Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 3721258, 12 pages
https://doi.org/10.1155/2018/3721258

Research Article

A Numerical Method for Solving Elliptic Interface Problems Using Petrov-Galerkin Formulation with Adaptive Refinement

Liqun Wang,[1] **Songming Hou**,[2] **and Liwei Shi**[3]

[1]*Department of Mathematics, College of Science, China University of Petroleum (Beijing), 102249, China*
[2]*Department of Mathematics and Statistics and Center of Applied Physics, Louisiana Tech University, Ruston, LA 71272, USA*
[3]*Department of Science and Technology Teaching, China University of Political Science and Law, Beijing, China*

Correspondence should be addressed to Liwei Shi; sliweihmily@gmail.com

Received 29 May 2018; Accepted 30 July 2018; Published 10 September 2018

Academic Editor: Nunzio Salerno

Elliptic interface problems have wide applications in engineering and science. Non-body-fitted grid has the advantage of saving the cost of mesh generation. In this paper, we propose a Petrov-Galerkin formulation using non-body-fitted grid for solving elliptic interface problems. In this method, adaptive mesh refinement is employed for cells with large errors. The new mesh still has all triangles being right triangles of the same shape. Numerical experiments show side-by-side comparison that to obtain the same accuracy, our new method has much less overall CPU time compared with the previous method even with some cost on mesh generation.

1. Introduction

Elliptic interface problems have wide applications in a variety of disciplines, such as electromagnetism, fluid dynamics, and material science. There are two types of grids used for solving such problems: the body-fitted grid and non-body-fitted grid. For the body-fitted grid, mesh generation has more computational cost than the non-body-fitted grid. The quality of the triangles is also an issue that needs special care. We focus on the discussion using non-body-fitted grid.

In the past three decades, much attention has been paid to the numerical solution of elliptic interface problems on non-body-fitted regular Cartesian grids since the pioneering work of Peskin [1] on the first-order accurate immersed boundary method. Motivated by the immersed boundary method, to improve accuracy, in [2], the immersed interface method (IIM) was presented. The method achieves second-order accuracy by incorporating the interface conditions into the finite difference stencil in a way that preserves the interface conditions in both solution and its flux in the normal direction, $[u] \neq 0$ and $[\beta u_n] \neq 0$. The corresponding linear system is sparse but may not be symmetric or positive definite if there is a jump in the coefficient.

Naturally, the standard finite element method has the property of symmetricity and positive definiteness. However, it used body-fitted grid. Can finite element method be applied using non-body-fitted grid? In [3], the immersed finite element (IFE) method is introduced for interface problems with homogeneous jump conditions. The idea is that the test and trial function basis are the same and are continuous but not smooth across the interface in an interface triangle. In [4], the method is generalized to deal with nonhomogeneous flux jump condition. In [5], the partially penalized IFE method is introduced to penalize the discontinuity of the IFE function on neighboring interface triangles. There are some work in the IFE formulation in three dimensions as well, such as [6, 7].

As mentioned above, special treatment is required to use the IFE on non-homogeneous jump conditions. Is it possible for a finite element method to handle non-homogeneous jump conditions the same way as homogeneous jump conditions? In [8], a non-traditional finite element formulation for solving elliptic equations with smooth or sharp-edged interfaces was proposed with non-body-fitting grids for $[u] \neq 0$ and $[\beta u_n] \neq 0$. It achieved second-order accuracy in the L^∞ norm for smooth interfaces and about 0.8th order for sharp-edged interfaces. In [9, 10], the method is analyzed and implemented in three dimensions. The resulting linear system is non-symmetric but positive definite.

In [11], the matched interface and boundary (MIB) method was proposed to solve elliptic equations with smooth interfaces. In [12], the MIB method was generalized to treat sharp-edged interfaces. With an elegant treatment, second-order accuracy was achieved in the L^∞ norm. In [13], the MIB method was extended to three-dimensional interface problems. Also, there has been a large body of work from the finite volume perspective for developing high order methods for elliptic equations in complex domains, such as [14, 15] for two-dimensional problems and [16] for three-dimensional problems. Another class of methods is the Boundary Condition Capturing Method [17–19]. In [20, 21], gradient recovery techniques were developed to improve the accuracy of gradient computation.

In this paper, based on our early work on non-traditional finite element method using a non-symmetric weak formulation with uniform Cartesian grid, we improve the performance by using adaptive mesh refinement on the cells where the numerical error is large. The main idea is as follows: first, we use two different uniform Cartesian coarse grids so that we could compare their results to find at which cells the error is larger. Then, we do adaptive mesh refinement for these cells. Finally, we use our non-symmetric weak formulation on this non-uniform grid. For most problems the large errors occur only around the interface and therefore an alternative method refining around the interface is also proposed. We do present an example in which large errors occur away from the interface though. All triangles are right triangles of the same shape in the new grid. This grid is just a minor modification from the uniform Cartesian grid and the mesh generation cost is very low. However, since smaller triangles are used where the error is large, in the end the L^∞ error is reduced. Numerical results show that to obtain the same L-infinity error, this new method needs much less overall CPU time compared with the previous method.

2. Equations and Weak Formulation

In this section, we briefly go through the equations and weak formulation in our previous work [8]. This is the foundation of our adaptive refinement method in this paper. Before adaptive refinement, the only difference between our previous work and the work in this paper is that the uniform triangular mesh is different.

Consider an open bounded domain $\Omega \subset R^2$. Let Γ be an interface of co-dimension 1, which divides Ω into two disjoint open subdomains, Ω^- and Ω^+, hence $\Omega = \Omega^- \cup \Omega^+ \cup \Gamma$.

We seek solutions of the variable coefficient elliptic equation away from the interface Γ given by

$$-\nabla \cdot (\beta(\mathbf{x}) \nabla u(\mathbf{x})) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega \setminus \Gamma, \tag{1}$$

in which $\mathbf{x} = (x_1, x_2)$ denotes the spatial variables. The coefficient $\beta(\mathbf{x})$ is assumed to be a 2×2 matrix that is uniformly elliptic on each disjoint subdomain, Ω^- and Ω^+, and its components are continuously differentiable on each disjoint subdomain, but they may be discontinuous across the interface Γ. The right-hand side $f(\mathbf{x})$ is assumed to lie in $L^2(\Omega)$.

Given functions a and b along the interface Γ, we prescribe the jump conditions

$$\begin{aligned}[u]_\Gamma(\mathbf{x}) &\equiv u^+(\mathbf{x}) - u^-(x) = a(\mathbf{x}) \\ [(\beta\nabla u)\cdot \mathbf{n}]_\Gamma(\mathbf{x}) &\equiv \mathbf{n}\cdot(\beta^+(\mathbf{x})\nabla u^+(\mathbf{x})) - \mathbf{n} \\ &\quad \cdot (\beta^-(\mathbf{x})\nabla u^-(\mathbf{x})) = b(\mathbf{x}).\end{aligned} \tag{2}$$

The "$\pm$" superscripts refer to limits taken from within the subdomains $\Omega^\pm$.

Finally, the boundary conditions are given by

$$u(\mathbf{x}) = g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega. \tag{3}$$

The jump conditions are enforced strongly in the local system. Also, the flux jump condition appears in the weak formulation.

