Commun. Comput. Phys.
doi: 10.4208/cicp.070211.150811s

Vol. **12**, No. 2, pp. 515-527
August 2012

An Adaptive Mesh Refinement Strategy for Immersed Boundary/Interface Methods

Zhilin Li[1,*] and Peng Song[2]

[1] *Center for Research in Scientific Computation & Department of Mathematics, North Carolina State University, Raleigh, NC 27695, USA; and Nanjing Normal University, China.*
[2] *Operations Research Program, North Carolina State University, Raleigh, NC 27695, USA.*

Received 7 February 2011; Accepted (in revised version) 15 August 2011

Available online 20 February 2012

Abstract. An adaptive mesh refinement strategy is proposed in this paper for the Immersed Boundary and Immersed Interface methods for two-dimensional elliptic interface problems involving singular sources. The interface is represented by the zero level set of a Lipschitz function $\varphi(x,y)$. Our adaptive mesh refinement is done within a small tube of $|\varphi(x,y)| \le \delta$ with finer Cartesian meshes. The discrete linear system of equations is solved by a multigrid solver. The AMR methods could obtain solutions with accuracy that is similar to those on a uniform fine grid by distributing the mesh more economically, therefore, reduce the size of the linear system of the equations. Numerical examples presented show the efficiency of the grid refinement strategy.

AMS subject classifications: 52B10, 65D18, 68U05, 68U07

Key words: Adaptive mesh refinement, immersed boundary method, immersed interface method, elliptic interface problem, Cartesian grid method, level set representation, singular sources.

1 Introduction

In this paper, we develop an adaptive mesh refinement (AMR) technique for the immersed boundary (IB) method and immersed interface method (IIM) for the following elliptic interface problem:

$$\Delta u = f, \qquad (x,y) \in \Omega, \tag{1.1a}$$

$$u|_{\partial\Omega} = g, \tag{1.1b}$$

*Corresponding author. *Email addresses:* zhilin@math.ncsu.edu (Z. Li), psong@ncsu.edu (P. Song)

 ©2012 Global-Science Press

together with the jump conditions across the interface Γ: $\big(X(s),Y(s)\big)$,

$$[u]_\Gamma = w(s), \qquad [u_n]_\Gamma = v(s). \tag{1.2}$$

Here, $\Omega \subset \mathbb{R}^2$ is assumed to be a rectangular domain; the interface $\Gamma \in C^2$ is a curve separating Ω into two sub-domains Ω^-,Ω^+ such that $\Omega = \Omega^- \cup \Omega^+ \cup \Gamma$.

Note that when $w(s) = 0$, the above problem can be written as

$$\Delta u = f + \int_\Gamma v(s)\delta(x - X(s))\delta(y - Y(s))ds, \qquad u|_{\partial\Omega} = g. \tag{1.3}$$

This is the simplest model of Peksin's Immersed Boundary (IB) method, see for example, [7, 19–23] for the IB method, the references therein, and applications. The IB method was originally designed using Cartesian grids. To improve the accuracy of the IB method, the second order immersed interface method (IIM) [1,4,10–13] and the augmented immersed interface method (AIIM) [8, 9, 14, 16, 17] have been developed. Both IB and IIM were originally started with uniform Cartesian grids.

Advantages using uniform Cartesian grids include simplicity, robustness, and no additional cost in the grid generation for free boundary and moving interface problems. Another consideration is that many fast solvers on uniform grids can be employed.

However, uniform meshes may not be efficient or sufficient for some problems that require high resolutions in some part of the solution domain. In order to resolve the accuracy near the interface one can use refined discretization near the interface to improve the interface treatment.

Local grid refinement may be effective for interface problems since (1) often we are mainly interested in the solution near and/or on the interface; (2) the solution away from the interface is smooth enough and therefore dose not require a fine grid to resolve it; (3) often accurate gradient computation near the interface is needed. There are a few adaptive techniques developed for the IB method using Lagrangian formulation, see for example [2, 5, 6]; and for the level set method, see for example [18, 27, 28].

Many AMR techniques use information about some approximate solution to determine where to employ a local mesh refinement technique. If the location of the interface is known in advance, however, then using this a priori information to guide the AMR process may result in a more efficient method. While one AMR approach has been developed for the immersed finite element method (IFEM) in [29], no adaptive mesh refinement technique has been developed to the Immersed Interface Method using a finite difference discretization.

In this paper, we propose an adaptive mesh refinement (AMR) technique for the interface problem above for IB and IIM methods that use finite difference discretizations. Let $\varphi(x)$ be a Lipschitz continuous function whose zero level set ($\varphi = 0$) is the interface. If we have a uniform Cartesian grid with mesh size h_1. We briefly outline our idea of the local refinement strategy.

