UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

A Fast Iterative Algorithm for Elliptic Interface Problems

Zhilin Li

September 1995

CAM Report 95-40

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90024-1555

A FAST ITERATIVE ALGORITHM FOR ELLIPTIC INTERFACE PROBLEMS [*]

ZHILIN LI [†]

Abstract. A fast, second order accurate iterative method is proposed for the elliptic equation

$$\nabla \cdot (\beta(x,y)\nabla u) = f(x,y)$$

in a rectangular region Ω in 2 space dimensions. We assume that there is an irregular interface across which the coefficient β, the solution u and its derivatives, and/or the source term f may have jumps. We are especially interested in the cases where the jump in β is large. The interface may or may not align with a underlying Cartesian grid. The idea in our approach is to precondition the differential equation before applying the immersed interface method proposed by LeVeque and Li [SINUM, 4 (1994), pp. 1019-1044]. In order to take advantage of fast Poisson solvers on a rectangular region, an intermediate unknown function, the jump in the normal derivative across the interface, is introduced. Our discretization is equivalent to using a second order difference scheme for a corresponding Poisson equation in the region, and a second order discretization for a Neumann-like interface condition. Thus second order accuracy is guaranteed. A GMRES iteration is employed to solve the Schur complement system derived from the discretization. A new weighted least squares method is also proposed to approximate interface quantities from a grid function. Numerical experiments are provided and analyzed. The number of iterations in solving the Schur complement system appears to be independent of both the jump in the coefficient and the mesh size.

Key words. elliptic equation, discontinuous coefficients, immersed interface method, Cartesian grid, Schur complement, GMRES method, preconditioning

AMS subject classifications. 65N06, 65N22, 65N50, 65F35

1. Introduction.

Consider the elliptic equation

$$\begin{aligned} &\nabla\left(\beta(x,y)\nabla u\right) = f(x,y), \quad x \in \Omega \\ &\text{Given BC} \quad \text{on} \quad \partial\Omega \end{aligned} \tag{1.1}$$

in a rectangular domain Ω in two space dimensions. Within the region, suppose there is an irregular interface Γ across which the coefficient β is discontinuous. Referring to Fig 1, we assume that $\beta(x,y)$ has a constant value in each sub-domain,

$$\beta(x,y) = \begin{cases} \beta^+ & \text{if } (x,y) \in \Omega^+ \\ \beta^- & \text{if } (x,y) \in \Omega^-. \end{cases} \tag{1.2}$$

The interface Γ may or may not align with a underline Cartesian grid.

Depending on the properties of the source term $f(x,y)$, we usually have jump conditions across the interface Γ:

$$[u] \overset{\text{def}}{=} u^+\left(X(s),Y(s)\right) - u^-\left(X(s),Y(s)\right) = w(s), \tag{1.3}$$

$$[\beta u_n] \overset{\text{def}}{=} \beta^+ u_n^+\left(X(s),Y(s)\right) - \beta^- u_n^-\left(X(s),Y(s)\right) = v(s), \tag{1.4}$$

[*] This work was supported by URI grant #N00014092-J-1890 from ARPA, NSF Grant DMS-9303404, and DOE Grant DE-FG06-93ER25181.

[†] Department of Mathematics, University of California at Los Angeles, Los Angeles, CA 90095. (zhilin@math.ucla.edu).

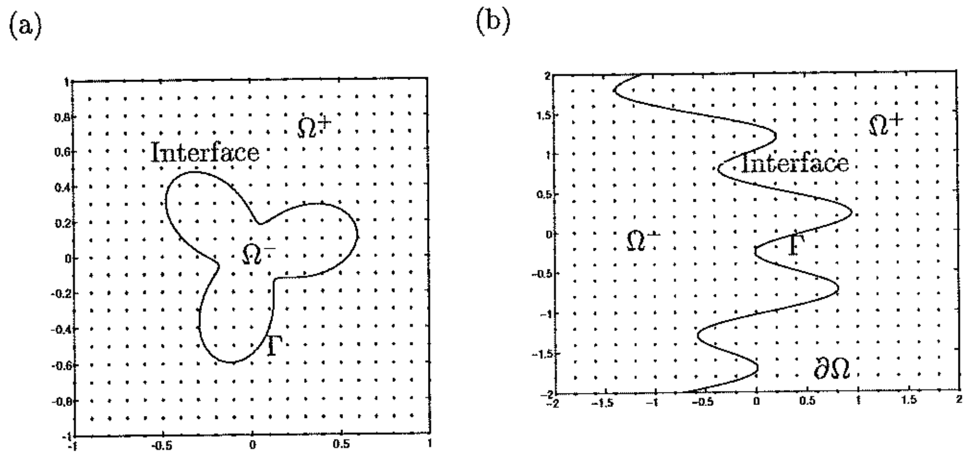

FIG. 1. *Two typical computational domains and interfaces with uniform Cartesian grids.*

where $(X(s), Y(s))$ is the arc-length parameterization of Γ, the superscripts $-$ or $+$ denotes the limiting values of a function from one side or the other of the interface. These two jump conditions can be either obtained by physical reasoning or derived from the differential equation, see [2, 9, 14] etc. Note in potential theory, $v(s) \not\equiv 0$ corresponds to a single layer source along the interface Γ, while $w(s) \not\equiv 0$ corresponds to a double layer source. The normal derivative u_n usually has a kink across the interface due to the discontinuity in the coefficient β. If $w(s) \not\equiv 0$, then the solution u itself would be discontinuous across the interface.

There are many applications in solving elliptic equations with discontinuous coefficients, for example, steady state heat diffusion or electrostatic problems, multi-phase and porous flow, solidification problems, and bubble computations etc. There are two main concerns in solving (1.1)–(1.4) numerically:

- How to discretize (1.1)–(1.4) to certain accuracy. It is difficult to study the consistency and the stability of a numerical scheme because of the discontinuities across the interface.
- How to solve the resulting linear system efficiently. Usually if the jump in the coefficient is large, then the resulting linear system is ill-conditioned, and the number of iterations in solving such a linear system is large and proportional to the jump in the coefficient.

There are a few numerical methods designed to solve elliptic equations with discontinuous coefficients, for example, harmonic averaging, smoothing method, and finite element approach *etc.*, see [2] for a brief review of different methods. Most of these methods can be second order accurate in the l-1 or the l-2 norm, but not in the l-∞ norm, since they may smooth out the solution near the interface.

A. Mayo and A. Greenbaum [14, 15] have derived an integral equation for elliptic interface problems with piecewise coefficients. By solving the integral equation, they can solve such interface problems to second order accuracy in the l-∞ norm using the techniques developed by A. Mayo in [13, 14] for solving Poisson and biharmonic equations on irregular regions. The total cost includes solving the integral equation and a regular Poisson equation using a fast solver, so this gives a fast algorithm. The possibility of extension to variable coefficients is mentioned in [14].

R.J. LeVeque and Z. Li have recently developed a different approach for discretizing elliptic problems with irregular interfaces called the *immersed interface method* (IIM) [2, 9], which can handle both discontinuous coefficients and singular sources. This approach has also been applied to three dimensional elliptic equations [7], parabolic equations[10, 11, 12], hyperbolic wave equations with discontinuous coefficients [4, 5], and the incompressible Stokes flow problems with moving interfaces [3, 6]. L. Adams [1] has successfully implemented a multi-grid algorithm for the immersed interface method. However, there are some numerical examples with large jumps in the coefficients in which the immersed interface method may fail to give accurate answers or converge very slowly.

In this paper, we propose a fast algorithm for elliptic equations with large jumps in the coefficients. The idea is to precondition the elliptic equation before using the immersed interface method. In order to take advantage of fast Poisson solvers on rectangular regions, we introduce an intermediate unknown function $[u_n](s)$ which is defined only on the interface. Then we discretize a corresponding Poisson equation, which has different sources from the original one, using the standard five-point stencil with some modification in the right hand side. Our discretization is equivalent to using a second order difference scheme to approximate the Poisson equation in the interior region Ω^+ and Ω^-, and a second order discretization for the Neumann-like interface condition

$$\beta^+ u_n^+ - \beta^- u_n^- = v. \tag{1.5}$$

Thus from the error analysis for elliptic equations with Neumann boundary conditions, for example, see [17], we would have second order accurate solution at all grid points including those near or on the interface. A GMRES method is employed to solve the Schur complement system derived from the discretization. A new weighted least squares method is proposed to approximate interface quantities such as $u_n^{\pm}$ from a grid function defined on the entire domain. This new technique has been successfully applied in the multi-grid method for interpolating the grid function between different levels [1] with remarkable improvement in the computed solution. These ideas will be discussed in detail in the following sections. The method described in this paper seems to be very promising not only because it is second order accurate, but also because the number of iterations in solving the Schur complement system is almost independent of both the jumps in the coefficients and the mesh size. This has been observed from our numerous numerical experiments, though we have not been able to prove this theoretically. Our new method has been used successfully for the computation of some inverse problems [20].

This paper is organized as follows. In Section 2, we precondition (1.1)–(1.4) to get an equivalent problem. In Section 3, we use the IIM idea to discretize the equivalent problem and derive the Schur complement system. The weighted least squares approach to approximate $u_n^{\pm}$ from the grid function u_{ij} is discussed in Section 4. Some implementation details are addressed in Section 5. Brief convergence analysis is given in Section 6. An efficient preconditioner for the Schur complement system is proposed in Section 7. Numerical experiments and analysis can be found in Section 8. Some new approaches in the error analysis involving interfaces are also introduced there.

2. Preconditioning the PDE to an equivalent problem. The problem we intend to solve is the following:

Problem (I).

$$\nabla\left(\beta(x,y)\nabla u\right) = f(x,y), \tag{2.6a}$$

$$\textit{Given BC} \quad \textit{on} \quad \partial\Omega, \tag{2.6b}$$

with specified jump conditions along the interface Γ

$$[u] = w(s), \tag{2.7a}$$

$$[\beta u_n] = v(s). \tag{2.7b}$$

Consider the solution set $u_g(x,y)$ of the following problem as a functional of $g(s)$.