We introduce the weak solution by the standard procedure of multiplying (1) by a test function ψ and integration by parts:

$$\int_{\Omega^+} \beta\nabla u \cdot \nabla\psi + \int_{\Omega^-} \beta\nabla u \cdot \nabla\psi = \int_\Omega f\psi - \int_\Gamma b\psi, \tag{4}$$

where ψ is in $H_0^1(\Omega)$. Note that although the test function ψ is the same as in the standard finite element method, the function u is not a linear combination of such basis functions. Instead, the jump conditions are enforced strongly.

3. Numerical Method

In this paper, we restrict ourselves to a rectangular domain $\Omega = (x_{\min}, x_{\max}) \times (y_{\min}, y_{\max})$ in the plane. Given positive integers I and J, set $\Delta x = (x_{\max} - x_{\min})/I$ and $\Delta y = (y_{\max} - y_{\min})/J$. We define a uniform Cartesian grid $(x_i, y_j) = (x_{\min} + i\Delta x, y_{\min} + j\Delta y)$ for $i = 0, \ldots, I$ and $j = 0, \ldots, J$. Each (x_i, y_j) is called a grid point. For the case $i = 0, I$ or $j = 0, J$, a grid point is called a boundary point; otherwise it is called an interior point. The grid size is defined as $h = \max(\Delta x, \Delta y) > 0$.

Two sets of grid functions are needed and they are denoted by

$$H^{1,h} = \left\{\omega^h = \left(\omega_{i,j}\right) : 0 \le i \le I, 0 \le j \le J\right\} \tag{5}$$

and

$$H_0^{1,h} = \left\{\omega^h = \left(\omega_{i,j}\right) \in H^{1,h} : \omega_{i,j} = 0 \texttt{ if } i = 0, I \texttt{ or } j = 0, J\right\} \tag{6}$$

Cut every rectangular region $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$ into four pieces of right triangular regions. Collecting all those triangular regions, we obtain a uniform triangulation $T^h : \bigcup_{K\in T^h} K$; see Figure 1. This is slightly different from the triangulation in our previous work [8] for convenience of adaptive mesh refinement.

We call a cell K an interface cell if its vertices belong to different subdomains; we write $K = K^+ \cup K^-$. K^+ and K^- are

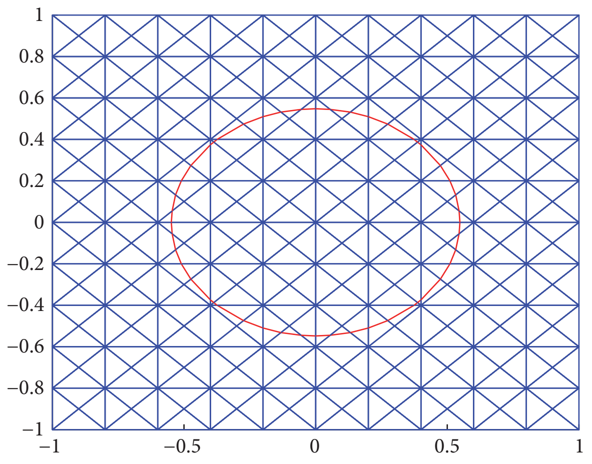


Figure 1: A uniform triangulation.

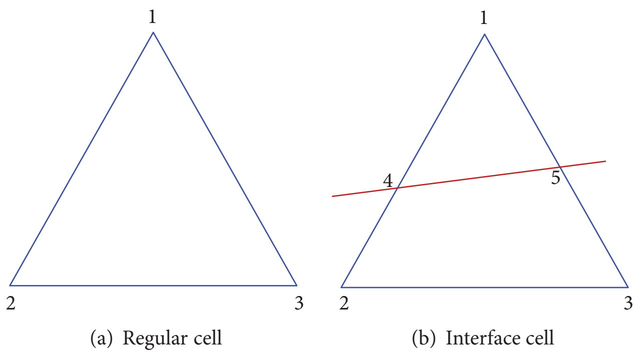


Figure 2: Cells.

separated by a straight interface segment Γ_K^h. We call a cell K a regular cell if all its vertices belong to the same subdomain, either Ω^+ or Ω^-; see Figure 2.

In order to introduce our non-traditional finite element method, two finite element isomorphism mappings are needed from coefficient vector to finite element functions. The first one is $T^h : H^{1,h} \longrightarrow H_0^1(\Omega)$. For any $\psi^h \in H_0^{1,h}$, $T^h(\psi^h)$ is a standard continuous piecewise linear function, which is a linear function in every triangular cell and $T^h(\psi^h)$ matches ψ^h on grid points. The second one U^h is constructed as follows. For any $u^h \in H^{1,h}$ with $u^h = g^h$ at boundary points, $U^h(u^h)$ is a piecewise linear function and matches u^h on grid points. It is a linear function in each regular cell, just like the first extension operator $U^h(u^h) = T^h(u^h)$ in a regular cell. In each interface cell, it consists of two pieces of linear functions; one is on K^+ and the other is on K^-.

In each cell, we shall construct an approximate solution u^h to the interface problem taking into account the jump conditions.

Case 1. If K is a regular cell (see Figure 2(a)), we can have the following equation:

$$\int_K \beta \nabla U^h\left(u^h\right) \cdot \nabla T^h\left(\psi^h\right) = \int_K f T^h\left(\psi^h\right). \tag{7}$$

Case 2. If K is an interface cell (see Figure 2(b)), notice that points $2, 3, 4, 5$ are coplanar, and the value of point 5 can be denoted as a linear combination of the values of points $2, 3, 4$: $u_5 = c_1 u_2 + c_2 u_3 + c_3 u_4$. Then a local system defined on the interface cell K can be constructed as

$$\begin{aligned} [u]_4 &= a_4, \\ [u]_5 &= a_5, \\ c_1 u_2 + c_2 u_3 + c_3 u_4 &= u_5, \\ [(\beta \nabla u) \cdot \mathbf{n}]_6 &= b_6, \end{aligned} \tag{8}$$

where point 6 is the midpoint of the line segment from 4 to 5.

Solve this local system and get the values of $u_4^\pm, u_5^\pm$, which are denoted by the linear combinations of u_1, u_2, u_3. Then we have the following equation:

$$\begin{aligned} &\int_{K^+} \beta \nabla U^h\left(u^h\right) \cdot \nabla T^h\left(\psi^h\right) + \int_{K^-} \beta \nabla U^h\left(u^h\right) \\ &\quad \cdot \nabla T^h\left(\psi^h\right) \\ &= \int_{K^+} f T^h\left(\psi^h\right) + \int_{K^-} f T^h\left(\psi^h\right) - \int_{\Gamma_K^h} b T^h\left(\psi^h\right). \end{aligned} \tag{9}$$

```
Input: Triangulation set $T$, interior point set $P$
Output: New triangulation set $T$, new interior point set $P$
 1: Let $t_n$ be the length of triangulation $T$
 2: for $i = 1$ to $t_n$ do
 3:   if $T_i$'s refine sign is equal to 1 then
 4:     $s \longleftarrow 1, it \longleftarrow 1$
 5:     while s=1 do
 6:       Let $(x, y)$ be the right angle point of $T_i$
 7:       Let $(x_m, y_m)$ be the middle point of the hypotenuse of $T_i$
 8:       Let $T_j$ be the triangle next to $T_i$ and share the hypotenuse with $T_i$
 9:       $(x_n, y_n) \longleftarrow 2(x_m, y_m) - (x, y)$
10:       if $(x_m, y_m)$ is on the boundary then
11:         $s \longleftarrow 0$ (see cell $K_{10,8,13}$ in Figure 3)
12:       else if $(x_n, y_n) \in P$ then
13:         $s \longleftarrow 0$ (see cell $K_{9,6,7}$ in Figure 3)
14:       else
15:         $s \longleftarrow 1, it \longleftarrow it + 1, T_i \longleftarrow T_j$, add $T_i$ into a triangle set $NT$ (see cell $K_{15,4,7}$ in Figure 3)
16:       end if
17:     end while
18:     for $j = it$ down to 1 do
19:       $T_i \longleftarrow NT_j$
20:       Let $(x, y)$ be the right angle point of $T_i$
21:       Let $(x_m, y_m)$ be the middle point of the hypotenuse of $T_i$
22:       $(x_n, y_n) \longleftarrow 2(x_m, y_m) - (x, y)$
23:       if $(x_m, y_m)$ is on the boundary then
24:         Connect $(x, y)$ and $(x_m, y_m)$, separate the original cell into two new cells,
            delete the original cell from $T$ and add two new cells into $T$
25:       else if $(x_n, y_n) \in P$ then
26:         Connect $(x, y)$ and $(x_m, y_m)$, separate the two cells into four new cells,
            delete the two cells from $T$, add four new cells into $T$,
            and add the new point $(x_m, y_m)$ into $P$
27:       end if
28:     end for
29:   end if
30: end for
```

Algorithm 1: Adaptive mesh refinement.

Putting all the cells together, we propose the following method.

Method 1. Find a discrete function $u^h \in H^{1,h}$, such that $u^h = g^h$ on boundary points so that, for all $\psi^h \in H_0^{1,h}$, we have

$$\sum_{K\in T^h}\left(\int_{K^+}\beta\nabla U^h\left(u^h\right)\cdot\nabla T^h\left(\psi^h\right)+\int_{K^-}\beta\nabla U^h\left(u^h\right)\cdot\nabla T^h\left(\psi^h\right)\right)=\sum_{K\in T^h}\left(\int_{K^+}fT^h\left(\psi^h\right)+\int_{K^-}fT^h\left(\psi^h\right)-\int_{\Gamma_K^h}bT^h\left(\psi^h\right)\right). \quad (10)$$

In order to improve the efficiency of our method, we introduce the adaptive refinement method to get a refined mesh. The idea is to use the error scale of the coarse grid to get the refinement triangulation set. First calculate the numerical result of the uniform coarse grids using (n_x, n_y) and $(2n_x, 2n_y)$ grid points in both x and y directions, then use this numerical result to get the numerical error $|e_{(n_x,n_y)}|$ on the corresponding points. Based on the error scale on different grid points, we can decide the refinement scale of each triangulation and then get the refined mesh with our adaptive refinement method; we call this mesh adaptive mesh. The detailed algorithm is given in Algorithm 1.

Remark. When it comes to numerical computation of interface problems, the major part of the error is from the region around the interface, so the refined mesh around this region needs to be the smallest triangulation. This is why we propose the adaptive mesh with further refinement on the interface, called adaptive interface mesh.

4. Numerical Experiments

In all numerical experiments below, the level-set function $\phi(x, y)$, the coefficients $\beta^{\pm}(x, y)$, and the solutions

$$u = \begin{cases} u^+(x, y), & \text{in } \Omega^+ \\ u^-(x, y), & \text{in } \Omega^- \end{cases} \quad (11)$$

Table 1: Numerical results of Example 2 with different mesh.

Mesh	N	$\|e_u\|_\infty$	CPU Time
Uniform Mesh	404101	1.35e-3	6392.5s
Adaptive Mesh	147664	1.32e-3	2218.3s
Adaptive Interface Mesh	18249	1.28e-3	120.8s

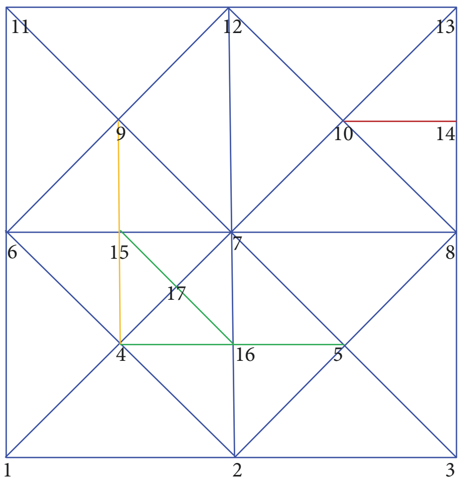


Figure 3: Refined mesh.

are given. Hence

$$f = -\nabla \cdot (\beta \nabla u), \quad a = u^+ - u^-, \quad b = (\beta^+ \nabla u^+) \cdot \mathbf{n} - (\beta^- \nabla u^-) \cdot \mathbf{n} \tag{12}$$

on the whole domain Ω. g is obtained as a proper Dirichlet boundary condition, since the solutions are given.

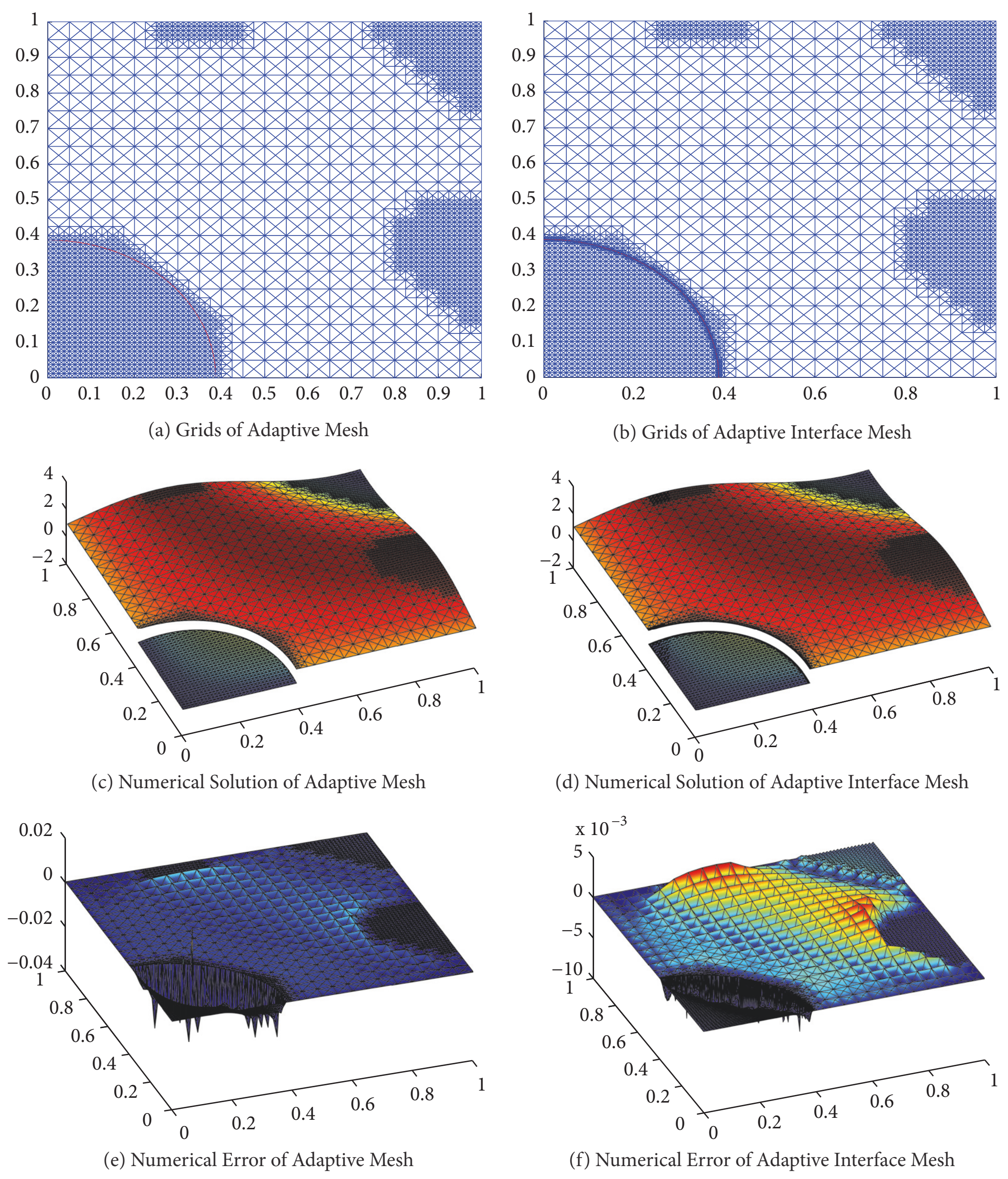
In Examples 2, 4, and 5, we solve the problem in the rectangular domain $\Omega = [-1,1] \times [-1,1]$. In Example 3, we solve the problem in the rectangular domain $\Omega = [0,1] \times [0,1]$. N is the number of interior points, and $N^{1/2}$ is the number of points in one dimension. The L^∞ norm and the energy norm in the whole domain Ω are measured in the following ways:

$$\|e_u\|_\infty = \max_{x\in\Omega} \left|u - u^h\right|,$$
$$\|e_u\|_\beta = \sqrt{\int_\Omega (\bigtriangledown (u-u^h))^T \beta (\bigtriangledown (u-u^h))} = \sqrt{\int_{\Omega^+} (\bigtriangledown (u-u^h))^T \beta (\bigtriangledown (u-u^h)) + \int_{\Omega^-} (\bigtriangledown (u-u^h))^T \beta (\bigtriangledown (u-u^h))}. \tag{13}$$

Example 2. The level-set function ϕ, the coefficients $\beta^\pm$, and the solution $u^\pm$ are given as follows:

$$\phi(x,y) = x^2 + y^2 - 0.15,$$
$$\beta^+(x,y) = 1000 \begin{pmatrix} x^2+3 & \cos(x) \\ \cos(x) & y^2+2 \end{pmatrix},$$
$$\beta^-(x,y) = \begin{pmatrix} x^2+3 & \cos(x) \\ \cos(x) & y^2+2 \end{pmatrix}, \tag{14}$$
$$u^+(x,y) = \sin(5xy) + 1,$$
$$u^-(x,y) = \sin(10xy).$$

Table 1 shows the error and CPU time on uniform triangular mesh, adaptive mesh, and adaptive interface mesh. From this table we can see that the adaptive mesh method has a better result than the uniform mesh method, and the method with adaptive interface mesh is significantly faster and more accurate than the other two methods. The left and right of Figure 4 are a comparison of the adaptive mesh, the numerical solution, and the numerical error using the adaptive mesh method with 3663 interior points (left) and the adaptive interface mesh method with 4167 interior points (right). The error plots do not include the outside boundary points where the solution is given with no error. From Figure 4(e) we can see that the maximum error comes from the area around the interface. Therefore we refine this place



(a) Grids of Adaptive Mesh

(b) Grids of Adaptive Interface Mesh

(c) Numerical Solution of Adaptive Mesh

(d) Numerical Solution of Adaptive Interface Mesh

(e) Numerical Error of Adaptive Mesh

(f) Numerical Error of Adaptive Interface Mesh

FIGURE 4: Numerical results of Example 2.

with the smallest triangulation. Figure 4(f) is the numerical error after the refinement; it is clear in this figure that the error around the interface has been decreased to the same scale as the error away from the interface.

Example 3. The level-set function ϕ, the coefficients $\beta^{\pm}$, and the solution $u^{\pm}$ are given as follows:

$$\phi(x,y) = x + 0.5,$$

$$\beta^{+}(x,y) = \begin{pmatrix} x^2+1 & 0 \\ 0 & y^2+1 \end{pmatrix},$$

$$\beta^{-}(x,y) = 1000\begin{pmatrix} x^2+1 & 0 \\ 0 & y^2+1 \end{pmatrix},$$

$$u^{+}(x,y) = x^4 + x^3 - 3x^2 + 2y, \quad x > 0.5$$

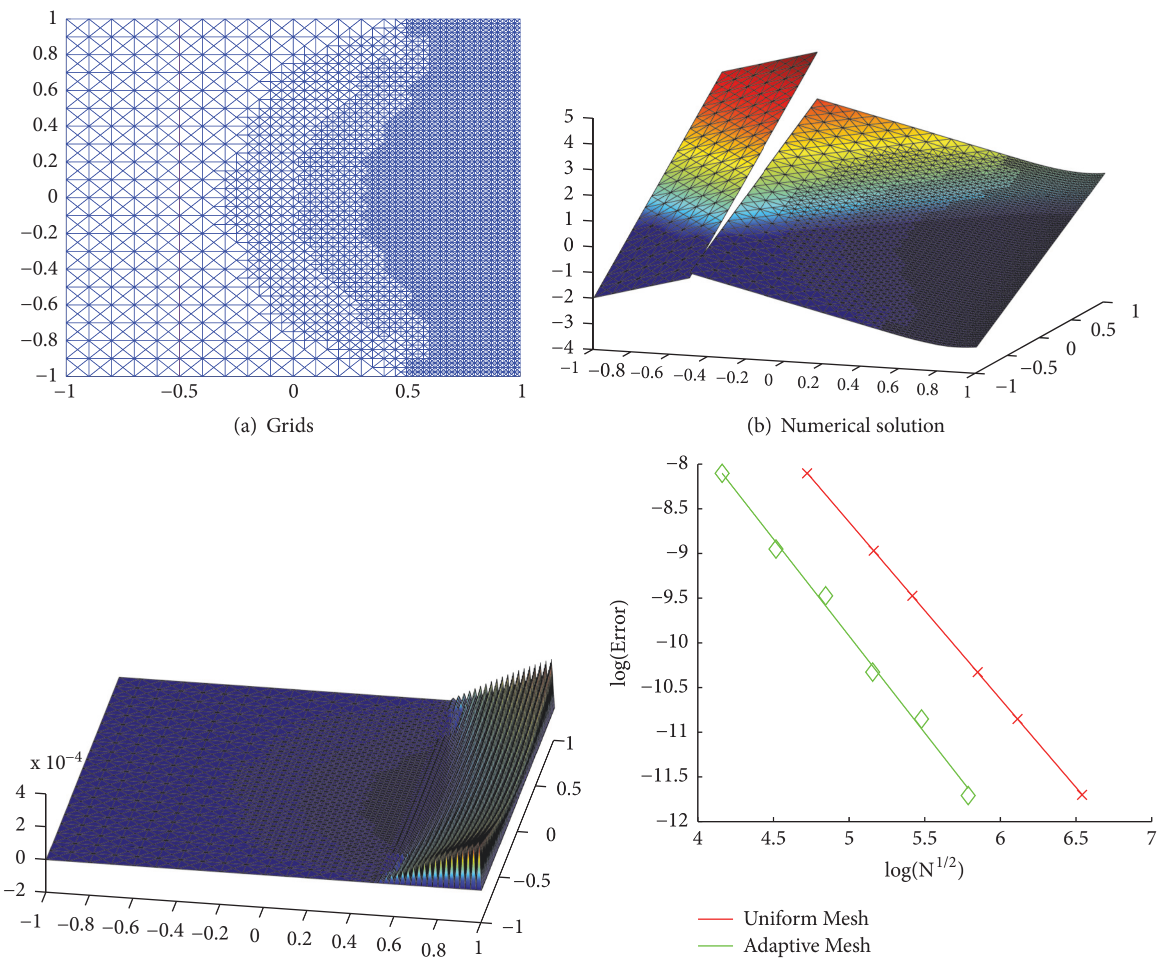
$$u^{+}(x,y) = -\frac{7}{4}(x-0.5) + 2y - \frac{9}{16}, \quad x \le 0.5$$

