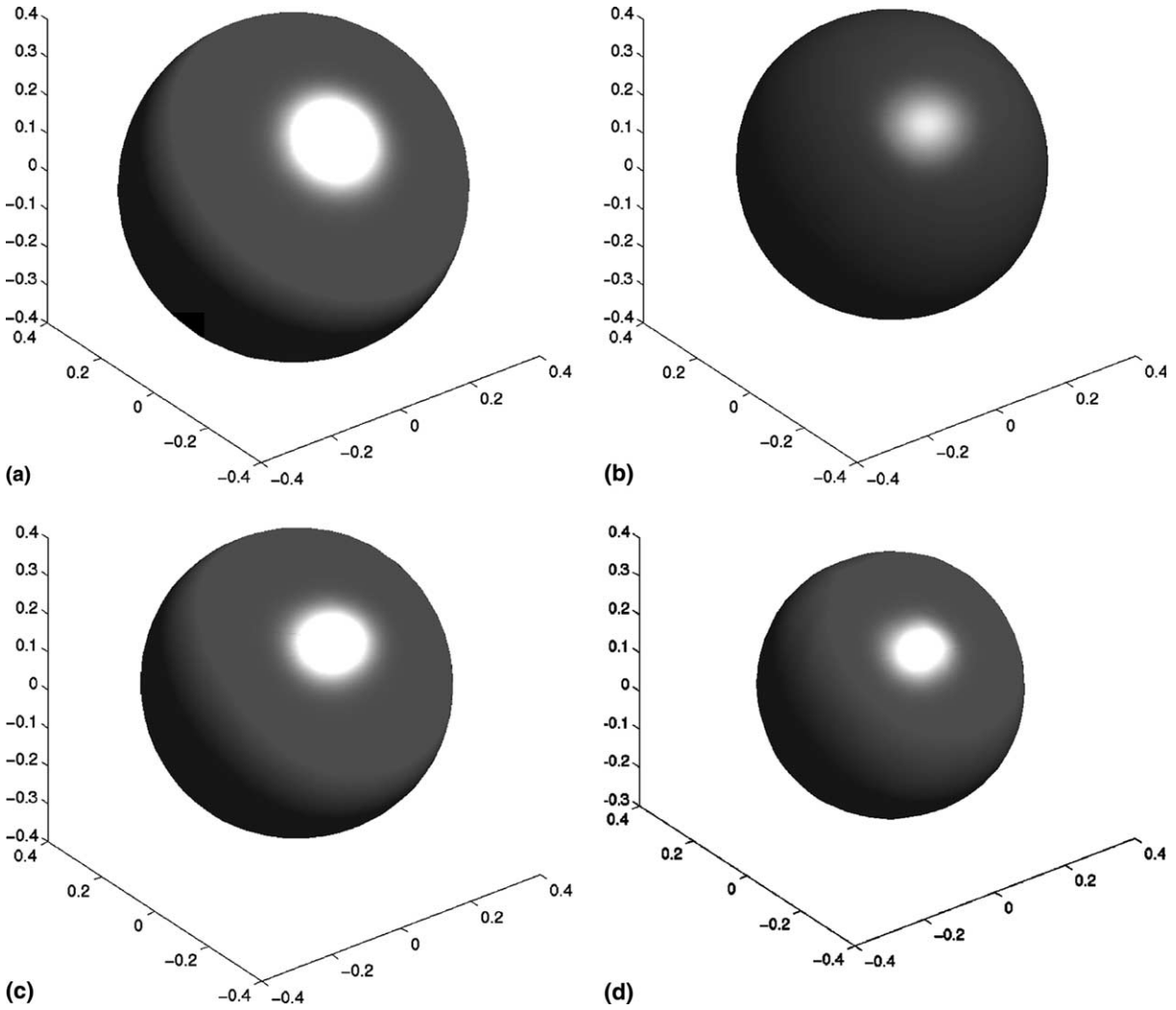

Fig. 4. The computed shape for the sphere case using a $60 \times 60 \times 60$ grid and $\epsilon = 0.001$ at different stages: (a) the initial guess; (b) after 11 iterations; (c) after 31 iterations; (d) after 51 iterations.