Problem (II).

$$\begin{aligned} \Delta u + \frac{\nabla\beta^-}{\beta^-}\cdot\nabla u &= \frac{f}{\beta^+} \quad \textit{if } x\in\Omega^+ \\ \Delta u + \frac{\nabla\beta^+}{\beta^+}\cdot\nabla u &= \frac{f}{\beta^-} \quad \textit{if } x\in\Omega^-, \end{aligned} \tag{2.8a}$$

$$\textit{Given BC} \quad \textit{on } \partial\Omega, \tag{2.8b}$$

with specified jump conditions[1]

$$[u] = w(s), \tag{2.9a}$$

$$[u_n] = g(s). \tag{2.9b}$$

Let the solution of Problem (I) be $u^*(x,y)$, and define

$$g^*(s) = [u_n^*](s) \tag{2.10}$$

along the interface Γ. Then $u^*(x,y)$ satisfies the elliptic equation (2.8a)-(2.8b) and jump conditions (2.9a)-(2.9b) with $g(s)\equiv g^*$. In other words, $u_{g^*}(x,y)\equiv u^*(x,y)$, and

$$[\beta\frac{\partial u_{g^*}}{\partial n}] = v(s). \tag{2.11}$$

is satisfied. Therefore, solving Problem (I) is equivalent to finding the corresponding g^* and then $u_{g^*}(x,y)$ in Problem (II). Notice that g^* is only defined along the interface, so it is one dimension lower than $u(x,y)$. Problem (II) is an elliptic interface problem which is much easier to solve because the jump condition $[u_n]$ is given instead of $[\beta u_n]$. With the immersed interface method, it is very easier to construct a second order scheme which also satisfies the conditions of the maximum principle. In this paper, we suppose β is piecewise constant as in (1.2), so Problem (II) is a Poisson equation with a discontinuous source term and given jump conditions. We can then use the standard five-point stencil to discretize the left hand side of (2.8a), but modify the right hand side to get a second order scheme, see [2, 9] for the detail. Thus we can take advantage of fast Poisson solvers for the discrete system. The cost in solving

[1] The jump conditions (2.9a) and (2.9b) depend on the singularities of the source term $f(x,y)$ along the interface. However, in the expression of (2.8a), we do not need information of $f(x,y)$ on the interface Γ, so there is no need to write $f(x,y)$ differently.

Problem (II) is just a little more than that in solving a regular Poisson equation on the rectangle with a smooth solution. For more general variable coefficient, the discussions in this paper are still valid except we can not use a fast Poisson solver because of the convection term $(\nabla\beta \cdot \nabla u)/\beta$ in (2.8a). However a multi-grid approach developed by L. Adams [1] perhaps can be used to solve Problem (II).

We wish to find numerical methods with which we can compute $u_{g^*}(x,y)$ to second order accuracy. We also hope that the total cost in computing g^* and u_{g^*} is less than that in computing u_{g^*} through the original Problem (I). The key to success is computing g^* efficiently. Below we begin to describe our method to solve g^*. Once g^* is found, we just need one more fast Poisson solver to get the solution $u^*(x,y)$.

3. Discretization. We use a uniform grid on the rectangle $[a,b] \times [c,d]$ where the Problem (I) is defined:

$$x_i = a + ih, \; y_j = a + jh, \; 0 \le i \le m, \; 0 \le j \le n.$$

We assume that $h = (b-a)/m = (d-c)/n$ for simplicity. We use a cubic spline $\vec{X}(s) = (X(s), Y(s))$ passing through a number of control points (X_k, Y_k), $k = 1, 2, \cdots, n_b$, to express the immersed interface, where s is the arc-length of the interface and (X_k, Y_k) is the position of the k-th point on the interface Γ. Other representations of the interface are possible. A level set formulation is currently under investigation.

Any other quantity $q(s)$ defined on the interface such as $w(s)$ and $g(s)$ can also be expressed in terms of a cubic spline with the same parameter s. Since cubic splines are twice differentiable we can gain access to the value of $q(s)$ and its first or second derivatives at any point on the interface in a continuous manner.

We use upper case letters to indicates the solution of the discrete problem and lower case letters for the continuous solutions.

Given W_k and G_k, the discrete form of jump conditions (2.9a) and (2.9b), with the immersed interface method, the discrete form of (2.8a) can be written as

$$L_h U_{ij} = \frac{f_{ij}}{\beta_{ij}} + C_{ij}, \;\; 1 \le i \le m-1, \;\; 1 \le j \le n-1, \tag{3.12}$$

where

$$L_h U_{ij} \overset{\text{def}}{=} \frac{U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1} - 4\,U_{i,j}}{h^2},$$

is the discrete Laplacian operator using the standard five-point stencil. Note that if (x_i, y_j) happens to be on the interface, then f_{ij}/β_{ij} is defined as the limiting value from a pre-chosen side of the interface. C_{ij} is zero except at those irregular grid points where the interface cuts through the five-point stencil. A fast Poisson solver, for example, FFT, ADI, Cyclic reduction, or Multi-grid, can be applied to solve (3.12). The solution U_{ij} depends on G_k, W_k, $k = 1, 2, \ldots$, continuously. In matrix and vector form we have

$$AU + \mathcal{B}(W, G) = F, \tag{3.13}$$

where $AU = F$ is the discrete linear system for the Poisson equation when W_k, G_k are all zero. The solution is smooth for such a system. $\mathcal{B}(W,G)$ is a mapping from $W = [W_1, W_2, \ldots, W_{n_b}]^T$ and $G = [G_1, G_2, \ldots, G_{n_b}]^T$ to C_{ij} in (3.12). From [2, 9] we know that $\mathcal{B}(W,G)$ depends on the first and second derivatives of $w(s)$, and the first derivative of $g(s)$, where the differentiation is carried out along the interface. At this stage we do not know whether such a mapping is linear or not. However in the discrete case, all the derivatives are obtained by differentiating the corresponding splines which are linear combination of the values on those control points. Therefore $\mathcal{B}(W,G)$ is indeed linear function of W and G and can be written as

$$\mathcal{B}(W,G) = BG - B_1 W,$$

where B and B_1 are two matrices with real entries. Thus (3.13) becomes

$$\begin{aligned} AU + BG &= F + B_1 W \\ &\overset{\text{def}}{=} \bar{F}. \end{aligned} \tag{3.14}$$

The solution U of the equation above certainly depends on G and we are interested in finding G^* which satisfies the discrete form of (2.7b)

$$[\beta U_n](G^*) - V = \beta^+ U_n^+(G^*) - \beta^- U_n^-(G^*) - V \equiv 0, \tag{3.15}$$

where the components of the vectors U_n^+ and U_n^- are discrete approximation of the normal derivative at control points from each side of the interface. In the next section, we will discuss how to use the known jump G, and sometimes also V, to interpolate U_{ij} to get U_n^+ and U_n^- in detail. As we will see in the next section, U_n^+ and U_n^- depend on U, G and V linearly

$$\begin{aligned} \beta^+ U_n^+ - \beta^- U_n^- - V &= EU + DG + \bar{P}V - V, \\ &= EU + DG - PV \end{aligned} \tag{3.16}$$

where E, D, and $\bar{P}$ are some matrices and $P = I - \bar{P}$. Combine (3.14) and (3.16) to obtain the linear system of equations for U and G:

$$\begin{bmatrix} A & B \\ E & D \end{bmatrix} \begin{bmatrix} U \\ G \end{bmatrix} = \begin{bmatrix} \bar{F} \\ PV \end{bmatrix}. \tag{3.17}$$

The solution U and G are the discrete forms of $u_{g^*}(x,y)$ and g^*, the solution of Problem (II) which satisfies (2.11).

The next question is how to solve (3.17) efficiently. The GMRES method applied to (3.17) directly or the multi-grid approach [1] are two attractive choices. However, in order to take advantage of fast Poisson solvers, we have decided to solve G in (3.17) first, and then to find the solution U by using one more fast Poisson solver. Eliminating U from (3.17) gives a linear system for G

$$\begin{aligned} (D - EA^{-1}B)G &= PV - EA^{-1}\bar{F} \\ &\overset{\text{def}}{=} \bar{V} \end{aligned} \tag{3.18}$$

This is an $n_b \times n_b$ system for G, a much smaller linear system compared to the one for U. The coefficient matrix is the Schur complement of D in (3.17). In practice, the matrices A, B, E, D, P, and the vectors $\bar{V}$, $\bar{F}$ are never formed. The matrix and vector form are merely for theoretical purposes. Thus an iterative method, such as the GMRES iteration [18], is preferred. The way we compute (3.16) will dramatically change the condition number of (3.18).

4. A weighted least squares approach for computing interface quantities from a grid function. When we apply the GMRES method to solve the Schur complement system of (3.18) for G^*, we need to compute the matrix-vector multiplication, which is equivalent to computing U_n^- and U_n^+ with the knowledge of U_{ij} and the jump condition $[U_n]$. This turns out to be a crucial step in solving the linear system (3.18) for G^*. Our approach is based on a weighted least squares formulation. The idea described here can also be, and has been, applied to the case, where we want to approximate some quantities on the interface from a grid function. For example, interpolating U_{ij} to the interface to get $U^-(X,Y)$ or $U^+(X,Y)$, where (X,Y) is some point on the interface. This new approach has also been successfully applied to the multigrid method for interpolating the grid function between different levels by L. Adams[1] with remarkable improvement in the computed solution.

We start from the continuous situation, the discrete version can be obtained accordingly. Let $u(x,y)$ be a piecewise smooth function, with discontinuities only along the interface. We want to interpolate $u(x_i,y_j)$ to get approximations to the normal derivatives $u_n^-(X,Y)$ and $u_n^+(X,Y)$, which are only defined on the interface, to second order.