- For the grid points (x_i, y_j) in the tube $|\varphi| \le \delta$, we generate a finer grid with smaller mesh size h_2 ($h_2 < h_1$).

- Generate the finite difference equations using the standard 5-point or 9-point stencil. Whenever the finite difference scheme at a grid point in the new mesh needs values at points that are not grid points in any level of the mesh, an interpolation scheme is used.
- The resulting finite difference equations can be solved, for example by an algebraic multi-grid solver.

The procedure can be repeated hierarchically. The details will be given in the paper.

The rest of paper is organized as follows. In the next section, we present the adaptive mesh strategy based on a level set function. The data structure is also described. In Section 3, we explain how to construct finite difference scheme using the IB and IIM methods at grid points and at hanging nodes. In Section 4, we present numerical examples and comparisons. The last section contains the conclusions and acknowledgments.

2 Generating the adaptive mesh

We assume that the interface problem is defined on a rectangular domain $\Omega=[a,b]\times[c,d]$. We start with a coarse Cartesian grid, $x_i=a+ih$, $y_j=c+jh$, $i=0,1,\cdots,m$, $j=0,1,\cdots,n$. The interface Γ is implicitly represented by a Lipschitz continuous function $\varphi(x,y)$:

$$\Gamma=\Big\{(x,y),\ \varphi(x,y)=0\Big\}. \tag{2.1}$$

In the discrete case, $\varphi(x,y)$ is defined at grid points as $\varphi(x_i,y_j)$. Often $\varphi(x,y)$ is the signed distance function from Γ.

To generate a finer mesh around interface Γ, we first select parent points within a tube of the interface according to

$$|\varphi(x,y)|\leq\lambda h, \tag{2.2}$$

where λ is a control coefficient to adjust the refinement region. The grid points $\mathbf{x}_{ij}=(x_i,y_j)$ within the tube are selected as parents. We build a refined mesh with new mesh resolution h/r (r is refinement ratio, $r=2$ or 4, for example) within the square: $|x-x_i|\leq h$ and $|y-y_j|\leq h$. Generating refined square for every parent points yields a refined region around interface. Its width is flexible controlled by λ. Fig. 1 shows an example of refinement mesh around a circular interface. If a finer mesh is needed, we can select from the second level grid points by:

$$|\varphi(x,y)|\leq\lambda h/r, \tag{2.3}$$

where h/r is the resolution of the second level refined mesh. We can repeat the process to get finer and finer mesh. Nevertheless, since IIM is second order already, it requires much less refinement as that of the IB method.

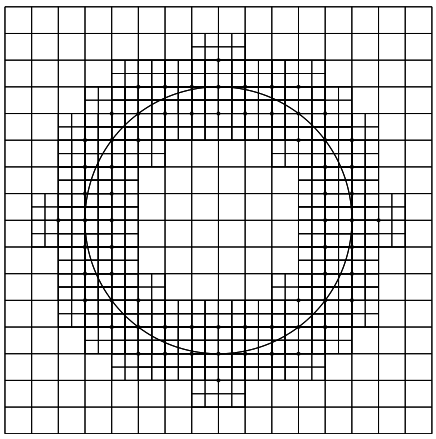

Figure 1: A two level refinement mesh generated within h distance from the interface, that is, $|\varphi_{ij}|=|\varphi(x_i.y_j)|\leq h$, around a circular interface with the refinement ratio being $r=2$ and the control coefficient being $\lambda=1$.

Note that the above generation procedure causes large amount of repetitions among different levels of the mesh. For example, some grid points can be considered to be parents or children multiple times in all levels. Some child grid points can be generated again and again by different parents in its neighbor. Although these repetitions do not affect the mesh pattern, they do cause book-keeping issues and waste of storage space. To overcome this problem, we have developed a procedure to flag, index, and store multiple levels of grid points without repetition. We use a three-level mesh as an example, see also Fig. 2 for a flow chart of the procedure, and Fig. 3 for the mesh. Following the generation steps described above, we start from the first level uniform mesh, obtain second and then third level of adaptive meshes. The indexing steps are in the reversed way. The third level grid points are indexed first and flagged as '3'. When we index the grid

