ABSTRACT

XIE, HUI. Finite Element Methods for Interface Problems with Locally Modified Triangulations. (Under the direction of Dr. Zhilin Li).

Interface problems arise in many applications such as heat conduction in different materials. The partial differential equations (PDEs) that describe these applications have domains that consist of different subdomains. The different subdomains can have complicated shapes or can have different properties. For instance, different subdomains can represent different phases of the same material, such as water and ice. The coefficients of the PDEs can be discontinuous across the interfaces of the subdomains, and the source terms can be singular. Due to these irregularities, the solutions to the PDEs can be nonsmooth or even discontinuous. Here we restrict ourselves to interface problems that do not depend on time and can be expressed in terms of elliptic or elasticity PDEs.

We present finite element methods (FEMs) for elliptic and elasticity problems with interfaces. The FEMs are based on body-fitted meshes with a locally modified triangulation. A FEM based on a body-fitted mesh uses a triangulation that is aligned with the interfaces. However, for complicated interfaces it can be difficult and expensive to generate such triangulations. That is why we use a locally modified triangulation based on Cartesian meshes. We first form a Cartesian mesh, then move the grid points near the interfaces to the interfaces. This leads to a locally modified triangulation. We use the standard FEM with the locally modified triangulation to solve the elliptic and elasticity problems with interfaces. By FEM theory, the method is second order accurate in the infinity norm for piecewise smooth solutions. We present some numerical examples to show the second order accuracy of the method.

We also present a new second order finite difference method that does not require computing the curvature. At points away from the interface we can approximate the PDE by using the standard 5-point scheme. At points where the interface crosses the 5-point scheme, we still use the 5-point scheme by introducing some ghost values for the grid points on the other side of interface. The price is that we need to find an equation for each ghost value. We use the interface conditions, either the jump in Dirichlet or Neumann boundary conditions, to form the equations for the ghost values to complete the linear system. We also present some numerical examples to show the second order accuracy of the method.

Finite Element Methods for Interface Problems with Locally Modified Triangulations

by
Hui Xie

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fullfillment of the
requirements for the Degree of
Doctor of Philosophy

Applied Mathematics

Raleigh, North Carolina

2009

APPROVED BY:

Dr. Kazufumi Ito	Dr. Zhilin Li Chair of Advisory Committee
Dr. Xiao-Biao Lin	Dr. Sharon R. Lubkin

DEDICATION

To my parents and sister.

BIOGRAPHY

Hui Xie received his B.S. degree in Computational Mathematics & Applied Software from Beijing University, China, in 1999. He received his M.S. degree in Computational Mathematics from Academy of Mathematics and Systems Science, Chinese Academy of Sciences in 2002. From Aug 2002 to Jul 2004, he worked as a software engineer at Beijing FEGEN software Co., Ltd., Beijing, China. In Aug 2004, he started his Ph.D. study in Applied Mathematics at North Carolina State University.

ACKNOWLEDGMENTS

I would like to gratefully acknowledge the patience, guidance and support provided by my advisor, Dr. Zhilin Li. I would like to thank my committee members, Dr. Kazufumi Ito, Dr. Xiao-biao Lin and Dr. Sharon R. Lubkin, for their time to serve as my advisory committee and review my dissertation. I would like to thank Dr. Harvey Charlton and Ms. Denise Seabrooks for their kind help throughout my Ph.D. study at NCSU. I would also like to thank Dr. Hoon Hong for his suggestions and help.

I would like to thank my parents and my sister for their support and love.

I would like to acknowledge the partial supports from US-ARO grant 49308-MA, and US-AFSOR grant FA9550-06-1-0241.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

In this thesis, we propose a finite element method for elliptic problems and plane elasticity problems with interfaces in which the physical parameters and solutions may be discontinuous across an arbitrary interface. We also propose a new finite difference method that does not require computing the curvature for elliptic problems with interfaces. We first give a brief background of the general interface problems.

1.1 Interface problems

Interface problems arise in a diverse range of applications such as heat conduction, multiphase flows and phase transitions in fluid mechanics, thin film and crystal growth simulations in material science, and mathematical biology problems modeled by partial differential equations involving moving fronts. The partial differential equations (PDEs) that describe these applications have domains that consist of different subdomains. The different subdomains can have complicated shapes or can have different properties. For instance, different subdomains can represent different phases of the same material, such as water and ice. The coefficients of the PDEs can be discontinuous across the interfaces of the subdomains, and the source terms can be singular. As a result, the solutions to the PDEs can be nonsmooth or even discontinuous.

For time-independent or fixed interface problems, the interfaces do not move. We solve the governing equations with given boundary conditions for the field variables we are interested in. The situation is different for time-dependent or moving interface problems. A commonly used approach to moving interface problems is a splitting method in which we fix the interface temporarily and solve the governing equations to determine the velocity of the interface. The computed velocity then is used to evolve the interface. Such a process can be done once or iteratively until a convergence criterion is satisfied. So how to solve fixed interface problems is the essential part of both stationary and moving interface problems. In this thesis, we will focus on solving fixed interface problems which do not depend on time.

The difficulties of the interface problems are the existence of nonsmoothness or discontinuity across the interfaces and the complexity or movement of the interfaces. Analytic solutions are rarely available for interface problems so we have to use numerical methods to find approximate solutions. Standard numerical methods which are designed for smooth solutions will lose accuracy in a neighborhood of interfaces where smoothness is lost. To recover the accuracy, we have to take effort in the neighborhood of interfaces.

Many methods have been developed for the interface problems such as

- Smoothing method for discontinuous coefficients;
- Harmonic averaging for discontinuous coefficients;
- Peskin's Immersed Boundary Methods (IBM);
- Ghost fluid methods;
- Immersed Interface Methods (IIM);
- Finite element methods (FEMs) with body-fitted meshes;
- Immersed finite element method(IFEM).

Some methods (immersed boundary methods, ghost fluid methods) are easy to implement but only achieve first order accuracy. Others (immersed interface methods, finite element methods with body-fitted meshes) achieve second order accuracy but are not easy to implement. We refer the reader to [15, 32] for further information.

In this thesis, we will use two methods to solve elliptic and elasticity problems with fixed interfaces. One method is the finite element method (FEM) with a locally modified mesh. The other is a new second order finite difference method that does not require computing the curvature.

The FEM with a locally modified mesh is motivated by the locally modified triangulations proposed in [2, 31] for Poisson equations on irregular domains. We use such meshes to solve the elastic and elasticity problem with interfaces using the Galerkin finite element method [4, 34]. We want to take advantage of Cartesian meshes and the finite element method using body-fitted meshes. The Cartesian meshes have several benefits over non-structured meshes. In the literature several finite element methods have been proposed using Cartesian meshes for elastic and elasticity interface problems. The non-conforming finite element method proposed in [16] is simple and enforces the homogeneous jump condition but it is not fully second order accurate because the basis functions are non-conforming. The conforming finite element method proposed in [9, 16] is second order accurate but it is not easy to implement as the basis functions have wider support in the neighborhood of the interface. The finite element method proposed in this thesis is an alternative approach based on body-fitted meshes with modification to the Cartesian mesh only in the neighborhood of the interface. Thus, the coefficient matrix is altered only in the elements that are near the interface. This nature is fully taken into account in developing an efficient algebraic solver in [11] using a sparse subspace iterative method. In this thesis, for elastic interface problems, the finite element method using the locally modified mesh was also compared with the finite element method based on a locally enriched mesh in which the intersections between the grid lines and the interface are added as additional nodal points. Both methods can lead to second order accurate solution for elliptic problems with interfaces. But the efficient iterative solver proposed in [11] in general can not be applied for the locally enriched mesh because the structure of the matrix has been changed. Other numerical simulations based on the locally modified mesh can be found in [10, 12].