Our approach is inspired by Peskin's method in interpolating a velocity field $u(x,y)$ to get the velocity of the interface using a discrete delta function. The continuous and discrete forms are the following:

$$u(\vec{X}) = \iint_\Omega u(x,y)\,\delta(X-x)\,\delta(Y-y)\,dxdy, \tag{4.19}$$

$$u(\vec{X}) = h^2 \sum_{i,j} u_{ij}\,\delta_h(X-x_i)\,\delta_h(Y-y_j), \tag{4.20}$$

where $\vec{X}=(X,Y)$, and δ_h is a discrete delta function. A commonly used one is

$$\delta_h(x) = \begin{cases} \frac{1}{4h}\left(1+\cos\left(\pi x/2h\right)\right) & \text{if } |x| < 2\,h \\ 0 & \text{if } |x| \ge 2\,h. \end{cases}$$

Notice that $\delta_h(x)$ is a smooth function of x. Peskin's approach is very robust and only a few neighboring grid points near $\vec{X}$ are involved. However this approach is only first order accurate and may smear out the solution near the interface.

Our interpolation formula for u_n^-, for example, can be written in the following form

$$u_n^-(\vec{X}) \approx \sum_{i,j} \gamma_{ij}\,u_{ij}\,d_\alpha(|\vec{X}-\vec{x}_{ij}|) - Q, \tag{4.21}$$

where $d_\alpha(r)$ is a function of the distance measured from the point $\vec{X}$,

$$d_\alpha(r) = \alpha\,\delta_{\alpha/2}(r) = \begin{cases} \frac{1}{2}\left(1+\cos\left(\pi r/\alpha\right)\right) & \text{if } r < \alpha \\ 0 & \text{if } r \ge \alpha. \end{cases} \tag{4.22}$$

Q is a correction term which can be determined once γ_{ij} are known. Although we are trying to approximate the normal derivatives here, the same principle also applies to the function values U as well with different choices of γ_{ij} and Q. Note no extra effort is needed to decide which grid points should be involved. Therefore, expression (4.21) is robust and depends on the the grid function u_{ij} continuously, two very attractive properties of Peskin's formula (4.20). In addition to the advantages of Peskin's approach, we also have flexibility in choosing the coefficients γ_{ij} and the correction term Q to achieve second order accuracy. The parameter α in (4.21) can be fixed or chosen according to problems, see Section 8.

Below we discuss how to use the immersed interface method to determine the coefficients γ_{ij} and the correction term Q. They are different from point to point on the interface. So they should really be labeled as $\gamma_{ij,\vec{X}}$, *etc.* But for simplicity of notation we will concentrate on a single point $\vec{X} = (X, Y)$ and drop the subscript $\vec{X}$.

Since the jump condition is given in the normal direction, we introduce local coordinates at (X, Y),

$$\begin{aligned} \xi &= (x - X)\cos\theta + (y - Y)\sin\theta, \\ \eta &= -(x - X)\sin\theta + (y - Y)\cos\theta, \end{aligned} \tag{4.23}$$

where θ is the angle between the x-axis and the normal direction. Under such new coordinates, the interface can be parameterized by $\xi = \chi(\eta)$, $\eta = \eta$. Note that $\chi(0) = 0$ and, provided the interface is smooth at (X, Y), $\chi'(0) = 0$ as well. The solution of the Poisson equation Problem (II) will satisfies the following interface relations, see [2, 9] for the derivation,

$$\begin{aligned} &u^+ = u^- + w, \\ &u_\xi^+ = u_\xi^- + g, \\ &u_\eta^+ = u_\eta^- + w', \\ &u_{\xi\xi}^+ = u_{\xi\xi}^- + g\,\chi'' - w'' + \left[\frac{f}{\beta}\right] \\ &u_{\eta\eta}^+ = u_{\eta\eta}^- - g\,\chi'' + w'', \\ &u_{\xi\eta}^+ = u_{\xi\eta}^- + g\,\chi'' + g'. \end{aligned} \tag{4.24}$$

Let (ξ_i, η_j) be the ξ-η coordinates of (x_i, y_j), then we have

$$u(x_i, y_j) \;=\; u^\pm + u_\xi^\pm\,\xi_i + u_\eta^\pm\,\eta_j + \frac{1}{2}u_{\xi\xi}^\pm\,\xi_i^2 + \frac{1}{2}u_{\eta\eta}^\pm\,\eta_j^2 + u_{\xi\eta}^\pm\,\xi_i\eta_j + O(h^3) \tag{4.25}$$

where the $+$ or $-$ sign depends on whether (ξ_i, η_j) lies on the $+$ or $-$ side of Γ.

Using Taylor expansion of (4.25) about (X, Y) in the new coordinates, after collecting terms we have

$$\begin{aligned} u_n^- \;&\approx\; \sum_{i,j} \gamma_{ij}\, u(x_i, y_j)\, d_\alpha(|\vec{X} - \vec{x}_{ij}|) - Q \\ &= a_1 u^- + a_2 u^+ + a_3\, u_\xi^- + a_4\, u_\xi^+ + a_5\, u_\eta^- + a_6 u_\eta^+ + a_7\, u_{\xi\xi}^- + a_8\, u_{\xi\xi}^+ \\ &\quad + a_9\, u_{\eta\eta}^- + a_{10}\, u_{\eta\eta}^+ + a_{11}\, u_{\xi\eta}^- + a_{12}\, u_{\xi\eta}^+ - Q + O(h^3 \max|\gamma_{ij}|) \\ &= (a_1 + a_2)u^- + (a_3 + a_4)u_\xi^- + (a_5 + a_6)u_\eta^- + (a_7 + a_8)u_{\xi\xi}^- \\ &\quad + (a_9 + a_{10})u_{\eta\eta}^- + (a_{11} + a_{12})u_{\xi\eta}^- + a_2[u] + a_4[u_\xi] + a_6[u_\eta] \\ &\quad + a_8[u_{\xi\xi}] + a_{10}[u_{\xi\eta}] + a_{12}[u_{\xi\eta}] - Q + O(h^3 \max|\gamma_{ij}|), \end{aligned}$$

where the a_j are given by

$$
\begin{aligned}
a_1 &= \sum_{(x_i,y_j)\in\Omega^-} \gamma_{ij}\, d_\alpha(|\vec{X}-\vec{x_{ij}}|) & a_2 &= \sum_{(x_i,y_j)\in\Omega^+} \gamma_{ij}\, d_\alpha(|\vec{X}-\vec{x_{ij}}|) \\
a_3 &= \sum_{(x_i,y_j)\in\Omega^-} \xi_k\gamma_{ij}\, d_\alpha(|\vec{X}-\vec{x_{ij}}|) & a_4 &= \sum_{(x_i,y_j)\in\Omega^+} \xi_k\gamma_{ij}\, d_\alpha(|\vec{X}-\vec{x_{ij}}|) \\
a_5 &= \sum_{(x_i,y_j)\in\Omega^-} \eta_k\gamma_{ij}\, d_\alpha(|\vec{X}-\vec{x_{ij}}|) & a_6 &= \sum_{(x_i,y_j)\in\Omega^+} \eta_k\gamma_{ij}\, d_\alpha(|\vec{X}-\vec{x_{ij}}|) \\
a_7 &= \sum_{(x_i,y_j)\in\Omega^-} \frac{1}{2}\xi_k^2\gamma_{ij}\, d_\alpha(|\vec{X}-\vec{x_{ij}}|) & a_8 &= \sum_{(x_i,y_j)\in\Omega^+} \frac{1}{2}\xi_k^2\gamma_{ij}\, d_\alpha(|\vec{X}-\vec{x_{ij}}|) \\
a_9 &= \sum_{(x_i,y_j)\in\Omega^-} \frac{1}{2}\eta_k^2\gamma_{ij}\, d_\alpha(|\vec{X}-\vec{x_{ij}}|) & a_{10} &= \sum_{(x_i,y_j)\in\Omega^+} \frac{1}{2}\eta_k^2\gamma_{ij}\, d_\alpha(|\vec{X}-\vec{x_{ij}}|) \\
a_{11} &= \sum_{(x_i,y_j)\in\Omega^-} \xi_k\eta_k\gamma_{ij}\, d_\alpha(|\vec{X}-\vec{x_{ij}}|) & a_{12} &= \sum_{(x_i,y_j)\in\Omega^+} \xi_k\eta_k\gamma_{ij}\, d_\alpha(|\vec{X}-\vec{x_{ij}}|).
\end{aligned}
$$

From the interface relations (4.24) we know that all the jumps in the expression above can be expressed in terms of the known information. Since $u_n^- = u_\xi^-$, we obtain the linear system of equations for the coefficients γ_{ij}:

$$
\begin{aligned}
a_1 + a_2 &= 0, \\
a_3 + a_4 &= 1, \\
a_5 + a_6 &= 0, \\
a_7 + a_8 &= 0, \\
a_9 + a_{10} &= 0, \\
a_{11} + a_{12} &= 0.
\end{aligned} \tag{4.26}
$$

Note that we would have the exact same equation if we want to interpolate a smooth function $u(x_i, y_j)$ to get an approximation u_n at $\vec{X}$ to second order accuracy. The discontinuities across the interface only contribute to the correction term Q. This agrees with our analysis in [2, 9] for Poisson equations with discontinuous and/or singular sources, where we can still use the classical five-point scheme but add a correction term to the right hand side at irregular grid points.

If the linear system (4.26) has a solution, then we can obtain a second order approximation to the normal derivative u_n^- by choosing an appropriate correction term Q. Therefore we want to choose α big enough, say $\alpha \geq 1.6h$, such that at least six grid points are involved. Usually we have an under-determined linear system which has infinitely many solutions. We should then choose the one γ_{ij}^* with the minimal 2-norm

$$
\sum_{i,j} (\gamma_{ij}^*)^2 = \min_{\gamma_{ij}} \sum_{i,j} \gamma_{ij}^2, \qquad \text{subject to} \quad (4.26).
$$

For such a solution, each γ_{ij}^* will have roughly the same magnitude $O(1/h)$, so $\gamma_{ij}^* d_\alpha(|\vec{X} - \vec{x_{ij}}|)$ is roughly a decreasing function of the distance measured from $\vec{X}$. This is one of desired properties of our interpolation. In practice, only a hand full of grid points, controlled by the parameter α, are involved. Those grid points which are closer to (X, Y) have more influence than others which are further away.