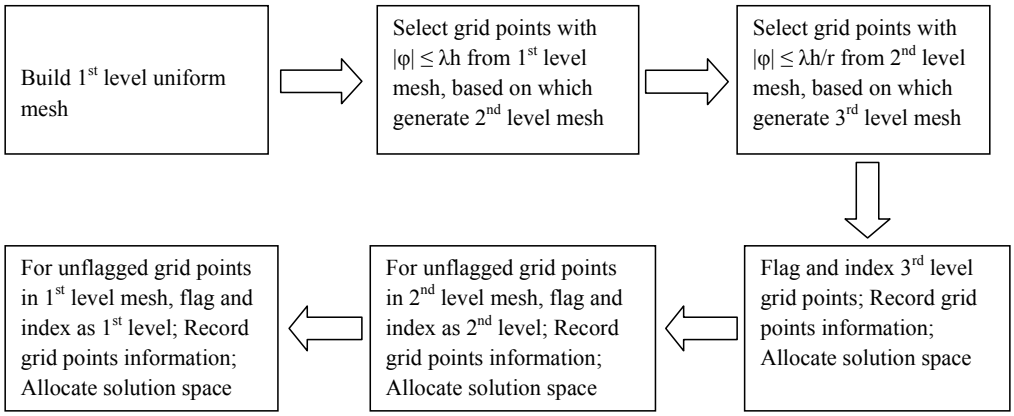

Figure 2: Flow chart of a three-level adaptive mesh generation and indexing procedure.

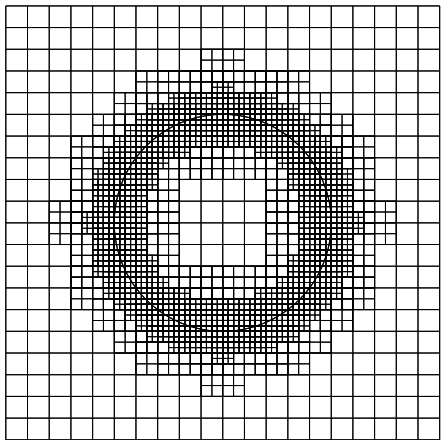

Figure 3: A three-level mesh with the refinement ratio $r=2$ and the control coefficient $\lambda=2$.

points within the second level mesh, we can disregard those grid points that are indexed as '3' since the grid points in the third level mesh is completely within the second. We just need to index those grid points that are left in the second level mesh and flag as '2'. Lastly, we disregard the grid points in both the second and the third levels and just index those grid points that are left and flag them as '1'. In each mesh layer, we index grid points and record their positions in the domain. Once all the grid points in a layer are indexed, the required solution space is then allocated for this level according to its size.

We summarize some properties and restrictions of the adaptive mesh described above:

- Every grid point is flagged as a unique level. If a point in domain is not flagged, that means it is not a grid point and should be expressed by interpolation for use. With the flag information, we can determine which finite difference scheme to use on each grid point, and from which level to call its neighbors required in that scheme.
- The $(k+1)$-th level mesh is entirely embedded in the k-th level mesh. This is to prevent the complicated case where more than two levels show up together locally. Moreover, grid points on border of two levels always belong to the finer level.

Fig. 3 shows a sample three-level mesh around a circular interface. The adaptive mesh satisfies all requirements mentioned above.

2.1 The data structure of the AMR

We illustrate the data structure of our AMR algorithm in Fig. 4 to index grid points, or the unknowns of the finite difference equations. For simplicity, we assume that the domain is a square. We start with the first level uniform Cartesian mesh of $m\times m$ mesh, based on which a second level adaptive mesh is generated with refinement ratio r. There are at most $(mr+1)\times(mr+1)$ possible grid points in the refined domain (the second level) and the unrefined domain. We use the integer coordinate (i,j), $0\leq i,j\leq mr$ to record grid points

rather than their actual location (x_i,y_j)'s. Suppose we count K grid points in the first level and L grid points in the second level, then the double precision vector $U1_{K\times 1}$ and $U2_{L\times 1}$ are allocated to store the solutions of the two levels. We use an array $Locate1_{K\times 2}$ to link the k-th $(1\le k\le K)$ in the first level mesh to its integer coordinate (i,j) in the square domain. Similarly we use an array $Locate2_{L\times 2}$ to link the l-th $(1\le l\le L)$ in the second level mesh to its integer coordinate. Inversely, matrix $Pointer_{(mr+1)\times(mr+1)}$ maps the positions index (i,j) to its corresponding index in the level 1 or 2 meshes. Because some positions index have never been flagged, the matrix *Pointer* can have many undefined entries. Moreover, matrix $Flag_{(mr+1)\times(mr+1)}$ maps each position index to its flag information: 0(unavailable, not a grid point in any level of the mesh), in the level 1 or level 2 mesh. Notice that except for the solution vectors $U1$ and $U2$, all other data structures are integer pointer arrays or matrices to store grids information. Note that some AMR methods based on quadtree/octtree-based structure or the structured AMR approach of Berger and co-workers [2, 3, 18, 27, 28] can save the storage as well. The objective of our paper is to reduce the unknowns on coarse level apart from interface where the solution is already good enough. We believe that with more effort, our AMR approach can save the storage as well but the coding will become more complicated.