The new second order method is a finite difference method. At points away from the interface we can approximate the PDE by using the standard 5-point scheme which gives second order accuracy. At points where the interface crosses the 5-point scheme, we have to modify the 5-point scheme. These points are called irregular grid points hereafter. It is sufficient to consider such irregular grid points lying outside the interface. We use real values for points outside the interface and ghost values for points inside the interface in the 5-point scheme. So we do not change the form of 5-point scheme by introducing ghost values for grid points inside the interface or more generally outside the subdomain. The price of this is that we need to find an equation for each ghost value. We will use the interface conditions, either the jump in Dirichlet or Neumann boundary conditions, to form the equations for the ghost values to complete the linear system. Noticing that we have normal derivatives in interface conditions, we will use a local coordinates system defined by the normal and tangential directions of the interface at some point on the interface. We use Taylor expansions to expand all the variables including the ghost value that we need to find an equation to third order accuracy. Then we use a least squares approximation to the interface conditions to find the equation for the ghost value. We do not need the curvature in this process. In the literature several finite difference methods have been proposed using Cartesian meshes for interface problems. The immersed boundary method [21] was developed by Peskin in 1977 to model the blood flow in the heart [20]. In general, it is a first order method. The Immersed Interface Method (IIM) was originally invented by R. J. LeVeque and Z. Li [13, 14, 15]. It is a second order method but it needs to compute the curvature which is usually difficult.

1.2 Model problems

1.2.1 Elliptic problems with interfaces

We consider the elliptic problems with interfaces

$$-\nabla\cdot(\beta\,\nabla u) = f(\vec{x}) - \int_{\Gamma} C(s)\,\delta(\vec{x}-\vec{X}(s))ds, \quad \vec{x}=(x,y)\in\Omega=\Omega^{+}\bigcup\Omega^{-}, \tag{1.1}$$

with a given boundary condition on $\partial\Omega$, where $\Gamma\in C^2$ is an interface between the subdomains Ω^+ and Ω^-; see Fig. 1.1 for an illustration. The diffusion coefficient β can have a finite jump across the interface Γ. We assume that $\beta\geq\beta_{min}>0$, and f is a bounded function, δ is the two dimensional Dirac-delta function, $\vec{X}(s)$ is the arc-length parametrization of the interface Γ. The second term at the right hand side is a distribution which satisfies

$$\iint_{\Omega}\int_{\Gamma} C(s)\,\delta(\vec{x}-\vec{X}(s))\Psi(\vec{x})dsd\vec{x} = \int_{\Gamma} C(s)\,\Psi(\vec{X}(s))ds \tag{1.2}$$

for any arbitrary smooth function $\Psi(\vec{x})$ that vanishes on the boundary $\partial\Omega$.

From the equation (1.1), it is easy to obtain the following jump conditions

$$[u]_{\Gamma} = u^{+} - u^{-} = 0, \tag{1.3}$$

$$\left[\beta\frac{\partial u}{\partial n}\right]_{\Gamma} = \beta^{+}\frac{\partial u^{+}}{\partial n} - \beta^{-}\frac{\partial u^{-}}{\partial n} = C, \tag{1.4}$$

where the jump is defined as the difference of the limiting values from the outside of the interface to the inside, and n is the unit normal direction of the interface Γ pointing outward. Therefore, the model interface problem can be written in an equivalent form:

$$\begin{cases} -\nabla \cdot (\beta \nabla u) = f(\vec{x}), \quad \vec{x} = (x, y) \in \Omega \setminus \Gamma \\ [u]_\Gamma = 0, \\ [\beta \frac{\partial u}{\partial n}]_\Gamma = C, \\ u|_{\partial\Omega} = u_0(\vec{x}), \end{cases} \tag{1.5}$$

where u_0 is a given function. We refer the reader to [15, 16] for the derivation and other related information.

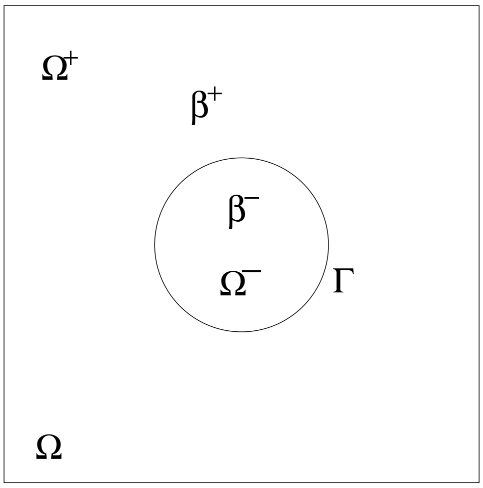


Figure 1.1: A rectangular domain Ω with an immersed interface Γ.

1.2.2 Elasticity problems with interfaces

Elasticity problems with interfaces have wide applications in continuum mechanics, particularly for problems that involve stresses and strains, for example, [7, 18, 33].

Let $\mathbf{x} = (x, y)$ be a point in space and $\mathbf{u} = (u_1(x, y), u_2(x, y))$ be the displacement of a plate which is composed of different materials. The relation between strains and displacements of the plate is given by

$$\varepsilon_{ij}(\mathbf{u}) = \varepsilon_{ji}(\mathbf{u}) = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \quad i, j = 1, 2, \tag{1.6}$$

i.e.,

$$\varepsilon_{11} = \frac{\partial u_1}{\partial x}, \quad \varepsilon_{22} = \frac{\partial u_2}{\partial y}, \quad \varepsilon_{12} = \varepsilon_{21} = \frac{1}{2}\left(\frac{\partial u_1}{\partial y} + \frac{\partial u_2}{\partial x}\right). \tag{1.7}$$

Assuming that the material is linearly elastic and isotropic, and that the displacements are small, we have the following relation between stresses and strains, or the constitutive relation from the Hooke's law,

$$\sigma_{ij} = \lambda \, (\nabla \cdot \mathbf{u}) \, \delta_{ij} + 2\mu\varepsilon_{ij}(\mathbf{u}), \quad i, j = 1, 2, \tag{1.8}$$

where λ and μ are the Lam$\acute{e}$ coefficients, and

$$\delta_{ij} = \begin{cases} 1, & i = j, \\ 0, & i \neq j, \end{cases}$$

$$\nabla \cdot \mathbf{u} = \frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y}.$$

Let $\sigma = (\sigma_{ij})$ be the stress tensor, $\mathbf{f}(\mathbf{x}) = (f_1, f_2)$ be the applied body forces. Then the stress tensor satisfies the following partial differential equation,

$$-\nabla \cdot \sigma = \mathbf{f}, \tag{1.9}$$

i.e.,

$$\begin{cases} -\dfrac{\partial \sigma_{11}}{\partial x} - \dfrac{\partial \sigma_{12}}{\partial y} = f_1, \\ -\dfrac{\partial \sigma_{21}}{\partial x} - \dfrac{\partial \sigma_{22}}{\partial y} = f_2. \end{cases} \tag{1.10}$$

From (1.8)-(1.10), we can re-write the above system as the system of plane elasticity equations of the following,

$$\begin{aligned} -\left\{(\lambda + 2\mu)\frac{\partial^2 u_1}{\partial x^2} + (\lambda + \mu)\frac{\partial^2 u_2}{\partial x \partial y} + \mu\frac{\partial^2 u_1}{\partial y^2}\right\} &= f_1, \\ -\left\{(\lambda + 2\mu)\frac{\partial^2 u_2}{\partial y^2} + (\lambda + \mu)\frac{\partial^2 u_1}{\partial x \partial y} + \mu\frac{\partial^2 u_2}{\partial x^2}\right\} &= f_2, \end{aligned} \tag{1.11}$$

In the vector form, it is

$$-\mu\triangle\mathbf{u} - (\lambda + \mu)\, grad\ div\ \mathbf{u} = \mathbf{f}. \tag{1.12}$$