When we know the coefficients γ_{ij}, we also know the a_k's. From the a_k's and the interface relations (4.24), we can determine the correction term Q easily,

$$\begin{aligned} Q \;&=\; a_2\, w + a_4\, g + a_6\, w' + a_8 \left(g\chi'' - w'' + \left[\frac{f}{\beta}\right] \right) \\ &\quad + a_{10}\,[w'' - g\,\chi''] + a_{12}\,(w'\chi'' + g'). \end{aligned} \tag{4.27}$$

Thus we are able to compute u_n^- to second order accuracy. We can derive a formula for u_n^+ in exactly the same way. However, with the relation $u_n^+ = u_n^- + g$, we can write down a second order interpolation scheme for u_n^+ immediately

$$u_n^+ \;\approx\; \sum_{i,j} \gamma_{ij}\, u(x_i, y_j)\, d_\alpha(|\vec{X} - \vec{x}_{ij}|) - Q + g, \tag{4.28}$$

where γ_{ij} is the solution we computed for u_n^-. In the next section, we will explain an important modification of either (4.21) or (4.28) depends on the magnitude of β^- or β^+.

We should mention another intuitive approach, one-sided interpolation, in which we only use grid points on the proper side of the interface in computing a limiting value at the interface:

$$u_n^- = \sum_{(x_i,y_j)\in\Omega^-} \gamma_{ij}\, u_{ij}; \qquad u_n^+ = \sum_{(x_i,y_j)\in\Omega^+} \bar{\gamma}_{ij}\, u_{ij}.$$

This approach does not make use of the interface relations (4.24), so we have to have at least six points from each side in order to have a second order scheme. Note that we can also use the least squares technique described in this section for one-sided interpolation. This approach has been tested already. The weighted least squares approach using the interface relations (4.24) appears superior in practice. It has the following advantages:

- Fewer grid points are involved. When we make use of the interface relation, compared to the one-sided interpolation, the number of grid points which are involved is reduced roughly by half.
- Second order accuracy with smaller error constant. The grid points involved in our approach are clustered around the point (X,Y) on the interface, and those which are closer to (X,Y) have more influence than those which are further away in our weighted least squares approach. We have smaller error constant in the Taylor expansions compared to the one-sided interpolation. The error constant can be as much as $8 \sim 27$ times smaller as the one-sided interpolation. In two dimensional computation, we can not take m and n to be very large, to have a smaller error constant sometimes is as important as to have a high order accurate method.
- Robust and smoother error distribution. We have a robust way in choosing the grid points which are involved. The interpolation formulas (4.21) and (4.28) depend continuously on the location (X,Y) and the grid points (x_i, y_j), and so does the truncation error for these two interpolation schemes. In other words, we will have a smooth error distribution. This is very important in moving interface problems where we do no want to introduce any non-physical oscillations.

- No break downs. In one-sided interpolation, sometimes we can not find enough grid points in one particular side of the interface, then the one-sided interpolation will break down. In our approach, every grid point on one side is connected to the other by the interface relations (4.24). So no break down will occur.
- Trade off or disadvantages. The only trade off of our weighted least squares approach is that we have to solve a under-determined 6 by p linear system of equation (where $p \geq 6$) instead of solving one that is 6 by 6. The larger α is, the more computational cost in solving (4.26). Fortunately, the linear system has full row rank and can be solved by the LR-RU method [8] or other efficient least squares solvers.

5. Some details in implementation. The main process of our algorithm is to solve the Schur complement system (3.18) using the GMRES method with an initial guess

$$G^{(0)} = \left\{G_1^{(0)}, G_2^{(0)}, \cdots, G_{n_b}^{(0)}\right\}.$$

We need to derive the right hand side, and explain how to compute the matrix-vector multiplication of the system without explicitly forming the coefficient matrix. The right hand side needs to be computed just once which is described below.

5.1. Computing the right hand side of the Schur complement system. If we take $G = 0$, and apply one step of the immersed interface method to solve Problem (II) to get $U(0)$, then

$$U(0) = A^{-1}\bar{F}.$$

With the knowledge of $U(0)$ and $G = 0$, we can compute the normal derivatives on each side of the interface to get $U_n^{\pm}(0)$ using the approach described in the previous section. Thus the right hand side of the Schur complement system is

$$\begin{aligned} \bar{V} &= PV - EA^{-1}\bar{F} \\ &= PV - EU(0) \\ &= -(\beta^+ U_n^+(0) - \beta^- U_n^-(0) - V). \end{aligned}$$

The last two equalities are obtained from (3.16) and (3.18) with $G = 0$. Now we are able to compute the right hand side of the Schur complement system.

5.2. Computing the matrix-vector multiplication of the Schur complement. Now consider the matrix-vector multiplication

$$(D - EA^{-1}B)Q \tag{5.29}$$

of the Schur complement, where Q is an arbitrary vector of dimension n_b. This involves essentially two steps.

1. A fast Poisson solver for computing

$$U(Q) = A^{-1}(\bar{F} - BQ) \tag{5.30}$$

which is the solution of Problem (II) with $G = Q$, see (3.14).

2. The weighted least squares interpolation to compute $U_n^+(Q)$ and $U_n^-(Q)$. The residual vector in the flux jump condition is

$$R(Q) = V - (\beta^+ U_n^+(Q) - \beta^- U_n^-(Q)) \tag{5.31}$$

which is the same residual vector of the second equation in (3.17) from our definition. In other words, see also (3.16)

$$PV - (DQ + EU(Q)) = R(Q). \tag{5.32}$$

The matrix-vector multiplication (5.29) then can be computed from the last equality of the following derivation:

$$\begin{aligned}(D - EA^{-1}B)\,Q &= DQ - EA^{-1}BQ \\ &= DQ + EU(Q) - EA^{-1}\bar{F} && \text{from (5.30)} \\ &= DQ + EU(Q) - P\,V + \bar{V} \\ &= -R(Q) + \bar{V} && \text{from (5.32).}\end{aligned}$$

Note that from the second line to the third line we have used the following

$$EA^{-1}\bar{F} = P\,V - \bar{V}$$

which is defined in (3.18).

It worth to point out that once our algorithm is successfully terminated, which means that the residual vector is close to the zero vector, we not only have an approximation Q to the solution G^*, an approximation $U(Q)$ to the solution U, bult also approximations $U_n^{\pm}(Q)$ to the normal derivatives from each side of the interface. The normal derivative information is very useful for some moving interface problems where the velocity of the interface depends on the normal derivative of the pressure.

6. Convergence Analysis.

As to this point, we have had a complete algorithm for solving the original elliptic equations of the form Problem (I). We have transformed the original elliptic equation to a corresponding Poisson equation with different source term and jump conditions, or internal boundary conditions, (2.9b) and (2.11). The jump condition (2.11) is Neumann-like boundary condition which involves the normal derivatives from each side of the interface. In our algorithm, the classical five-point difference scheme at regular grid points is used. This discetization is second order accurate. As discussed in Section 4, the Neumann-like internal boundary condition (2.11) is also discretized to second order. So from the analysis in Chapter 6 of [17] on Neumann conditions, we should be able to conclude second order convergence globally for our computed solution, provide that we can solve the Poisson Problem (II) to second order accuracy. This is confirmed in our early work [2, 9]. Numerical experiments have confirmed second order accuracy of the computed solution for numerous test problems, see Section 8.

7. An efficient preconditioner for the Schur complement system.

With the algorithm described in previous sections, we are able to solve Problem I to second order accuracy. In each iteration we need to solve a Poisson equation with a modified right hand side. A fast Poisson solver such as a fast Fourier transformation method (FFT), cyclic reduction, etc. [19], can be used. The number of iterations of the GMRES method depends on the condition number of the Schur complement. If we make use of both (4.21) and (4.28) to compute $U_n^{\pm}$, the condition number seems to be proportional to $1/h$. Therefore the number of iterations will grow linearly as we increase the number of grid points.

Below we propose a modification in the way of computing $U_n^{\pm}$ which seems to improve the condition number of the Schur complement system dramatically.

If u_n^+ and u_n^- are the exact solutions, that is

$$\beta^+ u_n^+ - \beta^- u_n^- - v = 0,$$

then we can solve u_n^- or u_n^+ in terms of v, β^-, β^+, and $[u_n]$. It is easy to get

$$u_n^- = \frac{v - \beta^+[u_n]}{\beta^+ - \beta^-}, \tag{7.33}$$

or

$$u_n^+ = \frac{v - \beta^-[u_n]}{\beta^+ - \beta^-}. \tag{7.34}$$

The idea is simple and intuitive. We use one of the formulas (4.21) or (4.28) obtained from the weighted least squares interpolation to approximate u_n^- or u_n^+, and then use (7.34) or (7.33) to approximate u_n^+ or u_n^- to force the solution to satisfy the flux jump condition. This is actually an acceleration process, or a preconditioner for the Schur complement system (3.18).

With this modification, the number of iterations for solving the Schur complement system seems to be independent of the mesh size h, and almost independent of the jump $[\beta]$ in the coefficient as well, see the next section for more details. Although we have not been able to prove this claim, the algorithm seems to be extraordinary successful.

Whether we use the pair (4.21) and (7.34) or the other (4.28) and (7.33) have only a little affect on the accuracy of the computed solutions and the number of iterations. The algorithm otherwise behaves the same and the analysis in the next section seems to be true no mater what pair we choose.

We have been using the following criteria to choose the desired pair

$$\text{If} \quad \beta^+ < \beta^- : \quad \left\{ \begin{array}{l} U_n^+ \ \text{ is determined by (4.28)} \\ U_n^- = \dfrac{V - \beta^+ G}{\beta^+ - \beta^-}, \end{array} \right.$$

$$\text{If} \quad \beta^+ > \beta^- : \quad \left\{ \begin{array}{l} U_n^- \ \text{ is determined by (4.21)} \\ U_n^+ = \dfrac{V - \beta^- G}{\beta^+ - \beta^-}, \end{array} \right.$$

which seems always better than the choice of the other way around.