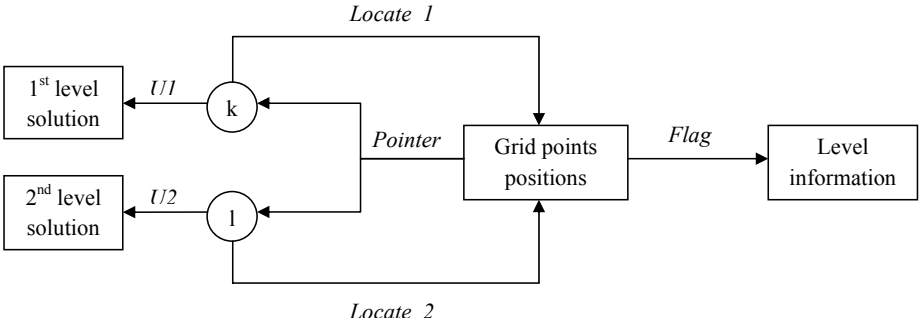

Figure 4: The data structure to book-keep the information of two levels grid points.

3 Finite difference schemes

For most of the grid points in each level, the standard 5-point finite difference scheme is used

$$\frac{U_{i-1,j}+U_{i+1,j}+U_{i,j-1}+U_{i,j+1}-4U_{ij}}{h^2}=f_{ij}+C_{ij}, \tag{3.1}$$

assuming that all the grid points involved are in the same level. The correction term for the immersed boundary method is

$$C_{ij}=\sum v(s_k)\delta_h(x_i-X_k)\delta_h(y_j-Y_k)\Delta s_k, \tag{3.2}$$

where δ_h is the discrete cosine delta function defined by

$$\delta_\epsilon(x)=\begin{cases}\dfrac{1}{4\epsilon}(1+\cos(\pi x/2\epsilon)), & \text{if } |x|<2\epsilon,\\ 0, & \text{if } |x|\geq 2\epsilon,\end{cases} \tag{3.3}$$

$\mathbf{X}_k=(X_k,Y_k)$ is a set of control points (Lagrangian particles) on the interface, $\Delta s_k=|\mathbf{X}_{k+1}-\mathbf{X}_k|$. In the level set function representation of the interface, $\mathbf{X}_k$ can be chosen as the projection of irregular grid points from one particular side of the interface, see for example, [15]. Note that C_{ij} is zero almost everywhere except in the neighborhood of the interface $|\varphi_{ij}|<2h$. Note that, the test results using other type of discrete delta functions reveal similar behavior as that of the discrete cosine delta function.

It is well known that the IB method is first order accurate especially near the interface since the solution is smeared out. The immersed interface method is a second order sharp interface method which means it maintains second order accuracy in the L^∞ norm everywhere including the interface and its neighborhood and the jump conditions are enforced. The IIM for the interface problem (1.1)-(1.2) is similar to the IB method in the sense that the standard central five-point finite difference scheme is used everywhere but C_{ij} is only non-zero in the neighborhood of the interface $|\varphi_{ij}|<h$ instead of $2h$ and is chosen carefully so that second order accuracy can be achieved everywhere.

For a given grid point $\boldsymbol{x}=(x_i,y_j)$, we define

$$\varphi_{ij}^{max}=\max\left\{\varphi_{i-1,j},\varphi_{i+1,j},\varphi_{ij},\varphi_{i,j-1},\varphi_{i,j+1}\right\}, \tag{3.4}$$

$$\varphi_{ij}^{min}=\min\left\{\varphi_{i-1,j},\varphi_{i+1,j},\varphi_{ij},\varphi_{i,j-1},\varphi_{i,j+1}\right\}. \tag{3.5}$$

We call (x_i,y_j) an irregular grid point in reference to the central five point stencil if $\varphi_{ij}^{max}\varphi_{ij}^{min}\leq 0$.