Note that, in practice, it is common to use the Young's modulus E and Poisson's ratio ν instead of the Lam$\acute{e}$ coefficients λ and μ in the expression (1.8). The relations between λ

and μ, and E and ν, are given by

$$\mu = \frac{E}{2(1+\nu)}, \tag{1.13}$$

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \quad \text{(plane strain)} \quad \lambda = \frac{\nu E}{1-\nu^2} \quad \text{(plane stress)}. \tag{1.14}$$

We want to obtain the numerical solution of the elasticity system that has an interface Γ in the solution domain. Across the interface Γ, the material coefficients may have finite jumps; so does the flux $\sigma\mathbf{n}$ (see Fig. 1.2 for an illustration). Now the problem can be written as follows:

$$\begin{aligned} -\nabla \cdot \sigma &= \mathbf{f} \quad \text{in } \Omega^+ \bigcup \Omega^- && (1.15)\\ [\mathbf{u}]_\Gamma &= 0, && (1.16)\\ [\sigma\mathbf{n}]_\Gamma &= \mathbf{q}, && (1.17)\\ \mathbf{u}|_{\partial\Omega} &= \mathbf{u}_0, && (1.18) \end{aligned}$$

where $\mathbf{f} = (f_1, f_2)$, $\mathbf{q} = (q_1, q_2)$, $\mathbf{u}_0 = (u_{0_1}, u_{0_2})$ are known vector functions, and $\Gamma \in C^2$ is a closed interface between the subdomains Ω^+ and Ω^-. The jump $[\,\cdot\,]_\Gamma$ is defined as the difference of the limiting values from the outside of the interface to the inside, and $\mathbf{n}$ is the unit normal direction of the interface Γ pointing outward. We refer the reader to [24, 28] for more information of the elasticity problems.

1.3 The FEM with a locally modified mesh

Many numerical methods designed for smooth solutions do not work efficiently for interface problems. One way to get rid of the difficulties is to use the finite element method (FEM) with a body-fitted mesh. The FEM is a numerical method for computing approximate solutions to PDEs. The FEM partitions the domain into pieces (triangles or quadrilaterals in 2D) which are also called elements – in contrast to finite difference methods, which discretize the domain with a rectangular grid. The FEM can easily deal with the discontinuity in the coefficients of the PDEs or nonsmoothness in the solutions across the boundaries (edges) of the elements. The FEM is better suited for the solution of interface problems because the triangulation can be more easily adapted to the interfaces.

A body-fitted FEM uses a triangulation that is aligned with the interfaces. However, for complicated interfaces it can be difficult and expensive to generate such triangulations.

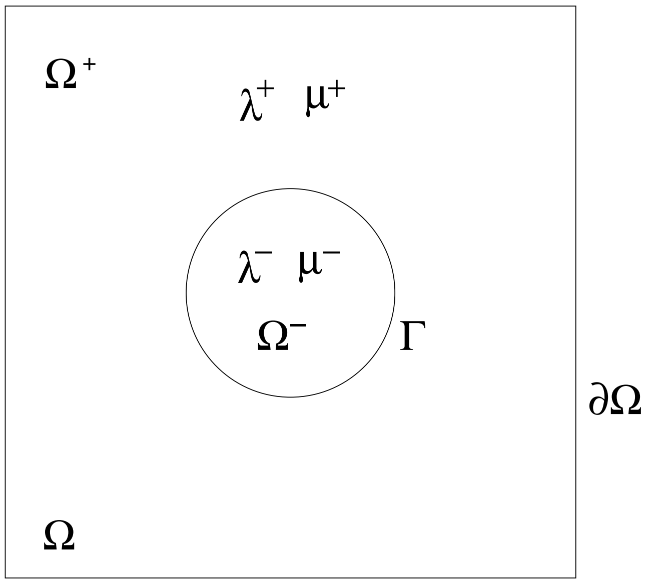


Figure 1.2: A rectangular domain Ω with an immersed interface Γ.

That is why we use a locally modified triangulation based on a Cartesian mesh. We first form the Cartesian mesh, then move the grid points near the interfaces onto the interfaces. This leads to a locally modified triangulation. The advantage of this process is that we only make local modifications. Thus the process is fast and inexpensive.

There are two essential steps one must take to solve a boundary value problem (BVP) using the FEM [29].

- In the first step, one rephrases the original BVP in its weak, or variational form. There is almost no cost to this step.
- The second step is the discretization, where the weak form is discretized in a finite dimensional space.

After this second step, we have concrete formulae for a large but finite dimensional linear problem whose solution will approximately solve the original BVP.

1.4 The level set representation of interfaces

The FEM with a locally modified mesh is based on a Cartesian mesh. We will track the interfaces implicitly by using level set methods [19, 25, 30]. The level set method is a numerical technique for tracking interfaces and shapes. We will use the zero level set of a Lipschitz continuous function $\varphi(x_1, x_2)$ to represent an interface

$$\Gamma = \{(x_1, x_2) \mid \varphi(x_1, x_2) = 0\},$$

and obtain interface information such as the normal directions curvature etc., through the function φ. This function is called a level set function. φ is generally chosen to be the signed distance function and assumed to take negative values inside the region enclosed by the curve Γ and positive values outside. In two dimensions, the zero level set is a plane curve known as a level curve. In three dimensions, the zero level set is a surface known as a level surface.

The advantage of the level set method is that one can perform numerical computations involving curves and surfaces on a fixed Cartesian grid without having to parameterize these objects. The level set method simplifies numerical computation. In particular, for moving interface cases, it is easy to follow interfaces that have topological changes such as merging or splitting, by the level set method.

For such a level set function, the resolution of the interface Γ is determined by the level set function. If the interface is complicated in reference to a mesh, then it may not be resolved well by the level set function. Consequently, the finite element solution obtained from our method may not resolve all fine details of the solution. A finer mesh is then needed to resolve the geometry and the solution.

1.5 The outline of the thesis

The thesis is organized as follows. Chapters 2-4 are devoted to the finite element methods with a locally modified mesh. Chapter 5 is devoted to a new second order finite difference method. In the next chapter, we discuss an important part of the FEM with a locally modified mesh: the mesh generation. In Chapter 3 we apply the method to elliptic problems with interfaces. In Chapter 4 we apply the method to elasticity problems with interfaces. In Chapter 5 we present a new second order finite difference method and apply it to elliptic problems with interfaces. We conclude in Chapter 6.

Chapter 2

The mesh generation processes

The generation of a body-fitted triangulation from a Cartesian grid is an essential part of our algorithm. We describe two different approaches here. Both of them start from a uniform Cartesian grid; see Fig. 2.1. In general, the Cartesian grid is not a body-fitted mesh for interface problems.

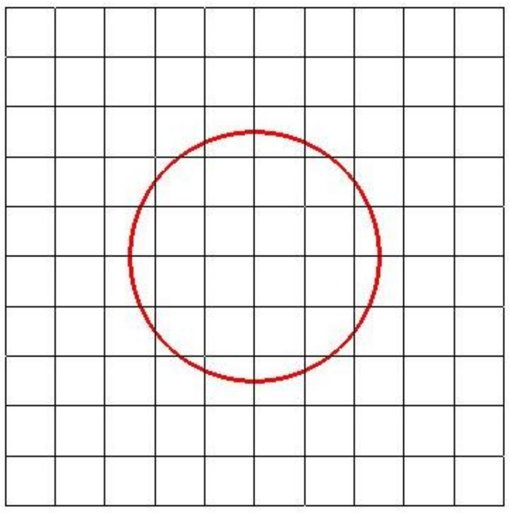


Figure 2.1: A Cartesian grid, not a body-fitted mesh.

2.1 A locally modified mesh

The first approach is to perturb the Cartesian grid, which does not increase the number of nodal points or the degrees of the freedom. The idea is the same as used in [2], but the implementation is different.