8. Numerical Experiments.

We have done many numerical experiments with different interfaces and various test functions. Since our scheme can handle jumps in the solution, we have great flexibility in choosing test problems. From the numerical tests we intend to determine:

- The accuracy of computed solutions. Are they second order accurate?

- The numbers of iterations as we change the mesh size h and the ratio of the discontinuous coefficient, $\rho = \beta^+/\beta^-$.
- The ability of the algorithm to deal with complicated interfaces and large jumps in the coefficient.

All the experiments are computed with double precision. The computational parameters include:

- Computational rectangle $[a, b] \times [c, d]$.
- The number of grid points m and n in the x- and y- directions respectively, we assume that $h = (b-a)/m = (d-c)/n$, where h is the mesh size.
- The number of control points n_b. The interface is expressed in terms of cubic splines passing through the control points.
- The parameter α in the weighted least squares interpolation. We take $\alpha = 2.1\,h$ unless specified differently.

The maximum norm is used to measure the errors in the computed solution U_{ij}, and the normal derivatives $U^-_{n\,p}$ and $U^+_{n\,p}$ from each side of the interface at the p-th control point. The relative errors are defined as follows

$$E_1 = \frac{\max\limits_{i,j} |u(x_i, y_j) - U_{ij}|}{\max\limits_{i,j} |u(x_i, y_j)|}, \tag{8.1}$$

$$E_2 = \frac{\max\limits_{1 \le p \le n_b} \left|u^-_n(\vec{X}_p) - U^-_{n\,p}\right|}{\max\limits_{1 \le p \le n_b} \left|u^-_n(\vec{X}_p)\right|}, \tag{8.2}$$

$$E_3 = \frac{\max\limits_{1 \le p \le n_b} \left|u^+_n(\vec{X}_p) - U^+_{n\,p}\right|}{\max\limits_{1 \le p \le n_b} \left|u^+_n(\vec{X}_p)\right|}, \tag{8.3}$$

where $\vec{X}_p$, $1 \le p \le n_b$ is one of control points on the interface. Each grid point is labeled as either in the inside Ω^- or the outside Ω^+ of the interface and the exact solution is determined accordingly. In other words, the exact solution is not determined from the exact interface but the discrete one. In Table 1, r_i, $i = 1, 2, 3$ is the ratio of successive errors. A ratio of 4 corresponds to second-order accuracy. In Table 1, k is the number of iterations required in solving the Schur complement system (3.18). The ratio of coefficients is defined as $\rho = \beta^-/\beta^+$. In the figures, we use S_i, $i = 1, 2, 3$ to express the slopes of least squares line of experimental data $(\log(h_i), \log(E_i))$.

Example 1. Consider the following interface

$$\begin{cases} X = r(\theta)\cos(\theta) + x_c \\ Y = r(\theta)\sin(\theta) + y_c, \end{cases} \qquad 0 \le \theta < 2\pi,$$

where

$$r(\theta) = r_0 + 0.2\sin(\omega\,\theta), \qquad 0 \le \theta < 2\pi,$$

within the computational domain $-1 \le x,\ y \le 1$. Fig 1 shows some interfaces with different parameters r_0, (x_c, y_c), and ω. Dirichlet boundary conditions, as well as the jump conditions $[u]$ and $[\beta u_n]$ along the interface, are determined from the exact solution

$$u(x,y) = \begin{cases} \dfrac{r^2}{\beta^-} & \text{if } (x,y) \in \Omega^- \\ \dfrac{r^4 + C_0 \log(2r)}{\beta^+} + C_1 \left(\dfrac{r_0^2}{\beta^-} - \dfrac{r_0^4 + C_0 \log(2r_0)}{\beta^+} \right) & \text{if } (x,y) \in \Omega^+, \end{cases}$$

where $r = \sqrt{x^2 + y^2}$. The source term can be determined accordingly:

$$f(x,y) = \begin{cases} \dfrac{4}{\beta^-} & \text{if } (x,y) \in \Omega^- \\ \dfrac{16\, r^2}{\beta^+} & \text{if } (x,y) \in \Omega^+. \end{cases}$$

We provide numerical results for three typical cases below.

Case A. The interface parameters are chosen as $r_0 = 0.5$, $(x_c, y_c) = (0,0)$, and $\omega = 0$. So the interface is a circle centered at the origin, see Fig 2(a). With $C_1 = 1$, the solution is continuous everywhere, but u_n and βu_n are discontinuous across the circle. It is easy to verify that $[\beta u_n] = -0.7$ when we take $C_0 = -0.1$. Fig 3(a) is the plot of the solution $-u$ with $\beta^- = 1$ and $\beta^+ = 10$.

Case B. The interface parameters are chosen as $r_0 = 0.5$, $x_c = y_c = 0.2/\sqrt{20}$, and $\omega = 5$, see Fig 2(a). We shift the center of the interface a little bit to have a non-symmetric solution relative to the grid. We want our test problems to be as general as possible. The interface is irregular but the curvature has modest magnitude. So with a reasonable number of points on the interface, we can express it well. Now it is almost impossible to find an exact solution which is continuous but not smooth across the interface, so we simply set $C_1 = 0$, and $C_0 = -0.1$. Fig 3(b) is the plot of the solution $-u$ with $\beta^- = 1$ and $\beta^+ = 10$.

Case C. The interface parameters are chosen as $r_0 = 0.4$, $x_c = y_c = 0.2/\sqrt{20}$, and $\omega = 12$, see Fig 2(b). Now the magnitude of the curvature is very large at some points on the interface and we have to have enough control points to resolve it. The solution parameters are set to be the same as in Case B.

Fig 4–6 and Table 1 are some plots and data from the computed solutions which we will analyze below.

8.1. Accuracy. Table 1 shows the results of grid refinement analysis for Case A with two very different ratio β^-/β^+, $\beta^+ > \beta^-$. When $\beta^-/\beta^+ = 0.5$, the ratio r_i are very close to 4 indicating second order convergence. With $\beta^- = 1$ and $\beta^+ = 10^4$, the error in the solution drops much more rapidly. This is because the solution in Ω^+ approaches a constant as β^+ becomes large, and it is quadratic in Ω^-. A second order accurate method would give high accurate solution in both regions. So it is not surprising to see the ratio r_1 is much larger than 4. For the normal derivatives, we expect second order accuracy again since $\beta^+ u_n^+$ is not quadratic and has magnitude of $O(1)$. This agrees with the results r_2 and r_3 in Table 1.

In Fig 4 we consider the opposite case when $\beta^-/\beta^+ = 10^4$, $\beta^+ < \beta^-$. In this case the solution is not quadratic so we see the expected second order accuracy. Fig 4(a)

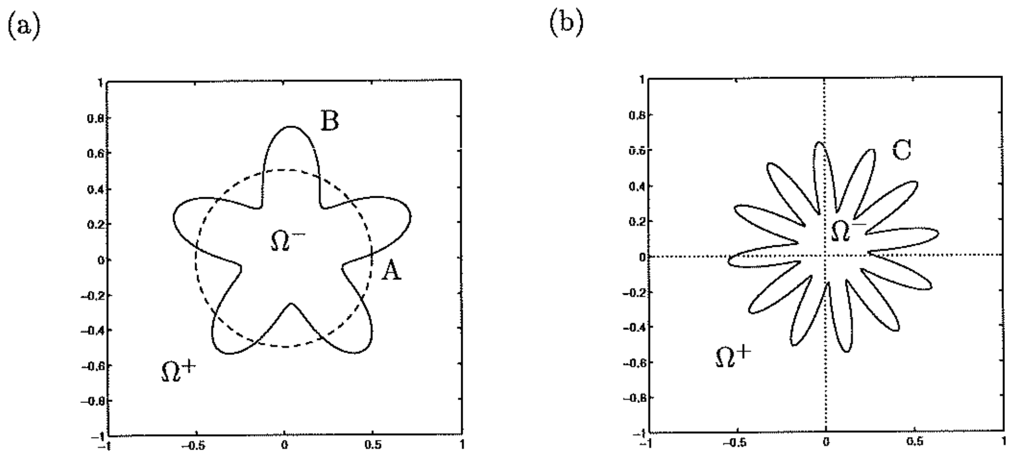

FIG. 2. *Different interfaces of Example 1. (a) Case* **A** *and* **B**. *(b) Case* **C**.