At an irregular grid point, the correction term can be determined dimension by dimension. For example, if the interface cuts the x stencil at x^* such that $x_i<x^*<x_{i+1}$, then

$$\frac{u(x_{i-1},y_j)-2u(x_i,y_j)+u(x_{i+1},y_j)}{h^2}\approx u_{xx}+[u_x]\frac{(x_i-x^*)}{h^2}+[u_{xx}]\frac{(x_i-x^*)^2}{2h^2}.$$

The last two terms are part of the correction term C_{ij}. The jump conditions $[u_x]$, $[u_{xx}]$, $[u_y]$, $[u_{yy}]$ are determined from the following relations in terms of the jump relations in the local coordinates ξ-η, that is, in the normal and tangential directions at (x^*,y_j) or (x_i,y^*),

$$[u_x]=[u_\xi]\cos\theta-[u_\eta]\sin\theta, \tag{3.6a}$$

$$[u_y]=[u_\xi]\sin\theta+[u_\eta]\cos\theta, \tag{3.6b}$$

$$[u_{xx}]=[u_{\xi\xi}]\cos^2\theta-2[u_{\xi\eta}]\cos\theta\sin\theta+[u_{\eta\eta}]\sin^2\theta, \tag{3.6c}$$

$$[u_{yy}]=[u_{\xi\xi}]\sin^2\theta+2[u_{\xi\eta}]\cos\theta\sin\theta+[u_{\eta\eta}]\cos^2\theta. \tag{3.6d}$$

The involved jump relations above in terms of the local coordinates ξ-η can be determined from the original jump condition (1.2) from the following equalities,

$$[u]=w(s), \qquad [u_{\xi}]=v(s), \qquad [u_{\eta}]=\frac{\partial w}{\partial s}, \tag{3.7a}$$

$$[u_{\xi\xi}]=\kappa v(s)+[f]-\frac{\partial^2 w(s)}{\partial s^2}, \tag{3.7b}$$

$$[u_{\eta\eta}]=-\kappa v(s)+\frac{\partial^2 w(s)}{\partial s^2}, \tag{3.7c}$$

$$[u_{\xi\eta}]=\kappa\frac{\partial w(s)}{\partial s}+\frac{\partial v(s)}{\partial s}, \tag{3.7d}$$

where $\frac{\partial v(s)}{\partial s}$ is the first surface derivative along the interface Γ, and κ is the curvature. We refer the readers to [13] for the details since the emphasis of this paper is about the AMR methods.

3.1 The finite difference scheme at hanging nodes

For grid points on border of two mesh levels, the finite difference schemes need to be modified because of different resolutions from two levels. These grid points are called hanging points, or hanging nodes. As a common practice in the literature, see for example, [3, 24, 25] and the references therein, we need to build some virtual points, so called "ghost points", to develop the finite difference scheme.

We illustrate the idea and the finite difference scheme at hanging nodes using Fig. 5, where the points labeled as 6,8,15 are hanging nodes, of which at least one grid point is missing in the standard 5-point finite difference stencil. The idea is to replace the missing point with a ghost point, which can be quadratically interpolated by its neighbor grid points with $\mathcal{O}(h^3)$ accuracy. Having this ghost point plugged into discrete Laplacian and divided by h^2, we get $\mathcal{O}(h)$ truncation error at this hanging point. Since all hanging points take one dimension lower than the whole domain, the global accuracy should not be affected.

We take the hanging point 8 as an example in building the finite difference scheme. The ghost point $g3$ is needed along with 7,9,14 in the central 5-point stencil for the grid point labeled as 8. The value of $g3$ can be interpolated by the values at 8,14 and $g1$ that can be interpolated by the values at points 1,2,3 or 2,3,4. It is easy to check that the following interpolation schemes

$$u_{g1}=\frac{3}{8}u_2+\frac{3}{4}u_3-\frac{1}{8}u_4, \tag{3.8}$$

$$u_{g3}=-\frac{1}{3}u_{14}+u_8+\frac{1}{3}u_{g1}, \tag{3.9}$$

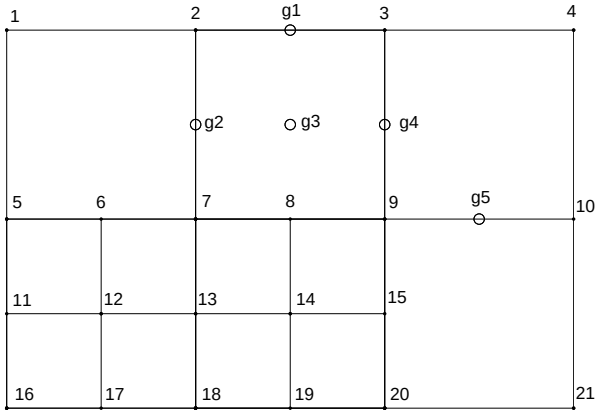

Figure 5: Hanging points and ghost points around the border of two levels mesh with refinement ratio $r=2$.

are third order accurate, where u_i denotes the value of u at the point labeled as i.