The procedure is the following:

Step 1 Generate a rectangular grid. We denote the grid points (x_i, y_j), $0 \le i \le M$, $0 \le j \le N$. For simplicity, we assume that the step sizes are the same in both x and y direction. We denote the step size by h.

Step 2 Define a level set function φ on the rectangular grid to represent the interfaces. Each grid point has a value s.t.

$$\begin{cases} \varphi(x_i, y_j) < 0, & \text{if } (x_i, y_j) \text{ is inside the interface } \Gamma; \\ \varphi(x_i, y_j) = 0, & \text{if } (x_i, y_j) \text{ is on the interface } \Gamma; \\ \varphi(x_i, y_j) > 0, & \text{if } (x_i, y_j) \text{ is outside the interface } \Gamma. \end{cases}$$

For example, we can define $\varphi(x, y) = x^2 + y^2 - \frac{1}{4}$ for a circle centered at $(0, 0)$ with radius $\frac{1}{2}$ which is the interface inside a box $[-1, 1] \times [-1, 1]$. So we will have negative values inside the circle, positive values outside the circle and zeroes on the interface.

Step 3 Find the intersections of rectangular grid lines and the interface Γ using the level set function φ. We search the horizontal and vertical grid lines and use linear interpolation to get the coordinates of intersection points. Below is a sample Fortran 77 code to find the intersections in vertical grid lines. The search in horizontal grid lines is similar.

```
c   the domain is [a0,b0]*[c0,d0]
c   v: level set function
c   xys: coordinates of intersection points
c   n records the number of intersection points
    n=0
    for i=1,M
       for j=1,N-1
          if (v(i,j)*v(i,j+1).le.0.d0) then
               n = n + 1
c   determine x-,y-coordinates of intersection points
c   by linear interpolation
c   (v2-v1)/h=(0-v1)/t    so t=v1/(v1-v2)*h
               xys(1,n) =a0+ (i - 1)*h
               t = v(i,j) - v(i,j+1)
               if (abs(t).lt.1.d-9) then
                  t = 0.d0
               else
                  t = v(i,j)/t
               end if
               xys(2,n) = c0+(j - 1 + t)*h
            end if
         end do
      end do
```

We call the intersection points *interface points* in this thesis. If the coordinates of an interface point lie in $(x_i - h/2, x_i + h/2] \times (y_j - h/2, y_j + h/2]$, then we call such a grid point (x_i, y_j) an *irregular grid point.* Otherwise a grid point is called a *regular grid point.*

We move each irregular grid point to a new location on the interface as a new nodal point (replacing the original one). If there is only one interface point associated with the irregular grid point, then we simply replace the irregular grid point with the interface point as the new nodal point. If there is more than one interface point associated with the grid point (x_i, y_j), we choose the nearest intersection to the grid point as the new nodal point; see Fig. 2.2 for an illustration.

In Fig. 2.2, the irregular grid point is P, and the interface points are $P1$ and $P2$. Since P is closer to $P2$ than to $P1$, we move P to $P2$ as a new nodal point. In Fig. 2.2, for simplicity of the discussion, we assume that points A, C, D, F, G, and point H are regular grid points which means that there is no need to move them. We move the point B to $P3$, and E to $P4$. Then we have four quadrilaterals around point $P2$; see Fig. 2.3 (a).

Remark for Step 2 & 3: *There are some differences between the implementation in this thesis and in [2]. In that paper, the author used a discrete indicator function of values in* $\{-1, 0, 1\}$ *instead of using a continuous level set function. The author did not disclose how to represent the interface and determine the coordinates of intersection points. The author used a different criterion to move irregular grid points rather than to choose the nearest interface points.*

Step 4 Form the triangulation. The emphasis is how to generate triangles at irregular grid points. Now we discuss how to generate the triangles after we have moved irregular

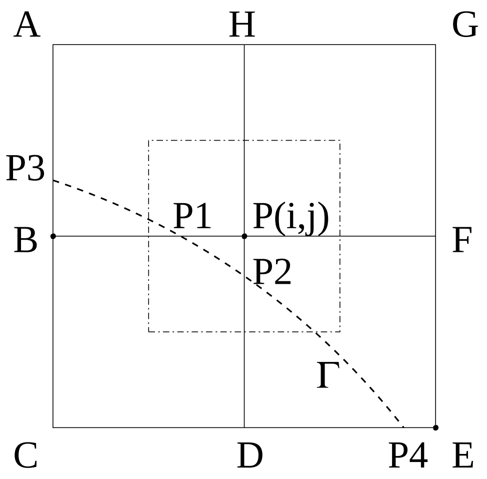


Figure 2.2: A typical irregular grid point P with coordinates (x_i, y_j) and the geometry. There are two interface points P1, P2 lie in the inner domain $(x_i - h/2, x_i + h/2] \times (y_j - h/2, y_j + h/2]$. So P is an irregular grid point. We will move P to its nearest interface point. Here we move P to $P2$ since $|PP2| < |PP1|$, where $|PP2|$ is the distance between P and $P2$ and so on.

grid points to get new nodal points. We use Fig. 2.3 to illustrate the idea. Each of the quadrilaterals of the perturbed grid is divided into two triangles along one of its diagonals. Let K be the number of irregular grid points (perturbed nodal points) in a quadrilateral. We divide the quadrilateral into triangles according to the following rules:

– If $K = 0$, then the quadrilateral is rectangular. We form the triangles by connecting two vertices from the lower left corner to the upper right corner.

– If $K = 2$ and the two irregular grid points are the opposite corners, then we connect the irregular grid points; see, for example, $P2$ and $P4$ in Fig. 2.3.

– If $K = 1$, or $K = 2$ and the two irregular grid points are not the opposite corners, then we connect the diagonal which gives better mesh quality. In Fig. 2.3 (a), we would connect $P2$ and C, $P3$ and H, H and F to form the triangulation; see Fig. 2.3 (b). Usually the criterion is to connect the shorter diagonal.

– The cases $K = 3$ and $K = 4$ are special cases that rarely happen if the interface Γ is smooth and the mesh is fine enough. We omit this discussion here but refer the readers to [2, 22] for the details.

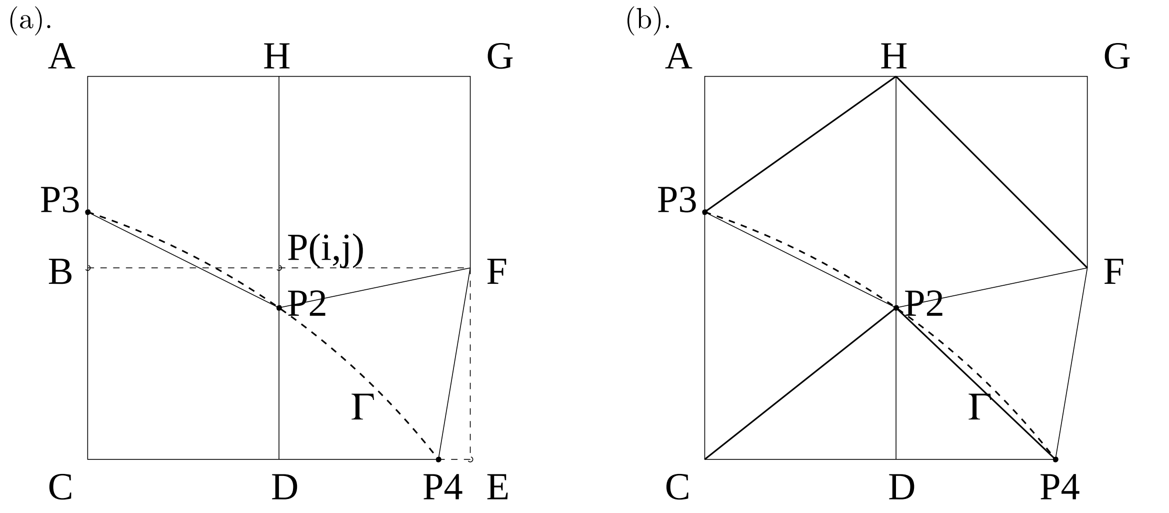


Figure 2.3: (a). The geometry near a perturbed nodal point $P2$. We assume points P, B, E are irregular grid points and they are replaced by the new nodal points P2, P3, P4 respectively. After these perturbations, there are four quadrilaterals around nodal point $P2$ which may not be rectangles. (b). Resulting triangles from the left quadrilaterals. For the quadrilateral P2 D P4 F where the interface Γ cuts through the quadrilateral along one of its diagonal P2 P4, we connect P2 and P4. For other quadrilaterals, we connect the shorter diagonal.