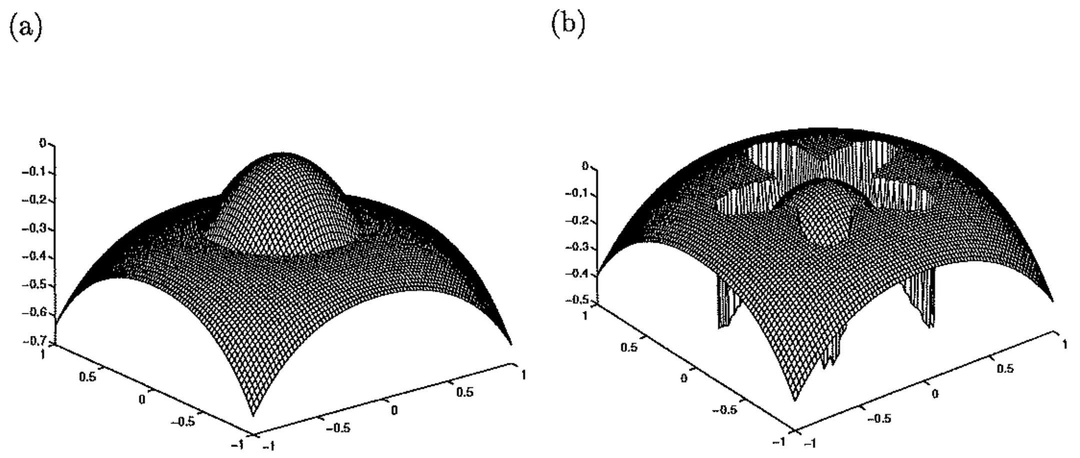

FIG. 3. *The solutions $-u$ of Example 1 with $\beta^+ = 10$, and $\beta^- = 1$. (a) Case* **A**, *a circular interface where the solution is continuous but $[\beta u_n] = -0.7$. (b) Case* **B**, *an irregular interface where both the solution and the flux $[\beta u_n]$ are discontinuous.*

TABLE 1
Numerical results and convergence analysis for Case A with $m = n_b = n$.

n	β^+	β^-	E_1	E_2	E_3	r_1	r_2	r_3	k
40	2	1	$2.285\ 10^{-3}$	$2.23\ 10^{-3}$	$7.434\ 10^{-3}$				7
80	2	1	$5.225\ 10^{-4}$	$5.956\ 10^{-3}$	$1.987\ 10^{-2}$	4.37	3.74	3.74	7
160	2	1	$1.269\ 10^{-4}$	$1.827\ 10^{-4}$	$6.101\ 10^{-4}$	4.12	3.26	3.26	7
320	2	1	$2.988\ 10^{-5}$	$5.038\ 10^{-5}$	$1.678\ 10^{-4}$	4.25	3.63	3.64	7

n	β^+	β^-	E_1	E_2	E_3	r_1	r_2	r_3	k
40	10000	1	$6.552\ 10^{-5}$	$6.331\ 10^{-4}$	$2.110\ 10^{-4}$				8
80	10000	1	$7.847\ 10^{-6}$	$8.366\ 10^{-5}$	$2.785\ 10^{-5}$	8.35	7.57	7.58	8
160	10000	1	$5.988\ 10^{-7}$	$9.192\ 10^{-7}$	$3.033\ 10^{-6}$	13.1	9.10	9.18	8
320	10000	1	$5.859\ 10^{-8}$	$2.058\ 10^{-7}$	$6.887\ 10^{-7}$	10.2	4.47	4.40	7

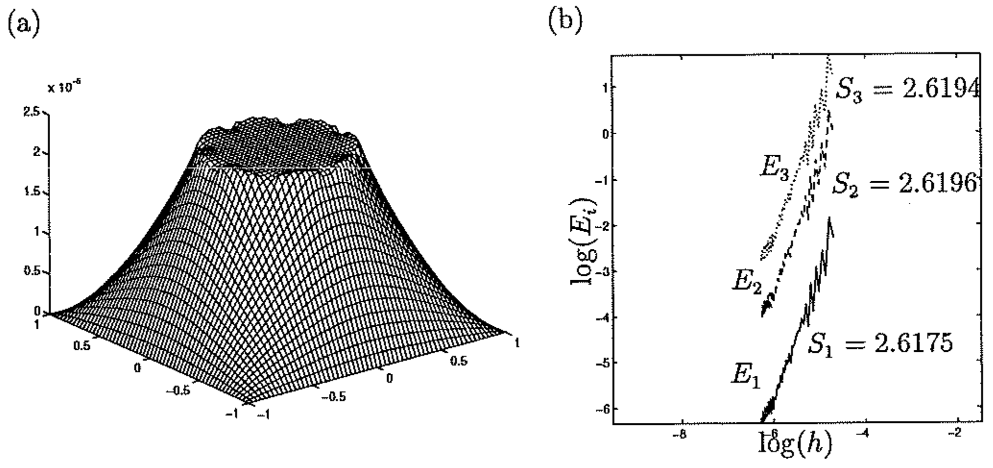

FIG. 4. *(a): Error distribution for Case A. (b): Errors E_i vs the mesh size h in* log-log *scale for Case A with $m = n = n_b$, $\beta^- = 10^4$ and $\beta^+ = 1$.*

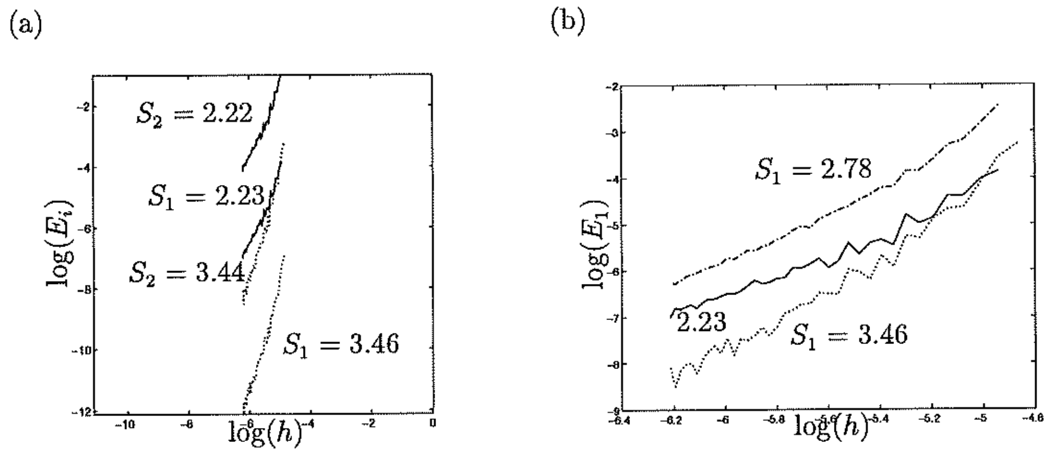

FIG. 5. *Errors E_i vs the mesh size h in* log-log *scale for Case B with $\beta^- = 10^3$ and $\beta^+ = 1$. (a) The solid line: $n_b = 130 + 3\,int\left(\sqrt[3]{(n-130)^2}\right)$. The dotted line: $n_b = n$. (b) The solid line and dotted line are the same as in (a) but on a different scale. The dash-dotted line is the result obtained with $\alpha = 3.6\,h$ and $n_b = n$.*

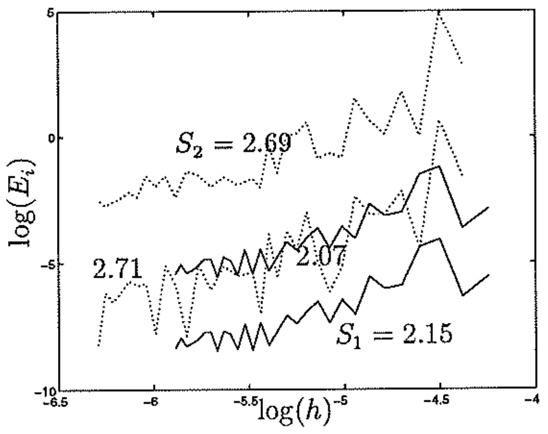

FIG. 6. *Errors E_i vs the mesh size h in* log-log *scale for Case C with $\beta^- = 100$ and $\beta^+ = 1$. The solid line: fixed n_b ($n_b = 520$). The dotted line: $n_b = n$.*

plots the error distribution over the region. The error seems to change continuously even though the maximum error occurs on or near the interface. Usually if the curvature is very big in some part of an interface, for example, near a corner, then we would observe large errors over the neighborhood of that part of the interface.

For interface problems, the errors usually do not decrease monotonously as we refine the grid unless the interface is aligned with one of the axes. We need to study the asymptotic convergence rate which is usually defined as the slope of the least squares line of the experimental data $(\log(h_i), \log(E_i))$. Fig 4(b) plots the errors versus the mesh size h in log-log scale for the case $n_b = n$. The asymptotic convergence rate is about 2.62 compared to 2 for a second order method. As h gets smaller we can see the curves for the errors become flatter indicating the asymptotic convergence rate will approach 2.

The dotted curves in Fig 5 and Fig 6 are the results for case B and C, where the interfaces are more complicated compared to case A. Again we take $m = n = n_b$. The asymptotic convergence rates are far more than two. Such behavior can also be observed from Example 4 in [16]. Does it mean that our method is better than second order? Certainly this is not true from our discretization. Below we explain what is happening.

For interface problems, the errors depend on the solution $u(x,y)$, the mesh size h, the jump in the coefficient $[\beta]$, the interface Γ and its relative position to the grid, and the maximal distance between control points on the interface, h_b. We can write the error in the solution, for example, as follows

$$E_1 = C\left(u, h, h_b, [\beta], \Gamma\right) h^2 + \bar{C}\left(u, h, h_b, [\beta], \Gamma\right) h_b^q. \tag{8.4}$$

The first term in the right hand side of (8.4) is the error from the discretization of the differential equation. The term $C\left(u, h, h_b, [\beta], \Gamma\right)$ has magnitude $O(1)$ but does not necessarily approach to a constant. The second term in the right hand side of (8.4) is the error from the discretization of the interface Γ. If we use a cubic spline interpolation, then $q > 2$. For Case A, the interface is well expressed and the first term in (8.4) is dominant, we have clearly second order convergence. For Case B and C, the interfaces are more complicated and the second term in (8.4) is dominant. That is why we have higher than second order convergence. Eventually, the error in the first term will dominate and we will then observe second order convergence.

To further verify the arguments above, we did some tests with fixed number of control points n_b. For example, we take $n_b = 540$ for Case C, see the solid line in Fig 6. Presumably the interface is expressed well enough and the second term in (8.4) is negligible. We see the slopes of the least squares line of the errors E_1 and E_2 are 2.15 and 2.07 respectively indicating second order convergence. Usually E_2 and E_3, the error in the normal derivatives u_n^- and u_n^+, behaves the same, so we only need to study one of them. If we let n_b change with the same speed as the number of grids m and n, then the second term in (8.4) is dominant and the slopes of the least squares line of the errors E_1 and E_2 are 2.71 and 2.69 respectively. Once n_b is large enough, the first term will dominate in (8.4) and the error will decrease quadratically. This can also be seen roughly from Fig 6. Note that the errors oscillate as n gets large whether we fix n_b or not. But the fluctuation becomes smaller as we refine the grid. The upper envelop of E_1 behaves the same as the least squares line of the experimental data $(\log(h_i), \log(E_i))$. So it is reasonable to use the asymptotic convergence rate to discuss the accuracy when errors do not behave monotonously.