$$u^{-}(x,y) = 2x + 3y + 3. \tag{15}$$

The left and right of Table 2 show the error on uniform triangular meshes and adaptive meshes with different grids, respectively. From this table we can see that the method with adaptive mesh is much faster than the method with uniform mesh. We plot the adaptive mesh using the adaptive mesh method with 5004 interior points in Figure 5(a). The numerical solution and numerical error are shown in (b) and

Table 2: Numerical results of Example 3 with different mesh.

	Uniform Mesh			Adaptive Mesh	
N	$\|e_u\|_\infty$	CPU Time	N	$\|e_u\|_\infty$	CPU Time
12641	3.02e-4	27.0s	4109	3.02e-4	21.3s
30505	1.27e-4	84.4s	8377	1.30e-4	37.4s
50881	7.69e-5	179.5s	16171	7.69e-5	73.0s
120541	3.27e-5	723.8s	30096	3.27e-5	155.7s
204161	1.94e-5	1789.4s	57391	1.94e-5	436.8s
479221	8.29e-6	18119.7s	106387	8.22e-6	1206.8s
	Order: 1.98			Order: 2.17	



(a) Grids

(b) Numerical solution

(c) Numerical error

(d) L^∞ error for different grids

Figure 5: Numerical results of adaptive mesh for Example 3.

(c) of Figure 5. After calculating the coarse grid error $|e_{(n_x,n_y)}|$, we noticed that the error around the interface is smaller than other areas away from the interface, so in this example we do not need to use the adaptive interface method to get a further refinement mesh around the interface, as we did in Example 2. In Figure 5(c), it is clear that the maximum error lies on the line $x = 1$, and from Figure 5(a) we can see that our method refines the grid around this line automatically and the smallest triangulation lies exactly on $x = 1$. Figure 5(d) is a comparison of the L^∞ error on uniform meshes and adaptive meshes with different grid, which shows that the convergency rate of the adaptive mesh is higher than the uniform mesh and that the adaptive mesh can get to higher than second-order accuracy in L^∞ norm even when the coefficients are matrix and the ratio $\beta^-/\beta^+ = 1000$, which is more efficient than the uniform mesh.

Table 3: Numerical results of Example 4 with different mesh.

Mesh	N	$\|e_u\|_\beta$	$\|e_u\|_\infty$	CPU Time
Adaptive IFEM Mesh in [22]	4021	4.11e-2		
Adaptive Mesh	933	4.41e-2	1.32e-3	15.3s
	3417	1.32e-2	3.75e-4	22.0s
Adaptive Interface Mesh	2017	1.28e-2	3.84e-4	17.6s
	3453	8.86e-3	1.65e-4	26.0s

Table 4: Numerical results of Example 5 with different mesh.

Mesh	N	$\|e_u\|_\infty$	CPU Time	Order
Uniform Mesh	1105	6.65e-3	2.6s	1.58
	8065	1.59e-3	18.5s	
	28561	5.61e-4	86.5s	
	99905	1.88e-4	561.3s	
	400513	6.59e-5	6448.1s	
Adaptive Mesh	567	6.13e-3	3.6s	1.62
	2385	1.56e-3	14.3s	
	9711	5.39e-4	66.2s	
	39038	1.82e-4	346.6s	
	155869	6.17e-5	3329.8s	
Adaptive Interface Mesh	874	5.76e-3	5.8s	1.86
	3037	1.42e-3	18.4s	
	11035	4.63e-4	72.0s	
	41714	1.35e-4	390.1s	
	161208	4.25e-5	3343.4s	

Example 4. This example is taken from [22]. We use this example to compare our method with the adaptive immersed finite element method. The level-set function ϕ, the coefficients $\beta^\pm$, and the solution $u^\pm$ are given as follows:

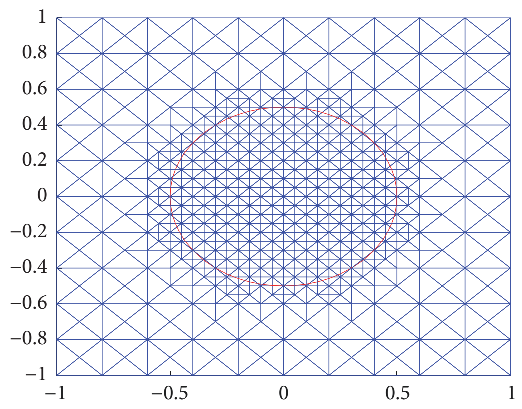
$$\begin{aligned}\phi(x,y) &= x^2+y^2-0.25,\\ \beta^+(x,y) &= 1000,\\ \beta^-(x,y) &= 1,\\ r(x,y) &= \sqrt{x^2+y^2},\\ u^+(x,y) &= \frac{r^3}{1000}+\left(1-\frac{1}{1000}\right)0.5^3,\\ u^-(x,y) &= r^3.\end{aligned} \tag{16}$$

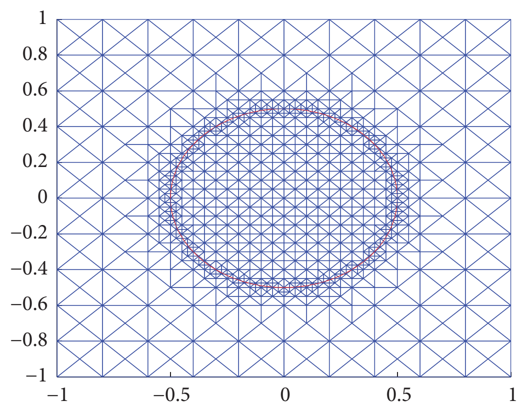
Table 3 is a comparison of the numerical results and CPU time of the adaptive IFEM mesh in [22], the adaptive mesh, and adaptive interface mesh in this paper. From this table we can see that our adaptive method with 933 interior points gives a better result than the adaptive IFEM mesh with 4021 interior points, and the result of our adaptive interface mesh is better than the adaptive mesh. The left and right of Figure 6 are a comparison of the adaptive mesh, the numerical solution, and the numerical error using the adaptive mesh method with 585 interior points (left) and the adaptive interface mesh method with 825 interior points (right). From this figure we can see that after the refinement using the adaptive method, the maximum error comes from area inside the interface but not exactly lies on the interface, so the difference of the results between the adaptive interface mesh and the adaptive mesh is not as large as it is in Example 2.