It can be proved [2] that for $\Gamma \in C^2$ this mesh generation algorithm leads to a quasi-uniform triangulation, that is, for each triangle the ratio between the length of the longest side and the length of the shortest side is bounded. Furthermore, the accuracy of the approximation of the interface Γ is $O(h^2)$.

2.2 A locally enriched mesh

For a comparison, we also briefly describe another approach in which the intersections between the grid lines and the interface are added as additional nodal points; see also [16]. With a piecewise linear finite element space this increases the number of degrees of freedom by $O(N)$. Furthermore, the structure of the Cartesian grid is lost. Below we describe how to generate the locally enriched mesh[1].

Step 1 Generate a rectangular grid (x_i, y_j) as before.

Step 2 For a regular quadrilateral, that is, for which the interface does not cut through the

[1]It is called a Cartesian mesh with added nodes in [16]

quadrilateral, we generate the triangles by connecting one of the diagonals as we discussed earlier.

For an irregular quadrilateral, we find the intersection of the grid lines and the interface as before. If the intersection is one of the vertices of the quadrilateral, nothing needs to be done. If not, then we add the intersection as an additional nodal point. We then use the newly added nodes and the vertices of the quadrilateral to form the triangles. Let K be the total number of added nodes in the irregular quadrilateral. The triangulation process is as follows:

- If $K = 1$: We connect this node with the two vertices on the opposite side. See Fig. 2.4.

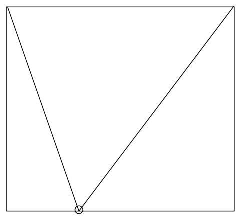


Figure 2.4: One added node.

- If $K = 2$:
  * If the two points are on the adjacent edges, we connect these two points and with the vertex farthest from these points. See Fig. 2.5.

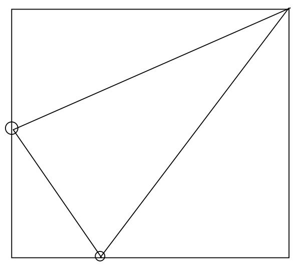


Figure 2.5: Two added adjacent nodes.

  * If the two points are on opposite edges, we connect these two points and get two quadrilaterals. Then we cut the quadrilaterals along one of the diagonals which cuts the obtuse angle. If the quadrilaterals are still rectangular, then

we join the left lower and right upper corner; see Fig. 2.6 for an illustration.

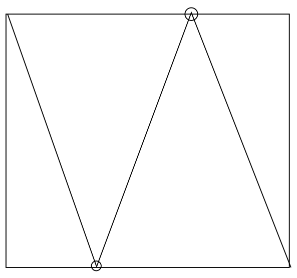


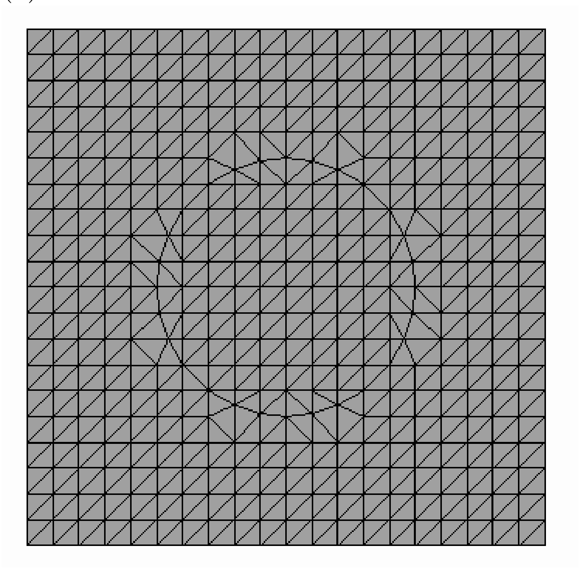
Figure 2.6: Two added opposite nodes.

- There can be some special cases that need different treatments: for example, if the interface is along one side of a quadrilateral. We omit the details here for simplicity.

The locally enriched mesh is a simple, direct, and intuitive approach, but the algorithm can lead to poor mesh quality such as skinny triangles.

For the purposes of comparison and illustration, in Fig. 2.7 we show two different meshes, in which Fig. 2.7 (a) is a locally modified mesh, while Fig. 2.7 (b) is a locally enriched mesh. In the next chapter, we will present a comparison of the errors of the finite element solutions using two different meshes.

(a).



(b).

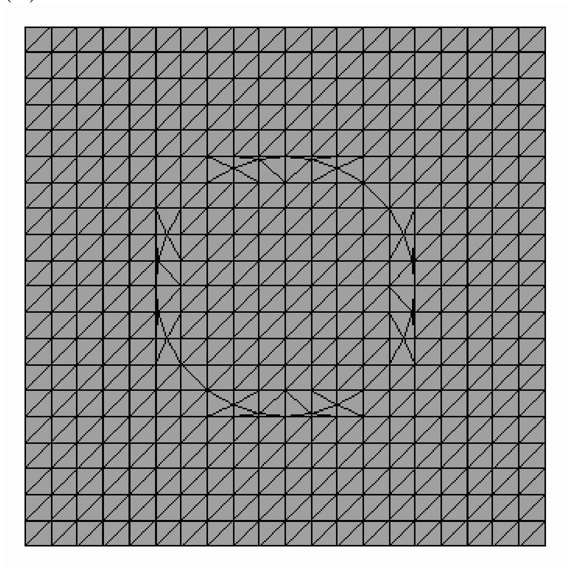


Figure 2.7: A comparison of two meshes. (a). A locally modified mesh; (b). A locally enriched mesh.

Chapter 3

The FEM for elliptic problems with interfaces

For convenience, we re-write the model problem again (see Section 1.2.1):

$$\begin{cases} -\nabla \cdot (\beta\, \nabla u) = f(\vec{x}), \quad \vec{x} = (x, y) \in \Omega \setminus \Gamma, \\ [u]_\Gamma = 0, \\ \left[\beta \frac{\partial u}{\partial n}\right]_\Gamma = C, \\ u|_{\partial\Omega} = u_0(\vec{x}). \end{cases} \tag{3.1}$$

We will use the standard notation for Hilbert space:

$$\begin{aligned} H^1(\Omega) &= \{w : w \in L^2(\Omega), w' \in L^2(\Omega)\} \\ H_0^1(\Omega) &= \{w \in H^1(\Omega) : w \mid \partial\Omega = 0\} \end{aligned}$$

3.1 The weak formulation

In the following we apply the standard Galerkin finite element method to a homogeneous Dirichlet boundary value problem. We multiply both sides of the equation (3.1) by a test function $v(x, y) \in H_0^1(\Omega)$ and integrate over the domain Ω^+ and Ω^- to obtain

$$\iint_{\Omega^+} [-(\beta u_x)_x - (\beta u_y)_y] v dx dy = \iint_{\Omega^+} f v dx dy, \tag{3.2}$$

$$\iint_{\Omega^-} [-(\beta u_x)_x - (\beta u_y)_y] v dx dy = \iint_{\Omega^-} f v dx dy. \tag{3.3}$$