As another test, we let n_b change slower than m and n. The solid lines in Fig 5(a) are obtained with $n_b = 130 + 3\,int\left(\sqrt[3]{(n-130)^2}\right)$, $n \geq 130$. Now we have roughly $h_b^q \approx h^2$ and the errors decrease quadratically with the mesh size h, but not h_b. The slopes of the least squares line of the errors E_1 and E_2 are 2.23 and 2.22 respectively.

We now discuss the effect of the different choice of the parameter α in the least squares approximation described in Section 4 on the solution. Most of the computations are done with $\alpha = 2.1\,h$ except in Fig 5(b), the dash-dotted line where $\alpha = 3.6\,h$. As we can expect, the smaller α is, the higher accuracy in the computed solution because the points involved are clustered together and the error in the Taylor expansion will be smaller. However, the smaller α is, the more oscillatory in the error as we refine the grid. For larger α, the computation cost increases quickly, but the error behaves much smoother with the mesh size h. Usually we can take small α for smooth interfaces, and larger α if we want a smoother error distribution for more complicated interfaces.

8.2. The number of iterations versus the mesh size h. Fig 7(a), also see Fig 10(a) for Example 2, shows the number of iterations versus the number of grid points m and n for case A, B, and C. It is not surprising to see that the number of iterations depends on the shape of the interface. The number of iterations required for Case C is larger than that for Case A and B. But it is wonderful to see that the number of iterations is almost independent of the mesh size h. For Case A, where the interface is a circle, we only need about $5 \sim 8$ iterations for all choices of the mesh size h for two extreme cases $\rho = 10^4$ and $\rho = 10^{-4}$. We will see in the next paragraph that this is also true for different choices of the ratio $\rho = \beta^-/\beta^+$. Note that the number of iterations is about two or three fewer than the numbers of calls of the fast Poisson solver. We need two or three of them for initial set up of the Schur complement system. In Fig 7(a), the lowest curve corresponds to case A with $\beta^- = 1$ and $\beta^+ = 10^4$, the lowest but the second curve corresponds to $\beta^- = 10^4$ and $\beta^+ = 1$. For case B, the number of iterations required is about $17 \sim 21$ for $\rho = 10^{-3}$ and $\rho = 10^3$ respectively. For case C, the most complicated interface, the number of iterations is about 46 with reasonable number of control points on the interface for $\rho = 10^{-2}$ and $\rho = 10^2$ respectively.

8.3. The number of iterations versus the jump ratio $\rho = \beta^-/\beta^+$. Fig 7(b), also see Fig 10(b) for Example 2 , plots the number of iterations versus the jump ratio ρ in *log-log* scale with fixed number of grids $m = n = n_b = 160$. As ρ goes away from the unit we have larger jump relatively in the coefficient. The number of iterations increases proportional to $|log(\rho)|$ when ρ is small but soon reaches a point after which the number of iterations will remain as a constant. Such points depends on the shape of the interface. For Case A, it requires only about $5 \sim 6$ iterations at the most for $\rho < 1$ and about $7 \sim 8$ iterations for $\rho > 1$ in solving the Schur complement system using the GMRES method. For Case B, the numbers are about $17 \sim 22$. For Case C, the most complicated interface in our examples, the numbers are about $47 \sim 69$. As we mentioned in the previous paragraph, also see Fig 7(a), for Case C, with only 160 control points we can not express the complicated interface Fig 2(b) very well. If we take more control points on the interface, then the number of iterations will be about 46.

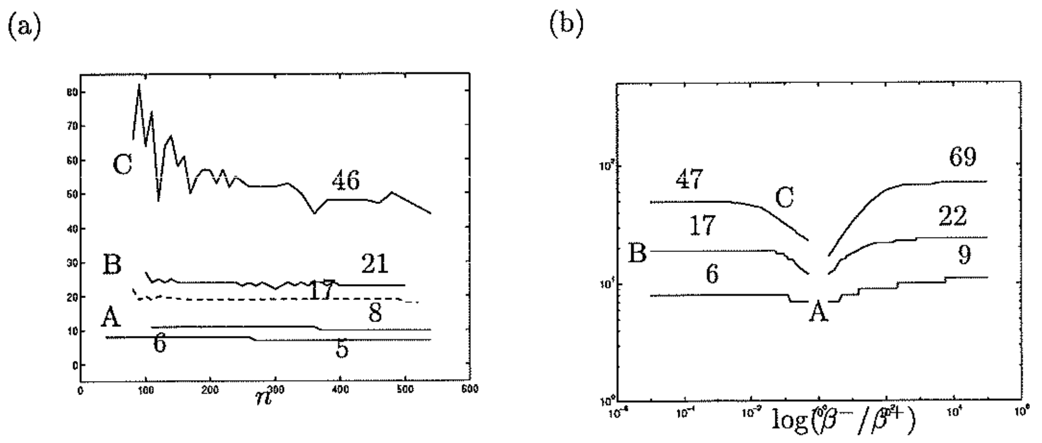

FIG. 7. *The number of iterations for Example 1 with $m = n = n_b$. (a) : vs the number of grids n. Case A: lower curve, $\beta^- = 1$, $\beta^+ = 10^4$; upper curve, $\beta^- = 10^4$, $\beta^+ = 1$. Case B: lower curve, $\beta^- = 1$, $\beta^+ = 10^3$; upper curve, $\beta^- = 10^3$, $\beta^+ = 1$. Case C: $\beta^- = 1$, $\beta^+ = 100$. (b) : vs the ratio of jumps β^-/β^+ in* log-log *scale with $m = n = n_b = 160$. We set $\beta^- = 1$ w hen $\beta^-/\beta^+ < 1$ and $\beta^+ = 1$ when $\beta^-/\beta^+ > 1$.*

Example 2. This geometry of this example is adapted from Problem 3 of [1]. The solution domain is the $[-1, 1] \times [0, 3]$ rectangle and the interface is determined by

$$\begin{cases} X = 0.6\cos(\theta) - 0.3\cos(3\,\theta) + x_c \\ Y = 0.7\sin(\theta) - 0.07\sin(3\,\theta) + y_c, \end{cases} \qquad 0 \le \theta < 2\pi,$$

Fig 8(a) show the solution domain and the interface Γ with $x_c = 0$ and $y_c = 1.5$. Again Dirichlet boundary condition, as well as the jump conditions $[u]$ and $[\beta u_n]$ are determined from the exact solution

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

The source term can be determined accordingly:

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

Fig 8(b) is a plot of the computed solution.

This example is different from Example 1 in several ways. The solution is independent of the coefficient β. But the magnitude of the jump $[\beta u_n]$ and the source term $[f]$, increase with the magnitude of the jump $[\beta]$. However we have observed similar behaviors in the numerical results as we discussed in Example 1. Example 1 and Example 2 are two extreme samples of elliptic interface problems. So we should be able to get some insights about the method proposed in this paper.

Fig 9 shows errors E_i versus mesh size h in *log-log* scale with different choice of n_b. In Fig 9(a), $\beta^- = 1$, $\beta^+ = 10^3$. The solid lines correspond to a fixed discretization of the interface, $n_b = 420$. As we expected, the asymptotic convergence rate for E_i are $S_1 = 2.1404$, $S_2 = 2.1268$, and $S_3 = 2.1272$. They are all close to 2 indicating

(a)

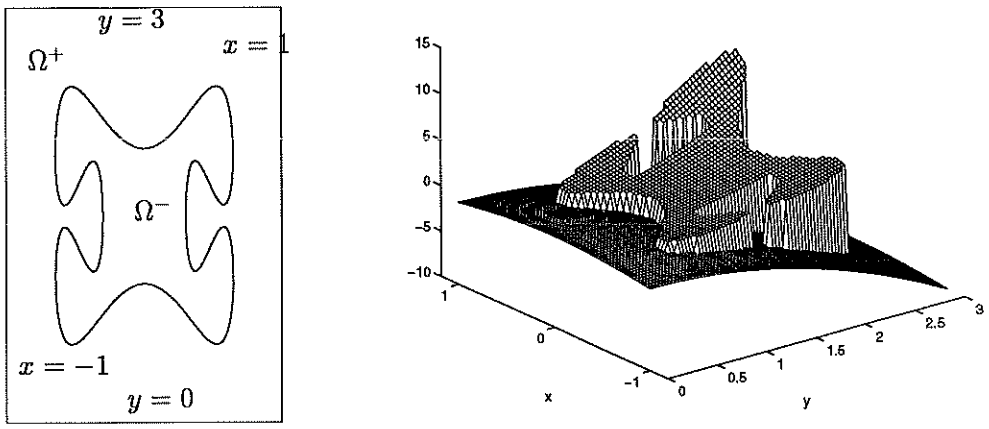

FIG. 8. *(a) The interface of Example 2. (b) The solution of Example 2.*

second order accuracy. The dotted line in Fig 9(a) correspond to a variable n_b which changes in the same rate as the number of grid point m in x-direction. The asymptotic convergence rate of E_i for are $S_1 = 4.0703$, $S_2 = 3.3413$, and $S_3 = 3.3473$. They are $S_1 = 3.2044$, $S_2 = 3.1790$, and $S_3 = 3.1897$ for $\beta^- = 10^3$ and $\beta^+ = 1$ in Fig 9(b). These numbers are all larger than 2 similar to the cases we saw in Fig 5, and Fig 6. We have explained such phenomena already.

Fig 10(a) plots the number of iteration versus the number of grids n with $m = n_b = 2n/3$. Again we consider two extreme cases, $\rho = 10^{-3}$ with $\beta^- = 1$, and $\rho = 10^3$ with $\beta^+ = 1$. Once the interface is well expressed somewhere after $n > 180$, the number of iteration will slightly decrease to a constant which is about 28 for $\rho = 10^{-3}$ and 34 for $\rho = 10^3$. Fig 10(b) plots the number of iteration versus the ratio ρ with fixed grid $m = n_b = 160, n = 240$. We set $\beta^- = 1$ when $\rho < 1$ and $\beta^+ = 1$ when $\rho > 1$. We observe the same behavior as in Fig 7(b). Initially the number of iterations increases proportional to $|log(\rho)|$ as ρ goes away from the unit, but it soon approaches a constant which is about 28 for $\rho < 1$ and 34 for $\rho > 1$.