After we plug in the above into the standard central 5-point finite difference scheme

$$\frac{u_{14}+u_7+u_9+u_{g3}-4u_8}{h^2/4}=f_8,$$

we get

$$\frac{\frac{2}{3}u_{14}+u_7+u_9+\frac{1}{8}u_2+\frac{1}{4}u_3-\frac{1}{24}u_4-3u_8}{h^2/4}=f_8, \tag{3.10}$$

which is the finite difference equation at the hanging node 8.

4 Numerical examples

We show some numerical results for AMR IB and IIM methods using a benchmark example that was first presented in [11], and later was used by the AMR-IB method in [24,25]. All computations were completed within one or several seconds using a notebook computer. The interface problem is the following

$$u_{xx}+u_{yy}=\int_\Gamma 2\delta(x-X(s))\delta(y-Y(s))ds, \tag{4.1}$$

or equivalently,

$$u_{xx}+u_{yy}=0, \quad \text{on } \Omega\backslash\Gamma, \qquad [u]_\Gamma=0, \quad \left[\frac{\partial u}{\partial n}\right]_\Gamma=2, \tag{4.2}$$

in the domain $\Omega: -1\le x,y\le 1$. The interface is the circle Γ: $x^2+y^2=1/4$. The true solution to the interface problem is

$$u(x,y)=\begin{cases} 1, & \text{if } r\le 1/2, \\ 1+\log(2r), & \text{if } r>1/2, \end{cases} \tag{4.3}$$

where $r=\sqrt{x^2+y^2}$. The Dirichlet boundary condition is used according to the true solution along the boundary of the square.

4.1 The results using the AMR immersed boundary method

We compare our method with the AMR-IB approach [25] to verify that our method is competitive. Note that, the mesh refinement algorithm of our approach is different from [25]. We use the same notations as those used in [25]. AMR $m(+i)[r]$ denotes the mesh that adds i levels of adaptive meshes with the refinement ratio r based on a uniform $m\times m$ mesh. The "error" presented below are the solution errors measured in the L^∞ norm at all grid points; and "Unknowns" is the number of grid points from all levels to be solved in the linear systems of finite difference equations. All results in this and next section are solved by the algebraic multigrid method Amg1r5, see reference [26].

From Table 1, we can see that our AMR method for IB method can save somewhere 60% to 90% amount of memory, that is, the number of unknowns, with about the same accuracy. In Table 1, the ratio is defined as the ratio of the two consecutive errors. For a first order method, the ratio should be around number 2, which confirms the well-known convergence rate of the IB method.

Table 1: Comparison of results using a uniform mesh and that of the AMR-IBM with the same finest resolution h.

	Uniform				AMR(+1)[4], $\lambda=1$			
Finest h	m	Unknowns	Error	Ratio	m	Unknowns	Error	Ratio
2/80	80	6241	1.34e-2		20	2181	1.31e-2	
2/160	160	25281	6.74e-3	1.99	40	5041	6.54e-3	2.00
2/320	320	101761	3.36e-3	2.01	80	13161	3.22e-3	2.03
2/640	640	408321	1.67e-3	2.01	160	39121	1.60e-3	2.01

In Table 2, we show different finest mesh, different levels, and control coefficient λ to show that our results are comparable and consistent to the results obtained in [24,25]. We also try multiple levels of AMR to get finer resolution and better results as that in [24,25]. In Table 2, up to 4 levels of AMR are generated. Moreover, our results also support the argument in [24,25] about the IB method that the accuracy attained by refining only the regions where solution is not smooth is the same as that if the whole domain is uniformly refined with the resolution of the finest level used in the adaptive grids. This is because for the IB method, the largest error for elliptic interface problems occur in the neighbored of the interface.

Table 2: Comparison of AMR-IB method with different levels.

Finest h	Mesh Selection	Unknowns	Error
2/640	160(+2)[2], $\lambda=2$	38821	1.46e-3
2/1280	80(+2)[4], $\lambda=1$	39981	5.04e-4
2/1280	80(+4)[2], $\lambda=2$	38829	4.51e-4

4.2 The results using the AMR immersed interface method

Since the Immersed Interface Method is uniformly second order accurate, the improvement using AMR may not seem to be as obvious as that of the AMR-IB method in terms of the accuracy. Nevertheless, for the reasons mentioned in the introduction section, it is important to develop the AMR strategy for IIM because we still can improve the accuracy in the region of the refined mesh; and can reduce the number of unknowns. In general, if the large error occurs near the neighborhood of the interface, then we can see the improvement in the accuracy using the AMR-IIM. If the large error occurs at the region outside of refinement region, then we may not see improvement in the accuracy using the AMR-IIM. Anyhow, the AMR-IIM will not affect global second order accuracy but reduce size the linear system of the finite difference equations.