Then applying Green's theorem on the above equations and using $v|_{\partial\Omega} = 0$, we get

$$\int_\Gamma \beta^+ v^+ \frac{\partial u^+}{\partial n} ds + \iint_{\Omega^+} \beta(x,y)\nabla u \nabla v dxdy = \iint_{\Omega^+} fvdxdy, \tag{3.4}$$

$$-\int_\Gamma \beta^- v^- \frac{\partial u^-}{\partial n} ds + \iint_{\Omega^-} \beta(x,y)\nabla u \nabla v dxdy = \iint_{\Omega^-} fvdxdy, \tag{3.5}$$

where n is the unit normal direction of the interface Γ pointing outward. Adding (3.4) to (3.5), we obtain

$$\int_\Gamma \beta^+ v^+ \frac{\partial u^+}{\partial n} ds - \int_\Gamma \beta^- v^- \frac{\partial u^-}{\partial n} ds + \iint_{\Omega} \beta(x,y)\nabla u \nabla v dxdy = \iint_{\Omega} fvdxdy. \tag{3.6}$$

Since $[u]_\Gamma = u^+ - u^- = 0$ and $[\beta \frac{\partial u}{\partial n}]_\Gamma = \beta^+ \frac{\partial u^+}{\partial n} - \beta^- \frac{\partial u^-}{\partial n} = C$, we have the weak form:

$$\iint_{\Omega} \beta(x,y)\nabla u \nabla v dxdy = \iint_{\Omega} fvdxdy - \int_\Gamma Cvds \quad \text{for all } v \in H_0^1(\Omega). \tag{3.7}$$

The problem then is to find $u \in H_0^1(\Omega)$ such that the weak form above is true for all $v \in H_0^1(\Omega)$.

3.2 The finite element discretization

If we introduce the following notation:

$$\begin{aligned} a(u,v) &= \iint_\Omega \beta(x,y)\nabla u \nabla v dxdy \\ f(v) &= \iint_\Omega fvdxdy - \int_\Gamma Cvds \\ V &= H_0^1(\Omega) \end{aligned}$$

The problem becomes
Find $u \in V$ such that

$$a(u,v) = f(v) \quad \textit{for all } v \in V. \tag{3.8}$$

The dimension of the Hilbert space $V = H_0^1(\Omega)$ is infinite. To find an approximate solution for (3.8), we need to use some finite-dimensional space V_n to approximate the infinite-dimensional space V. Here n denotes the dimension of the space V_n. For Galerkin's method, we choose V_n to be a subspace of V, i.e. $V_n \subset V$. So we get the following Galerkin discretization:

Find $u_n \in V_n$ *such that*

$$a(u_n, v_n) = f(v_n) \quad \textit{for all } v_n \in V_n. \tag{3.9}$$

If we denote the basis for V_n as $\{\varphi_j\}_{j=1}^n$, we can write

$$u_n = \sum_{j=1}^{n} u_j \varphi_j$$

Then (3.9) leads to a linear system

$$\sum_{j=1}^{n} u_j a(\varphi_j, \varphi_i) = f(\varphi_i), \quad i = 1, n \tag{3.10}$$

or

$$\begin{pmatrix} a(\varphi_1,\varphi_1) & \cdots & \cdots & a(\varphi_1,\varphi_n) \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a(\varphi_n,\varphi_1) & \cdots & \cdots & a(\varphi_n,\varphi_n) \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} f(\varphi_1) \\ \vdots \\ \vdots \\ f(\varphi_n) \end{pmatrix} \tag{3.11}$$

The matrix is often called the stiffness matrix. For our problem, the stiffness matrix is symmetric positive definite. We can solve the linear system for $u_j, j = 1, n$ by using direct or iterative methods.

To construct a finite-dimensional space V_n to approximate the infinite-dimensional space V, we need to subdivide the computational domain into subdomains (finite elements). This process is called triangulation and done in Chapter 2. Then we need to construct basis functions over the triangulation or mesh T_h. Different choices of basis functions lead to different methods. The FEM employs basis functions that are locally defined polynomials within each element $T \in T_h$. Elements are connected at specific points, called nodes. Each basis function is defined to be normalized with value 1 at one node and zero at all others. Because of this, the matrix in (3.11) is sparse, which means most of the entries are zeroes.

In this thesis, we will use piecewise linear polynomials as the basis functions. Let $N_1, \ldots, N_n$ be all the nodes in the triangulation. The linear basis function φ_j satisfies

$$\varphi_j(N_i) = \begin{cases} 1, & i = j; \\ 0, & i \neq j. \end{cases}$$

See Fig 3.1 for a typical basis function.

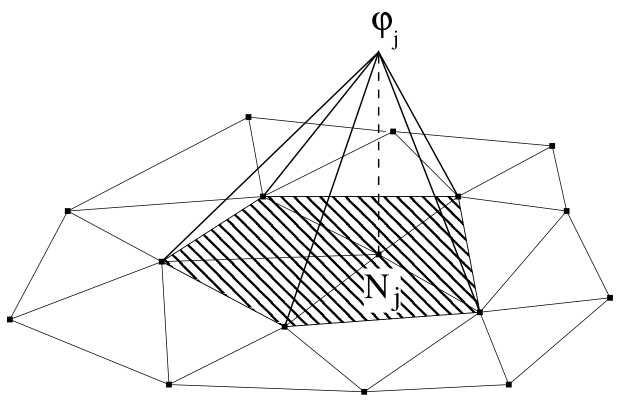


Figure 3.1: A basis function φ_j for node N_j.

3.3 Dealing with nonhomogeneous Dirichlet BC

We can change the nonhomogeneous Dirichlet BC to homogeneous BC by constructing a function $\tilde{u}_n$ to satisfy the nonhomogeneous Dirichlet BC. If we order the inner nodes prior to the boundary nodes and denote the basis functions as $\{\varphi_1, \cdots, \varphi_n, \tilde{\varphi}_1, \cdots, \tilde{\varphi}_{n_b}\}$ where $\{\varphi_1, \cdots, \varphi_n\}$ are the basis functions for inner nodes and $\{\tilde{\varphi}_1, \cdots, \tilde{\varphi}_{n_b}\}$ for boundary nodes, then we can write

$$u_n = \sum_{j=1}^{n} u_j \varphi_j + \sum_{j=1}^{n_b} \tilde{u}_j \tilde{\varphi}_j.$$

If we set

$$\tilde{u}_n = \sum_{j=1}^{n_b} \tilde{u}_j \tilde{\varphi}_j,$$

then

$$\sum_{j=1}^{n} u_j a(\varphi_j, \varphi_i) = f(\varphi_i) - a(\tilde{u}_n, \varphi_i), \quad i = 1, n.$$

In the next section, we will show how to use FEPG (Finite Element Program Generator) to generate all the Fortran 77 source code for finite element computation.

3.4 Generation of FE source code by FEPG

FEPG (Finite Element Program Generator) [17] is a very powerful tool to generate the FE source code. For ANY kind of finite element modelings, FEPG can automatically generate complete source code based on given PDE and algorithm expressions.

FEPG is suitable for solving FE problems in various scientific and engineering fields. It is a major breakthrough from the limitation that current general-purpose FE software packages are only suitable for particular fields and particular problems. Users only need to input all expressions and formulae for the FEM, then FEPG can automatically generate the complete source code, including element subroutines, algorithm programs, etc. In this way, a lot of coding time can be saved while the correctness and the consistency of the program can be guaranteed.

The work in this thesis was done with FEPG 5.2. In general, there are three types of files we need to provide for the FEPG system (Ver 5.2): GCN file, GIO file and PDE (FBC) files. There may be multiple PDE files for solving multi-physics problems. The GCN file organizes the computation procedure. The GIO file provides some general information. The PDE (FBC) files describe the volume (boundary) integrals in the weak form. Further details are in [17].