(a) (b)

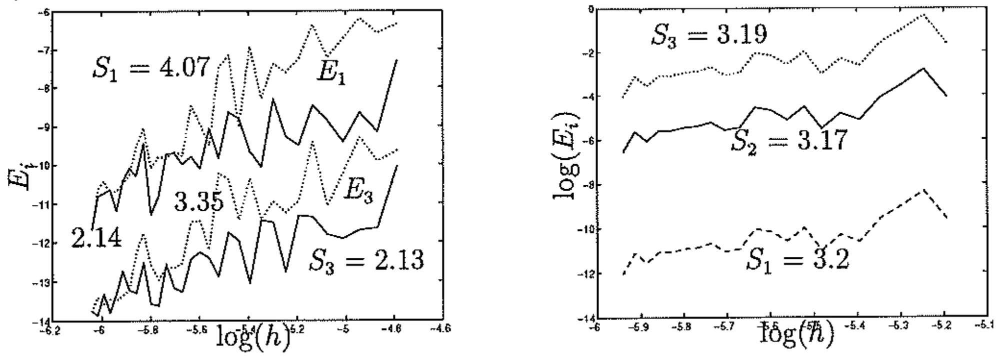

FIG. 9. *Errors E_i vs the mesh size h in* log-log *scale for Example 2 with $m = 2n/3$. (a) $\beta^- = 10^3$ and $\beta^+ = 1$. The solid line: fixed n_b, $n_b = 420$; The dotted line: $n_b = m$. (b) $\beta^- = 1$ and $\beta^+ = 10^3$. Error plot of E_i with $n_b = m$.*

Summary of the numerical experiments. In our computations, the largest error usually occurs at those points which are close to the part of the interface which

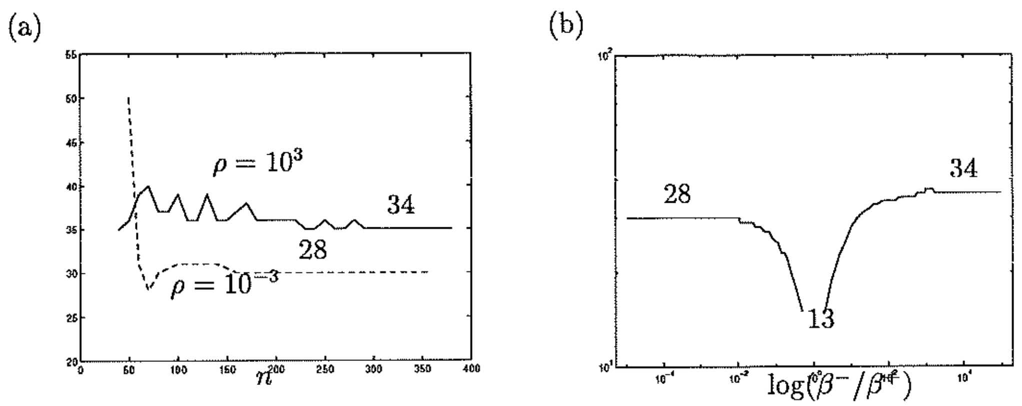

FIG. 10. *The number of iterations for Example 2 with $m = n_b = 2n/3$. (a) : vs the number of grids n. Lower curve: $\beta^- = 1$ and $\beta^+ = 10^3$; Upper curve: $\beta^- = 10^3$ and $\beta^+ = 1$. (b) : vs the ratio of jumps β^-/β^+ in* log-log *scale with fixed grid $m = n_b = 160$ and $n = 240$. We set $\beta^- = 1$ when $\beta^-/\beta^+ < 1$ and $\beta^+ = 1$ when $\beta^-/\beta^+ > 1$.*

has large curvature. Depending on the shape of the interface, we should take enough control points on the interface so the error in expressing the interface does not dominate the global error. However, once such a critical number is decided, we do not need to double it as we double the number of grid points, which saves some computational cost. We should still be able to maintain second order accuracy. The number of iteration for solving the Schur complement system using a GMRES method is almost independent of both the mesh size h as well as the jump in the coefficient.

9. Conclusions.

We have developed a second-order accurate fast algorithm for a type of elliptic interface problems with large jumps in the coefficient across some irregular interface. We precondition the original partial differential equation to obtain an equivalent Poisson problem with different source terms and a Neumann-like interface condition. The fast Poisson solver proposed in [2, 9] can be employed to solve the Schur complement system for the intermediate unknown, the jump in the normal derivative along the interface. Then we proposed a preconditioning technique for the Schur complement system which seems to be very successful. Numerical tests revealed that the number of iterations in solving the Schur complement system is independent of both the mesh size h and the jump in the coefficient, though we have not proved this strictly in theory. The idea introduced in this paper might be applicable to other related problem, for example, to domain decomposition techniques. A new least squares approach to approximate interface quantities from a grid function is also proposed. By analyzing the numerical experiments, we have discussed some issues in error analysis involving interfaces.

There is still a lot of room for improving the method described in this paper. For example, we have used cubic spline interpolations for closed interfaces. There are some advantages of this approach. But large errors can occur at the connection of the first and the last control points when we try to make the curve closed. That might also be one of reasons why the error does not decrease monotonously. As an alternative, a level set formulation is under investigation.

The next project following this paper is to study the case with variable coefficients.

We can rewrite (1.1) either as (2.8a) or

$$\begin{aligned} \nabla \left(\frac{\beta}{\beta_\Gamma^-} \nabla u \right) &= \frac{f}{\beta_\Gamma^-} \quad \text{if } x \in \Omega^-, \\ \nabla \left(\frac{\beta}{\beta_\Gamma^+} \nabla u \right) &= \frac{f}{\beta_\Gamma^+} \quad \text{if } x \in \Omega^+, \end{aligned} \tag{9.5}$$

where β_Γ^- and β_Γ^+ are the averages of the coefficients β from each side of the interface. Whether (2.8a) or (9.5) is used, we shall still introduce an intermediate unknown, the jump in the normal derivative across the interface if the jump condition is given in the form of $[\beta u_n]$. In this way, the coefficients of the difference scheme would be very close to those obtained form the classical five-point stencil. We can not take advantage of the fast Poisson solvers for variable coefficient anymore, but we can make use of the multi-grid method developed by L. Adams in [1].

10. Acknowledgments. It is my pleasure to acknowledge the encouragements and advice from various people including Prof. Randy LeVeque, Stanley Osher, Tony Chan, Loyce Adams, Jun Zou and Barry Merriman. Thanks also to Prof. Yousef Saad and Dr. Victor Eijkhout for helping me to implement and understand the GMRES method.

REFERENCES

[1] L. M. Adams. A multigrid algorithm for immersed interface problems. Proceedings of Copper Mountain Multigrid Conference, to appear, 1995.

[2] R. J. LeVeque and Z. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM J. Num. Anal.*, 31:1019–1044, 1994.

[3] R. J. LeVeque and Z. Li. Simulation of bubbles in creeping flow using the immersed interface method. in Proc. sixth international symposium on computational fluid dynamics, Lake Tahoe, Nevada, in press, 1995.

[4] R. J. LeVeque and C. Zhang. Immersed interface methods for wave equations with discontinuous coefficients. To appear, 1994.

[5] R. J. LeVeque and C. Zhang. Finite difference methods for wave equations with discontinuous coefficients. in Proc. 1995 ASCE Conference, S. Sture, ed., to appear, 1995.

[6] R.J. LeVeque and Z. Li. Immersed interface method for Stokes flow with elastic boundaries or surface tension. Tech. Report #95-01, University of Washington, submitted to *SIAM J. Sci. Stat. Compt.*, 1995.

[7] Z. Li. A note on immersed interface methods for three dimensional elliptic equations. *Computers and Mathematics with Applications*, in press.

[8] Z. Li. Uniform treatment of linear systems – algorithm and numerical stability. *Numer. Comput. Appl.*, 2:71–86, 1989.

[9] Z. Li. *The Immersed Interface Method — A Numerical Approach for Partial Differential Equations with Interfaces.* PhD thesis, University of Washington, 1994.

[10] Z. Li. Immersed interface method for moving interface problems. UCLA CAM Report #95-25, submitted to *Numerical Algorithms*, 1995.

[11] Z. Li and A. Mayo. ADI methods for heat equations with discontinuties along an arbitrary interface. In *Proc. Symp. Appl. Math.* W. Gautschi, editor, volume 48. AMS, 1993.

[12] A. Mayo. On the rapid evaluation of heat potentials on general regions. IBM Technical report 14305.

[13] A. Mayo. The fast solution of Poisson's and the biharmonic equations on irregular regions. *SIAM J. Num. Anal.*, 21:285–299, 1984.

[14] A. Mayo. The rapid evaluation of volume integrals of potential theory on general regions. *J. Comput. Phys.*, 100:236–245, 1992.

[15] A. Mayo and A. Greenbaum. Fast parallel iterative solution of Poisson's and the biharmonic equations on irregular regions. *SIAM J. Sci. Stat. Comput.*, 13:101–118, 1992.

[16] A. McKenney, L. Greengard, and Anita Mayo. A fast poisson solver for complex geometries. preprint, 1994.

[17] K. W. Morton and D. F. Mayers. *Numerical Solution of Partial Differential Equations.* Cambridge press, 1995.

[18] Y. Saad. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[19] Paul N. Swarztrauber. Fast Poisson solver. In *Studies in Numerical Analysis,* G. H. Golub, editor, volume 24, pages 319–370. MAA, 1984.

[20] A. Wiegmann and K. Bube. Computing some inverse problems. private communications.