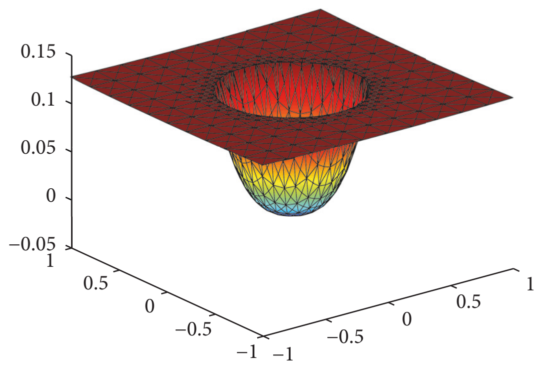
Example 5. This example is taken from [8]. It is an interface problem with a singular point. The level-set function ϕ, the coefficients $\beta^\pm$, and the solution $u^\pm$ are given as follows. The interface is Lipschitz continuous and it has a kink at $(0,0)$, and u is piecewise H^2:

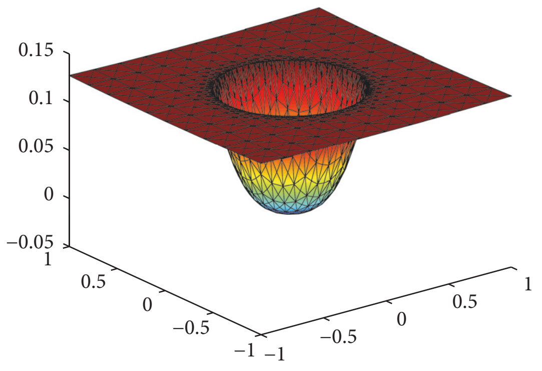
$$\begin{aligned}\phi(x,y) &= y-2x, \quad x+y>0\\ \phi(x,y) &= y+\frac{x}{2}, \quad x+y\le 0\\ \beta^+(x,y) &= 1,\\ \beta^-(x,y) &= 2+\sin(x+y),\\ u^+(x,y) &= 8,\\ u^-(x,y) &= \left(x^2+y^2\right)^{5/6}+\sin(x+y)\end{aligned} \tag{17}$$

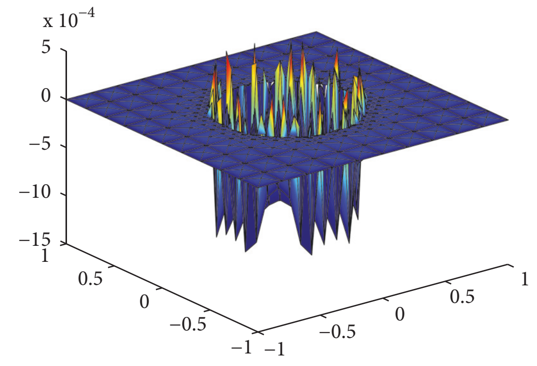
Figure 7 shows the jump of solution and flux in the normal direction on the interface. Table 4 is a comparison of the numerical results and CPU time of the uniform mesh, adaptive mesh, and adaptive interface mesh with different


(a) Grids of Adaptive Mesh

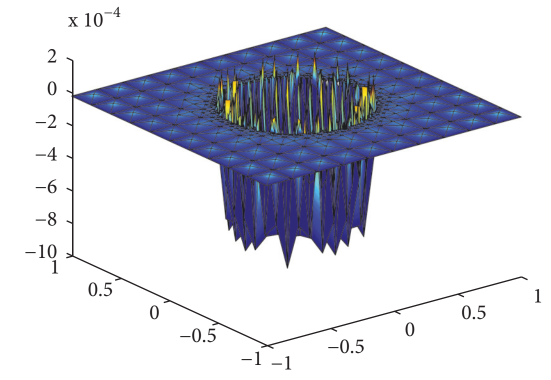

(b) Grids of Adaptive Interface Mesh


(c) Numerical Solution of Adaptive Mesh


(d) Numerical Solution of Adaptive Interface Mesh

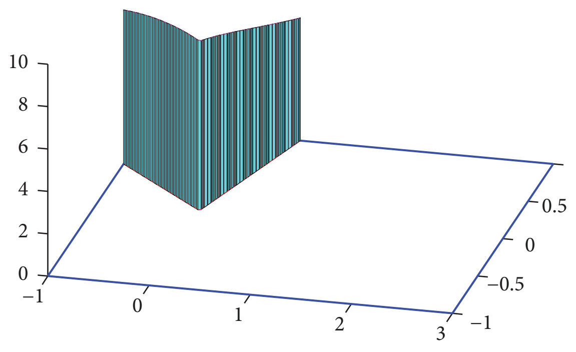


(e) Numerical Error of Adaptive Mesh



(f) Numerical Error of Adaptive Interface Mesh

FIGURE 6: Numerical results of Example 4.