In Table 3, we show the results of the AMR-IIM for the same example that used for the AMR-IB method. Even with a coarse grid 40×40, the error obtained from IIM is already smaller than that of IB with a 640×640 mesh. When we use a local refinement, the global error may not decrease since the largest error may occur outside of the local refined mesh, often in the regions near the border of the two meshes. The situation can be improved if we take larger λ to ensure refinement region is not too thin compared with whole domain. Also from Table 3, we can see that the AMR-IIM can save 40%-50% amount of memory, that is, the number of unknowns, with about the same accuracy. Moreover, the AMR-IIM always maintains second order accuracy.

Table 3: Comparison of results using a uniform mesh and that of the AMR-IIM with the same finest resolution h. The runtimes in second by algebraic multigrid solver are also provided.

	Uniform					AMR(+1)[2]					
Finest h	m	Unknowns	Error	Ratio	Time	m	λ	Unknowns	Error	Ratio	Time
2/40	40	1521	8.44e-4		0.03	20	1	805	9.07e-4		0.02
2/80	80	6241	2.45e-4	3.4	0.08	40	2	2745	2.86e-4	3.2	0.06
2/160	160	25281	6.69e-5	3.7	0.31	80	4	10201	7.80e-5	3.7	0.19
2/320	320	101761	1.57e-5	4.3	1.3	160	8	39225	1.89e-5	4.1	0.67
2/640	640	408321	3.65e-6	4.3	5.4	320	16	153805	5.04e-6	3.8	2.5

To further show the advantage of our adaptive mesh over a uniform mesh when applied to the IIM, we designed another series of numerical experiments in Table 4. Given a global error tolerance ϵ, we try to find out the minimum unknowns required by various meshes such that the error $E_\infty \le \epsilon$. We select proper AMR $m(+i)[r]$ to roughly obtain tolerance scale, and then control the refinement width λ to slightly adjust error below tolerance. From Table 4, we can see that when $\epsilon = 10^{-4}$ the uniform mesh needed about 16900 unknowns, the AMR 80(+1)[2] only needed about 7945 unknowns, and the AMR 40(+2)[2] needed about 8353 unknowns. When the error tolerance is lowered to $\epsilon = 2.5\times 10^{-5}$, the uniform mesh needed at least 65025 unknowns, but the AMR 160(+1)[2] needed only 30249 unknowns, and the AMR 80(+2)[2] needed about 28833 unknowns. In general, AMR methods can save about half of the computational cost than that of the uniform mesh for the same accuracy.

Table 4: Number of unknowns needed to have the $L_\infty < \epsilon$ for the AMR-IIM method with different levels.

Given Error Tolerance 1.00e-4			Given Error Tolerance 2.50e-5		
Mesh Selection	Error	Unknowns	Mesh Selection	Error	Unknowns
Uniform: $m=131$	1.00e-4	16900	Uniform: $m=256$	2.59e-5	65025
AMR: 80(+1)[2], $\lambda=1$	9.95e-5	7945	AMR: 160(+1)[2], $\lambda=2$	2.50e-5	30249
AMR: 40(+2)[2], $\lambda=5$	8.13e-5	8353	AMR: 80(+2)[2], $\lambda=9$	2.45e-5	28833

5 Conclusion

In this paper, we have developed an adaptive mesh refinement (AMR) strategy using the level set representation of the interface and applied it to the immersed boundary (IB) and interface (IIM) methods. While AMR methods do not increase accuracy of the discretization, they can efficiently make the use of higher resolution more economical and, ideally, can enable one to obtain accuracy that is similar to the uniformly fine discretization but at a greatly reduced cost. Not only our AMR approach can improve (particularly for the IB method) or maintain the same order of accuracy as the finest mesh, but also reduce the size of the resulting linear system of equations significantly. Note that, this is the first time that an AMR strategy has been applied to IIM using finite difference discretizations.

Acknowledgments

The work of this paper was partially supported by the US ARO grants 550694-MA, the AFSOR grant FA9550-09-1-0520, the US NSF grant DMS-0911434, and the NIH grant 096195-01, and CNSF 11071123.