Here we will illustrate how to write the GCN, GIO and PDE files for one example. The interface Γ is the circle $x^2+y^2 = 1/4$ within the computational domain $\Omega = (-1,1)\times(-1,1)$. The equations are

$$-(\beta u_x)_x - (\beta u_y)_y = f(x,y) - C\int_\Gamma \delta(\vec{x} - \vec{X}(s))ds, \tag{3.12}$$

with

$$f(x,y) = -8(x^2+y^2) - 4 \tag{3.13}$$

and

$$\beta(x,y) = \begin{cases} x^2+y^2+1, & x^2+y^2 \le \frac{1}{4} \\ b, & x^2+y^2 > \frac{1}{4}. \end{cases} \tag{3.14}$$

The Dirichlet boundary condition is given and we assume the solution u is continuous across the interface Γ and $[\beta u_n]|_\Gamma = C$. Here we set $b = 10$ and $C = 0.1$.

The weak form is

$$\iint_\Omega \beta(x,y)\nabla u \nabla v dxdy = \iint_\Omega f v dxdy - \int_\Gamma C v ds \quad \text{for all } v \in H_0^1(\Omega). \tag{3.15}$$

3.4.1 Writing GCN, GIO files

The filenames of GCN and GIO files must be the same. We list them below. The comments follow '\'.

GIO file "le.gio":

```
intf         \ Define the name of PDE file we will use.

#elemtype t3 \ Define the element shape (t for triangular elements)
             \and node number (3) of each element.
2dxy         \ Define the coordinate type.
             \This is 2d Cartesian coordinates.
```

GCN file "le.gcn":

```
defi            \Define solution algorithm for
                \every decoupled physical field.
a ell &         \The field "a" associates with the first
                \PDE name in the GIO file.
                \"ell" is the algorithm for solving the
                \elliptic equation provided by FEPG.

STARTilu a      \Initialize the solver which uses the conjugate
                \gradient method with ilu as pre-conditioner.
SOLVilu a       \Solve the linear system using the conjugate
                \gradient method with ilu as pre-conditioner.
```

We can select a different solver by replacing `ilu` with `sin` (symmetric direct method), `sor` (SOR iterative method) etc.

3.4.2 Writing PDE file

Next, we will show the PDE (FBC) files based on the weak form (3.15). We notice that there are both volume integrals and a boundary integral in the weak form. We need to write a PDE file for volume integrals and an FBC file for the boundary integral. If there is no boundary integral in the weak form, then we do not need to write the FBC file. The name of the PDE (FBC) file is given in the first part of the GIO file. We name it "intf.pde" ("intf.fbc").

In the "intf.pde" file, we will use "`ek`" to denote $\beta(x, y)$ and "`eq`" to denote $f(x, y)$. FEPG uses F77 naming conventions. This means that all variables starting with the letters "i-n" are integers and all others are real.

PDE file "intf.pde":

```
disp u,             \Define the name of the unknown function.
coor x,y,           \Define the coordinate system.
shap %1 %2          \Define the shape function.
                    \The parameters "%1, %2" are retrieved from
```

```
                      \"#elemtype t3" in GIO file.
gaus %3               \Define type of integration
                      \(nodal integral or Gaussian integral).
                      \"%3" is also retrieved from
                      \"#elemtype t3" in GIO file.
                      \For Gaussian integral, it gives
                      \the number of integral points.
mate ek eq 1d0;0d0;   \Define the material parameters "ek, eq"
                      \which will be used in weak form and their
                      \default values.
                      \So ek=1d0, eq=0d0.
                      \The values can be modified in the following
                      \"stif" segment.

stif                  \Define stiffness matrix.
$c6 elen=x*x+y*y          \Insert fortran code following the "$c6"
                          \to define the material parameters.
                          \"$c6" means beginning with 6 spaces
                          \which is the fortran 77 convention.
$c6 if(imate.eq.1) then   \Define material parameter "ek".
$c6   ek=elen+1d0         \"imate=1" signifies that the element is
$c6 else                  \inside the circle. We will give an
$c6   ek=10d0             \explanation below.
$c6 endif
$c6 eq=-(elen*8d0+4d0)    \Define material parameter "eq".
dist =+[u/x;u/x]*ek+[u/y;u/y]*ek    \Explained below.

load =+[u]*eq

end
```

Here is the explanation for "`imate`" in the stiff segment. When we generate the mesh in Chapter 2, we assign a material number for each element, '1' for elements inside the circle and '2' for others. So we can use an `if` statement to define the material parameter "`ek`" for $\beta(x, y)$ in (3.14).

The keyword "`dist`" in the stiff segment defines the volume integral in the left hand side of the weak form which contains the unknown function. The name of the unknown function is "`u`", which is defined in "`disp u`". The volume integral in the left hand side of the weak form (3.15) is

$$\iint_\Omega \beta(x, y)\nabla u \nabla v dx dy$$

or

$$\iint_\Omega \beta(x,y)u_x v_x dxdy + \iint_\Omega \beta(x,y)u_y v_y dxdy.$$

So we write it in the FEPG language as

```
dist =+[u/x;u/x]*ek+[u/y;u/y]*ek.
```

The first part "`+[u/x;u/x]*ek`" corresponds to $\iint_\Omega \beta(x,y)u_x v_x dxdy$. In the expression, the "`[ ; ]`" corresponds to a volume integral. The "`u/x`" preceding the semicolon ';' corresponds to u_x. The "`u/x`" following the semicolon ';' corresponds to v_x. The "`ek`" corresponds to $\beta(x,y)$. The second part has similar meaning.

The volume integral in the right hand side of the weak form (3.15) is

$$\iint_\Omega fvdxdy$$

We write it in the FEPG language as

```
load =+[u]*eq
```

Here, "`u`" corresponds to the test function v. The "`eq`" corresponds to $f(x,y)$. The "`[ ]`" corresponds to volume integral.

3.4.3 Writing FBC file

What is left in the weak form (3.15) is the boundary integral

$$-\int_\Gamma Cvds.$$

FBC file "intf.fbc":

```
disp u            \Define the name of the unknown function.
coor x            \Define the coordinate system.
shap %1 %2        \Define the shape function.
gaus %3           \Define type of integration.
mate eb 0.1d0     \Define the material parameter "eb"
                  \which will be used in weak form and its
                  \default value.

stif              \Define stiffness matrix.
dist=+[u;u]*0d0   \Explained below.

load=-[u]*eb

end
```

The format of FBC files is nearly the same as that for PDE files. Since the elements treated in FBC files are boundary elements, their dimension is one order lower than that for volume elements. The "`dist=+[u;u]*0d0`" means that there is no boundary integral in the left hand side of the weak form. The "`load=-[u]*eb`" corresponds to $-\int_\Gamma Cvds$.

3.4.4 Generation of FE source code

After preparing the "le.gcn", "le.gio" , "intf.pde" and "intf.fbc" files, we can just run "gio le" under the FEPG environment to generate all the Fortran 77 source code. Then we can insert some Fortran code to customize the output. FEPG also integrates a pre/post processor GID [5] into its system. It provides a user-friendly graphical user interface for geometrical modeling, data input and visualization of results for all types of numerical simulation programs.

The FEPG system helpfully provides many templates for GCN, GIO and PDE (FBC) files. There are some techniques to simplify the writing of PDE files by utilizing tensor products, gradient operators etc. which are discussed in [17].

3.5 Numerical results

We present numerical experiments using the proposed locally modified mesh method. We compare the results with the ones on locally enriched meshes having added nodes at all intersections between the interface and the grid lines. With this approach the dimension of the system of linear equations is larger and the Cartesian structure is lost. We use the standard piecewise linear Galerkin finite element method.