(a) Jump of solution on the interface

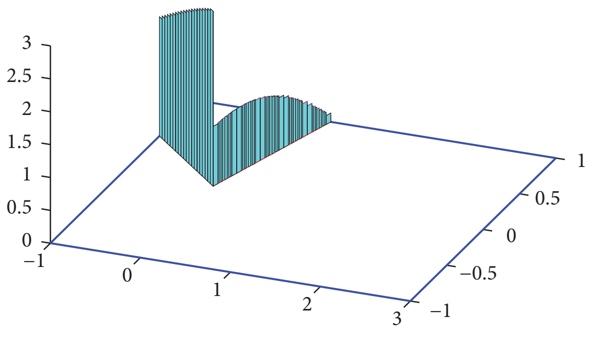
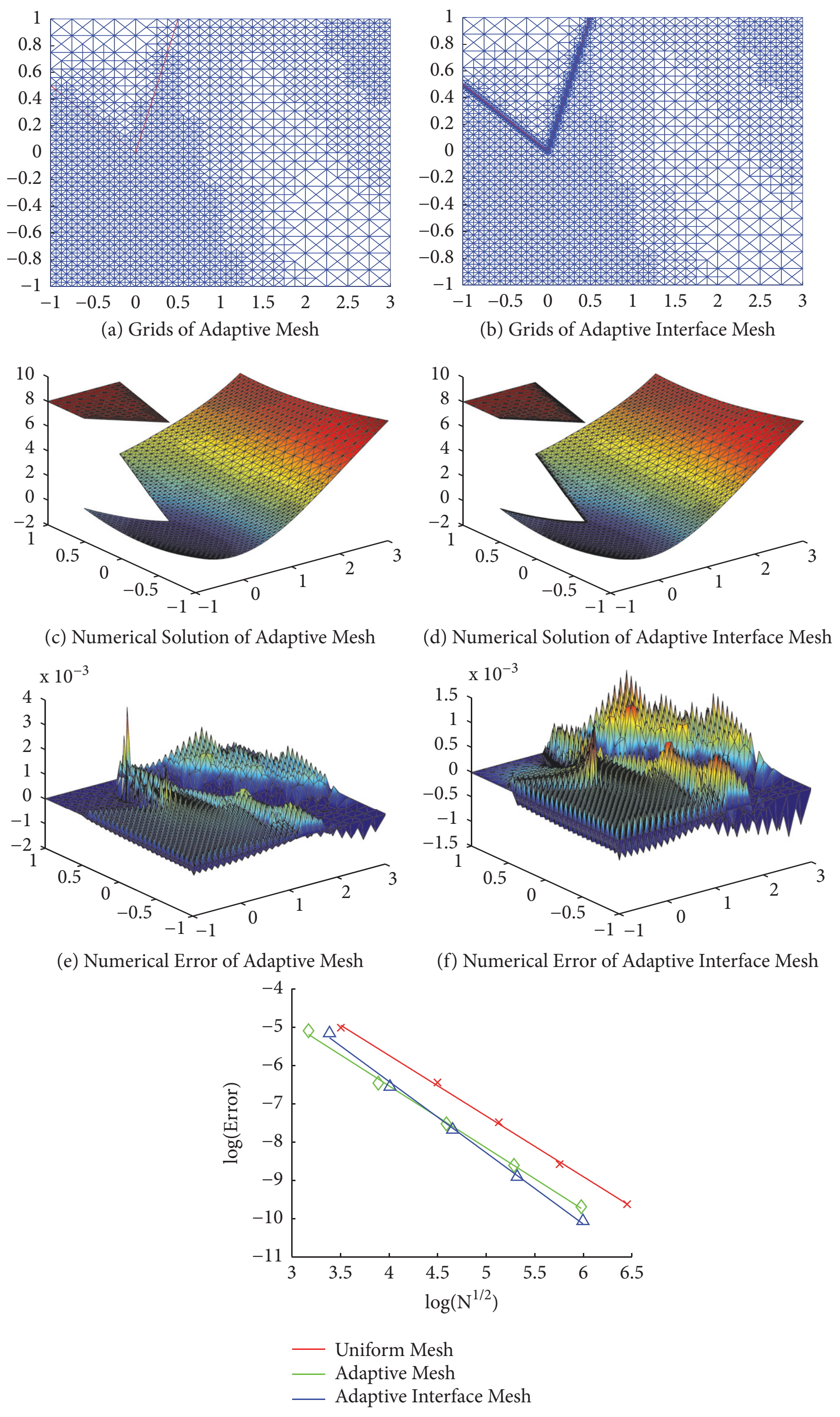

(b) Jump of flux in the normal direction on the interface

FIGURE 7: Jumps on the interface of Example 5.



(a) Grids of Adaptive Mesh

(b) Grids of Adaptive Interface Mesh

(c) Numerical Solution of Adaptive Mesh

(d) Numerical Solution of Adaptive Interface Mesh

(e) Numerical Error of Adaptive Mesh

(f) Numerical Error of Adaptive Interface Mesh

(g) L^{∞} Error for Different Grids

Figure 8: Numerical results of Example 5.

grids. All three methods can get to higher than 1.5th order accuracy. The result of the method using the adaptive mesh is obviously better than the method using the uniform mesh, and the result of the method using the adaptive interface mesh is slightly better result than the method using the adaptive mesh, and the difference of the results of three meshes gets clearer as the triangulation becomes smaller. The left and right of Figure 8 are a comparison of the adaptive mesh, the numerical solution, and the numerical error using the adaptive mesh method with 2385 interior points (left) and the adaptive interface mesh method with 3037 interior points (right). In this example we can see that after the refinement using the adaptive method, the maximum error lies around the interface, so we further refined the mesh around the interface and get a better result as shown in Figure 8(f).

5. Conclusion

In this paper, we proposed the adaptive refinement method to solve the elliptic interface problems using a non-symmetric weak formulation. The key idea is to use two uniform coarse grids to find the location of the maximum error. Sometimes this step can be omitted in practice because the cells with maximum error often locates near the interface, but we provided examples in which the cells with maximum errors are away from the interface. After fixing the location of the maximum error, we use the adaptive mesh refinement method to reduce the maximum error, so that the error of the whole area is of the same scale. Numerical experiments show that the adaptive refinement method has remarkably higher convergency and accuracy than the method using uniform Cartesian grid, and for the same L^{∞} error the overall CPU time is highly reduced using our new method. The study of the three-dimensional form of this new method will be our future work, so that we can solve problems that are more practical.

Data Availability

All data can be accessed in the Numerical Experiments section of this article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

L. Shi's research is supported by National Natural Science Foundation of China (No. 11701569). S. Hou's research is supported by Dr. Walter Koss Professorship made available through the State of Louisiana Board of Regents Support Funds. L. Wang's research is supported by Science Foundations of China University of Petroleum–Beijing (No. 2462015BJB05).

References

[1] C. S. Peskin, "Numerical analysis of blood flow in the heart," *Journal of Computational Physics*, vol. 25, no. 3, pp. 220–252, 1977.

[2] R. J. LeVeque and Z. L. Li, "The immersed interface method for elliptic equations with discontinuous coefficients and singular sources," *SIAM Journal on Numerical Analysis*, vol. 31, no. 4, pp. 1019–1044, 1994.

[3] Z. Li, "The immersed interface method using a finite element formulation," *Applied Numerical Mathematics*, vol. 27, no. 3, pp. 253–267, 1998.

[4] X. He, T. Lin, and Y. Lin, "Immersed finite element methods for elliptic interface problems with non-homogeneous jump conditions," *International Journal of Numerical Analysis & Modeling*, vol. 8, no. 2, pp. 284–301, 2011.

[5] T. Lin, Y. Lin, and X. Zhang, "Partially penalized immersed finite element methods for elliptic interface problems," *SIAM Journal on Numerical Analysis*, vol. 53, no. 2, pp. 1121–1144, 2015.

[6] R. Kafafy, T. Lin, Y. Lin, and J. Wang, "Three-dimensional immersed finite element methods for electric field simulation in composite materials," *International Journal for Numerical Methods in Engineering*, vol. 64, no. 7, pp. 940–972, 2005.

[7] S. Vallaghe and T. Papadopoulo, "A trilinear immersed finite element method for solving the electroencephalography forward problem," *SIAM Journal on Scientific Computing*, vol. 32, no. 4, pp. 2379–2394, 2010.

[8] S. Hou, W. Wang, and L. Wang, "Numerical method for solving matrix coefficient elliptic equation with sharp-edged interfaces," *Journal of Computational Physics*, vol. 229, no. 19, pp. 7162–7179, 2010.

[9] L. Wang, S. Hou, and L. Shi, "An improved non-traditional finite element formulation for solving three-dimensional elliptic interface problems," *Computers & Mathematics with Applications. An International Journal*, vol. 73, no. 3, pp. 374–384, 2017.

[10] L. Wang, S. Hou, and L. Shi, "A numerical method for solving three-dimensional elliptic interface problems with triple junction points," *Advances in Computational Mathematics*, vol. 44, no. 1, pp. 175–193, 2018.

[11] Y. C. Zhou, S. Zhao, M. Feig, and G. W. Wei, "High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources," *Journal of Computational Physics*, vol. 213, no. 1, pp. 1–30, 2006.

[12] S. Yu, Y. Zhou, and G. W. Wei, "Matched interface and boundary (MIB) method for elliptic problems with sharp-edged interfaces," *Journal of Computational Physics*, vol. 224, no. 2, pp. 729–756, 2007.