References

[1] J. T. Beale and A. T. Layton. On the accuracy of finite difference methods for elliptic problems with interfaces. Math. Comput. Sci., 1 (2006), 91–119.
[2] M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. J. Comput. Phys., 82 (1989), 64–84.
[3] M. Berger and I. Rigoutsos. An algorithm for point clustering and grid generation. IEEE on Systems, Man, and Cybernetics, 21 (1991), 1278–1286.
[4] S. Deng, K. Ito, and Z. Li. Three dimensional elliptic solvers for interface problems and applications. J. Comput. Phys., 184 (2003), 215–243.
[5] B. E. Griffith. A comparison of two adaptive versions of the immersed boundary method. Technical report, New York University, 2009, (preprint submitted to Elsevier Science).
[6] B. E. Griffith, R. D. Hornung, D. M. Mcqueen, and C. S. Peskin. An adaptive, formally second order accurate version of the immersed boundary method. J. Comput. Phys., 223 (2007), 10–49.
[7] B. E. Griffith and C. S. Peskin. On the order of accuracy of the immersed boundary method: Higher order convergence rates for sufficiently smooth problems. J. Comput. Phys., 208 (2005), 75–105.

[8] T. Hou, Z. Li, S. Osher, and H. Zhao. A hybrid method for moving interface problems with application to the Hele-Shaw flow. J. Comput. Phys., 134 (1997), 236–252.
[9] J. Hunter, Z. Li, and H. Zhao. Autophobic spreading of drops. J. Comput. Phys., 183 (2002), 335–366.
[10] K. Ito, Y. Kyei, and Z. Li. Higher-order, Cartesian grid based finite difference schemes for elliptic equations on irregular domains. SIAM J. Sci. Comput., 27 (2005), 346–367.
[11] R. J. LeVeque and Z. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. SIAM J. Numer. Anal., 31 (1994), 1019–1044.
[12] Z. Li and K. Ito. Maximum principle preserving schemes for interface problems with discontinuous coefficients. SIAM J. Sci. Comput., 23 (2001), 1225–1242.
[13] Z. Li and K. Ito. The Immersed Interface Method – Numerical Solutions of PDEs Involving Interfaces and Irregular Domains. SIAM Frontier Series in Applied mathematics, FR33, 2006.
[14] Z. Li, K. Ito, and M-C. Lai. An augmented approach for Stokes equations with a discontinuous viscosity and singular forces. Computers and Fluids, 36 (2007), 622–635.
[15] Z. Li and M-C. Lai. The immersed interface method for the Navier-Stokes equations with singular forces. J. Comput. Phys., 171 (2001), 822–842.
[16] Z. Li, X. Wan, K. Ito, and S. Lubkin. An augmented pressure boundary condition for a Stokes flow with a non-slip boundary condition. Commun. Comput. Phys., 1 (2006), 874–885.
[17] Z. Li, W-C. Wang, I-L. Chern, and M-C. Lai. New formulations for interface problems in polar coordinates. SIAM J. Sci. Comput., 25 (2003), 224–245.
[18] F. Losasso, R. Fedkiw, and S. Osher. Spatially adaptive techniques for level set methods and incompressible flow. Computers and Fluids, 35 (2006), 995–1010.
[19] C. S. Peskin. The immersed boundary method. Acta Numerica, 11 (2002), 479–517.
[20] C. S. Peskin and D. M. McQueen. A three-dimensional computational method for blood flow in the heart: (i) immersed elastic fibers in a viscous incompressible fluid. J. Comput. Phys., 81 (1989), 372–405.
[21] C. S. Peskin and D. M. McQueen. A three-dimensional computational method for blood flow in the heart: (ii) contractile fibers. J. Comput. Phys., 82 (1989), 289–297.
[22] C. S. Peskin and D. M. McQueen. A general method for the computer simulation of biological systems interacting with fluids. Symposia of the Society for Experimental Biology, 49 (1995), 265.
[23] C. S. Peskin and B. F. Printz. Improved volume conservation in the computation of flows with immersed elastic boundaries. J. Comput. Phys., 105 (1993), 33–46.
[24] A. Roma, C. S. Peskin, and M. Berger. An adaptive version of the immersed boundary method. J. Comput. Phys., 153 (1999), 509–534.
[25] A. M. Roma. A multilevel self adaptive version of the immersed boundary method. PhD thesis, New York University, 1996.
[26] J. W. Ruge and K. Stuben. Algebraic multigrid in Multigrid Methods. Frontiers in Applied Mathematics, Volume 3, SIAM, Philadelphia, 1987.
[27] J. Strain. Fast tree-based redistancing for level set computations. J. Comput. Phys., 152 (1999), 664–686.
[28] M. Sussman, A. Almgren, J. B. Bell, P. Colella, L. H Howell, and M. L. Welcome. An adaptive level set approach for incompressible two-phase flows. J. Comput. Phys., 148 (1999), 81–124.
[29] C-T. Wu, Z. Li, and M-C. Lai. Adaptive mesh refinement for elliptic interface problems using the non-conforming immersed finite element method. Int. J. Numer. Anal. Model., 1 (2010), 466–483.