7. Conclusions

In this paper, we described two numerical methods for three-dimensional elliptic interface problems in which the coefficient, the source term, the solution and its derivatives, have a discontinuity across an interface. The maximum preserving scheme coupled with the algebraic multigrid solver is relatively simpler to implement. The fast solver can only be applied to the Poisson problem with piecewise constant coefficient. The number of iterations is independent of the mesh sizes. More important, the computed normal derivatives from each side of the interface appear to be second-order accurate. The fast solver can be applied to Helmholtz/Poisson equations on irregular domains which may have many applications. The application to a free boundary problem in identifying an unknown shape using the level set method is illustrated.

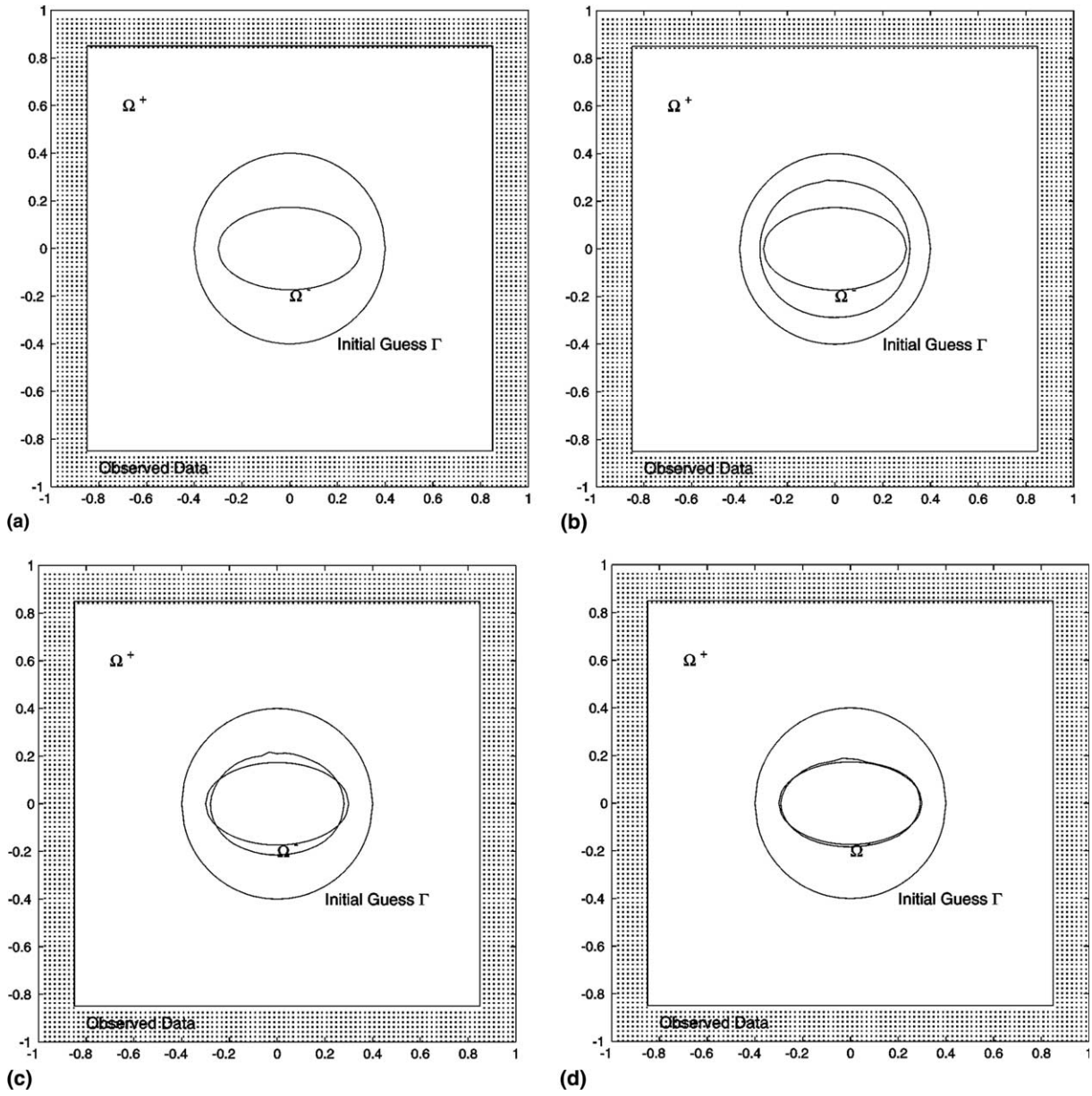

Fig. 5. The y–z transection of the computed shape for the ellipsoid case using a $60 \times 60 \times 60$ grid at different stages: (a) the initial guess; (b) after 41 iterations; (c) after 121 iterations; (d) after 201 iterations. The inner-most ellipse is the exact solution.

Acknowledgements

The second and third authors are partially supported by ARO Grants 39676-MA and 43751-MA. The third author is also partially supported by NSF Grants DMS-0073403 and DMS-0201094. We thank the North Carolina Super-computing Center (NCSC) for letting us use the computing facilities there.

References

[1] J. Adams, P. Swarztrauber, R. Sweet, Fishpack. Available from http://www.netlib.org/fishpack/.
[2] J. Bramble, J. King, A finite element method for interface problems in domains with smooth boundaries and interfaces, Adv. Comput. Math. 6 (1996) 109–138.
[3] D. Calhoun, A Cartesian grid method for solving the streamfunction-vorticity equations in irregular geometries, Ph.D Thesis, University of Washington, 1999.