[13] S. Yu and G. W. Wei, "Three-dimensional matched interface and boundary (MIB) method for treating geometric singularities," *Journal of Computational Physics*, vol. 227, no. 1, pp. 602–632, 2007.

[14] H. Johansen and P. Colella, "A Cartesian grid embedded boundary method for Poisson's equation on irregular domains," *Journal of Computational Physics*, vol. 147, no. 1, pp. 60–85, 1998.

[15] M. Oevermann and R. Klein, "A Cartesian grid finite volume method for elliptic equations with variable coefficients and embedded interfaces," *Journal of Computational Physics*, vol. 219, no. 2, pp. 749–769, 2006.

[16] M. Oevermann, C. Scharfenberg, and R. Klein, "A sharp interface finite volume method for elliptic equations on Cartesian grids," *Journal of Computational Physics*, vol. 228, no. 14, pp. 5184–5206, 2009.

[17] X.-D. Liu, R. P. Fedkiw, and M. Kang, "A boundary condition capturing method for Poisson's equation on irregular domains," *Journal of Computational Physics*, vol. 160, no. 1, pp. 151–178, 2000.

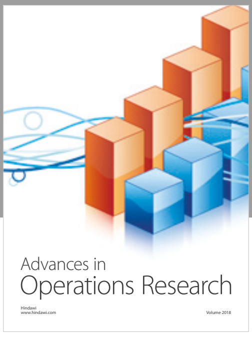
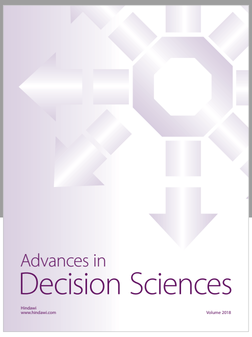
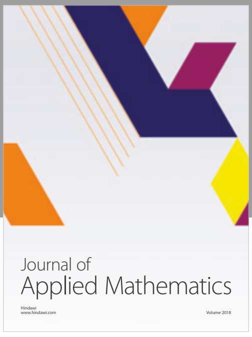
[18] X.-D. Liu and T. C. Sideris, "Convergence of the ghost fluid method for elliptic equations with interfaces," *Mathematics of Computation*, vol. 72, no. 244, pp. 1731–1746, 2003.

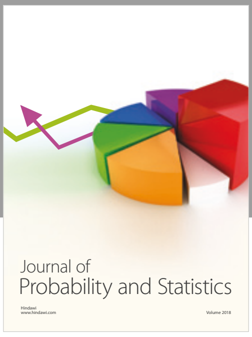
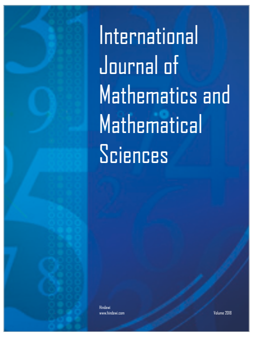
[19] P. Macklin and J. S. Lowengrub, "A new ghost cell/level set method for moving boundary problems: application to tumor growth," *Journal of Scientific Computing*, vol. 35, no. 2-3, pp. 266–299, 2008.

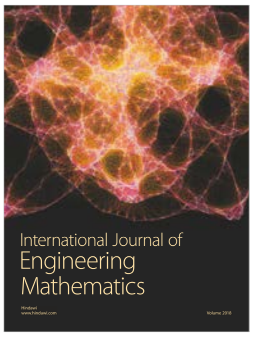
[20] H. Guo and X. Yang, "Gradient recovery for elliptic interface problem: I. body-fitted mesh," *Journal of Computational Physics*, vol. 338, pp. 606–619, 2017.

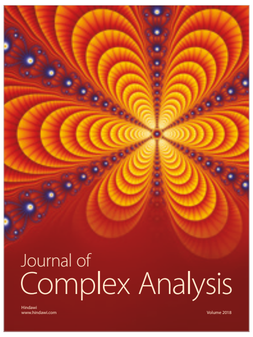
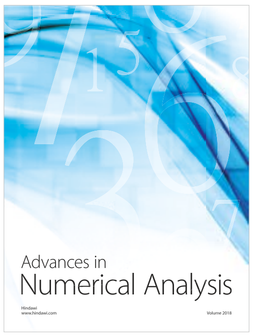
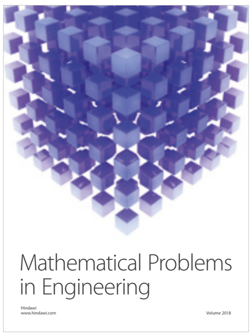
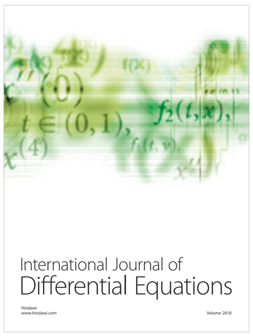
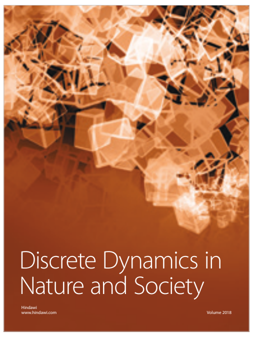
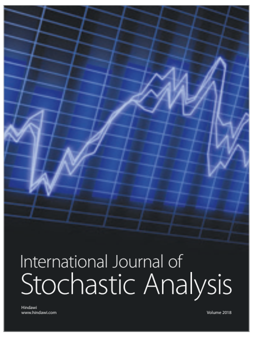
[21] H. Guo and X. Yang, "Gradient recovery for elliptic interface problem: II. immersed finite element methods," *Journal of Computational Physics*, vol. 338, pp. 606–619, 2017.

[22] C.-T. Wu, Z. Li, and M.-C. Lai, "Adaptive mesh refinement for elliptic interface problems using the non-conforming immersed finite element method," *International Journal of Numerical Analysis & Modeling*, vol. 8, no. 3, pp. 466–483, 2011.

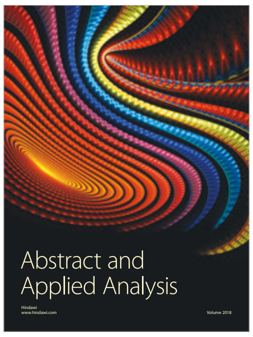
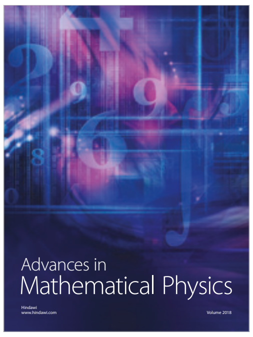

Advances in
Operations Research

Advances in
Decision Sciences

Journal of
Applied Mathematics

The Scientific
World Journal

Journal of
Probability and Statistics

International
Journal of
Mathematics and
Mathematical
Sciences

Journal of
Optimization

International Journal of
Engineering
Mathematics

International Journal of
Analysis

Journal of
Complex Analysis

Advances in
Numerical Analysis

Mathematical Problems
in Engineering

International Journal of
Differential Equations

Discrete Dynamics in
Nature and Society

International Journal of
Stochastic Analysis

Journal of
Mathematics

Journal of
Function Spaces

Abstract and
Applied Analysis

Advances in
Mathematical Physics