[4] H.W. Chen, L. Greengard, A method of images for the evaluation of electrostatic fields in systems of closely spaced conducting cylinders, SIAM J. Appl. Math. 58 (1998) 122–141.
[5] S. Deng. Immersed interface method for three dimensional interface problems and applications, Ph.D Thesis, North Carolina State University, 2000.
[6] K. Ito, K. Kunisch, Z. Li, Level-set function approach to an inverse interface problem, Inverse Problems 17 (2001) 1225–1242.
[7] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (1994) 1019–1044.
[8] R.J. LeVeque, Z. Li, Immersed interface method for Stokes flow with elastic boundaries or surface tension, SIAM J. Sci. Comput. 18 (1997) 709–735.
[9] Z. Li, The immersed interface method – a numerical approach for partial differential equations with interfaces, Ph.D Thesis, University of Washington, 1994.
[10] Z. Li, A fast iterative algorithm for elliptic interface problems, SIAM J. Numer. Anal. 35 (1998) 230–254.
[11] Z. Li, K. Ito, Maximum principle preserving schemes for interface problems with discontinuous coefficients, SIAM J. Sci. Comput. 23 (2001) 1225–1242.
[12] X. Liu, R. Fedkiw, M. Kang, A boundary condition capturing method for Poisson's equation on irregular domain, J. Comput. Phys. 160 (2000) 151–178.
[13] A. Mayo, A. Greenbaum, Fast parallel iterative solution of Poisson's and the biharmonic equations on irregular regions, SIAM J. Sci. Stat. Comput. 13 (1992) 101–118.
[14] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1988) 12–49.
[15] J.W. Ruge, K. Stuben, Algebraic multigrid, in: S.F. McCormick (Ed.), Multigrid Method, SIAM, Philadelphia, 1987.
[16] K. Schittkowski, QL-quadratic Programming, version 1.5, 1991. Available from http://www.uni-bayreuth.de/departments/math/~kschittkowski/ql.htm.
[17] A. Wiegmann, K. Bube, The immersed interface method for nonlinear differential equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 35 (1998) 177–200.