We present an example with a known exact solution from [15] so that we can compute the error and convergence rate. The interface Γ is the circle $x^2 + y^2 = 1/4$ within the computational domain $\Omega = (-1,1) \times (-1,1)$. The equations are

$$-(\beta u_x)_x - (\beta u_y)_y = f(x,y) - C\int_\Gamma \delta(\vec{x} - \vec{X}(s))ds, \tag{3.16}$$

with

$$f(x,y) = -8(x^2 + y^2) - 4 \tag{3.17}$$

and

$$\beta(x,y) = \begin{cases} x^2+y^2+1, & x^2+y^2 \le \frac{1}{4} \\ b, & x^2+y^2 > \frac{1}{4}. \end{cases} \tag{3.18}$$

The Dirichlet boundary condition is given by the exact solution

$$u(x,y) = \begin{cases} r^2 & \text{if} \quad r \le \frac{1}{2} \\ (1-\frac{1}{8b}-\frac{1}{b})/4 + (\frac{r^4}{2}+r^2)/b + C\log(2r)/b/2 & \text{if} \quad r > \frac{1}{2}. \end{cases} \tag{3.19}$$

where $r = \sqrt{x^2+y^2}$. For any b and C, the solution u is continuous across the interface $r = 1/2$ and $[\beta u_n]|_\Gamma = C$. For problems with a discontinuous solution, we refer the readers to the technique discussed in [9].

We use the standard Galerkin finite element method with piecewise linear basis functions to solve it. We consider two cases with $b = 10$ and $b = 1000$. In both cases β is discontinuous with $C = 0.1$.

We use the Finite Element Program Generator (FEPG) to generate the stiffness matrix and the load vector as in Section 3.4. We use the incomplete LU decomposition (ILU) along with the conjugate gradient method to solve the resulting linear system of equations.

The linear system for the locally enriched mesh is larger than that for the locally modified mesh. We expect longer CPU time for solving the linear system for the finite element method with the locally enriched mesh. On an Intel Pentium 1.6GHz processor with 768MB of RAM, the CPU time of the FE method using the locally enriched mesh with a 320×320 grid is 21 seconds, while the FE method using the locally modified mesh is about 19 seconds.

Let U_{ij} be the finite element solution at the grid points (x_i, y_j). Let $\tilde{u}$ be the interpolation solution such that $\tilde{u} = \sum_{ij} U_{ij}\varphi_{ij}$, where φ_{ij} is the basis function at point (x_i, y_j). We define e_∞, e_0, e_a, e_1 as the errors in the L^∞, L^2, energy, and H^1 norms, respectively, in the following way:

$$e_\infty = \max_{i,j} |u(x_i, y_j) - U_{ij}|, \tag{3.20}$$

$$e_0 = \sqrt{\int_\Omega (u-\tilde{u})^2 d\Omega}, \tag{3.21}$$

$$e_a = \sqrt{\int_\Omega (\beta(u_x-\tilde{u}_x)^2 + \beta(u_y-\tilde{u}_y)^2) d\Omega}, \tag{3.22}$$

and

$$e_1 = \sqrt{\int_\Omega ((u-\tilde{u})^2 + (u_x - \tilde{u}_x)^2 + (u_y - \tilde{u}_y)^2) d\Omega}. \tag{3.23}$$

Let N be the number of intervals in each direction on the Cartesian grid.

Case 1, $C = 0.1,\ b = 10$**.**

In Table 3.1 we show the result of a grid refinement analysis of the finite element solutions using the two different meshes. In the table, the first column is the number of intervals in the x and the y directions; the second column is the error in the maximum norm. The third column is the ratio of the two consecutive errors. The ratio approaches number four for quadratic convergence and number two for linear convergence. The other columns in this table and other tables have the similar meanings. The finite element methods are both roughly second order accurate in the maximum norm as expected. In Table 3.2, we also present the results in the L^2 (second order), energy and H^1 norms (first order).

In Figure 3.2, we show a linear regression analysis for $c = 0.1,\ b = 10$ for the finite element solution using the locally modified mesh. From the slope, we can observe that the convergence order is roughly 2. The mesh varies from $N = 100$ to $N = 320$ according to $N = 100 + 10k$, $k = 0, \dots, 22$.

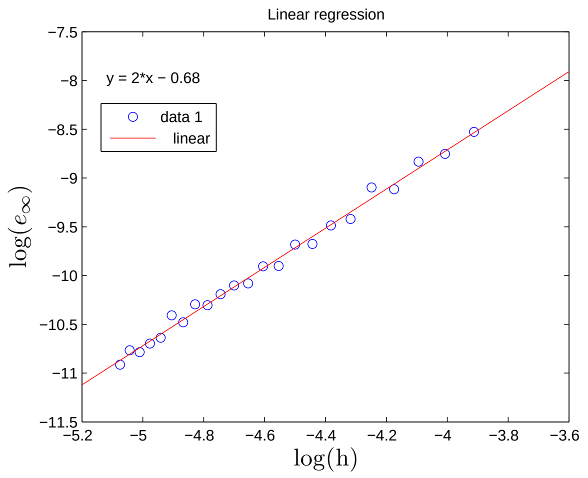


Figure 3.2: A linear regression analysis of the convergence order in log-log scale for $c = 0.1,\ b = 10$ for the finite element solution using the locally modified mesh. The mesh varies according to $N = 100+10k$, $k = 0, \dots, 22$. From the slope, the convergence order is roughly 2.

Case 2, $C = 0.1,\ b = 1000$**.**

The coefficient β has a bigger jump compared with Case 1. We show the same grid refine-

Table 3.1: A grid refinement analysis in the maximum norm of the finite element methods using two different meshes for Case 1.

$C = 0.1,\ b = 10$				
	locally modified mesh		locally enriched mesh	
N	e_∞	ratio	e_∞	ratio
20	3.5081E-3		3.4920E-3	
40	1.1067E-3	3.1700	8.9005E-4	3.9234
80	2.9460E-4	3.7565	2.3232E-4	3.8312
160	7.5932E-5	3.8797	6.3728E-5	3.6454
320	1.8183E-5	4.1761	1.6156E-5	3.9446

Table 3.2: A grid refinement analysis in L^2, energy, and H^1 norms of the finite element methods using two different meshes for Case 1.

$C = 0.1,\ b = 10$						
locally modified mesh						
N	e_0	ratio	e_a	ratio	e_1	ratio
20	2.4657E-3		3.1786E-2		1.5488E-2	
40	6.7869E-4	3.6330	1.0644E-2	2.9862	5.1462E-3	3.0097
80	1.8407E-4	3.6872	4.1852E-3	2.5433	1.9771E-3	2.6029
160	4.6440E-5	3.9635	1.5905E-3	2.6314	7.4842E-4	2.6417
320	1.1430E-5	4.0629	5.5541E-4	2.8636	2.6867E-4	2.7857
locally enriched mesh						
N	e_0	ratio	e_a	ratio	e_1	ratio
20	2.3961E-3		4.0277E-2		1.7297E-2	
40	6.2432E-4	3.8379	1.5061E-2	2.6742	6.0933E-3	2.8387
80	1.5986E-4	3.9055	5.7430E-3	2.6225	2.2593E-3	2.6970
160	4.0567E-5	3.9407	2.1418E-3	2.6815	8.3928E-4	2.6919
320	1.0233E-5	3.9643	7.5062E-4	2.8533	3.0689E-4	2.7348

ment results in Table 3.3. The results are quantitatively the same as in Case 1. In Table 3.4, we present the results in the L^2, energy, and H^1 norms.

Table 3.3: A grid refinement analysis in the maximum norm of the finite element methods using two different meshes for Case 2.

$C = 0.1,\ b = 1000$				
	locally modified mesh		locally enriched mesh	
N	e_∞	ratio	e_∞	ratio
20	3.2222E-3		3.2220E-3	
40	1.1167E-3	2.8855	8.2369E-4	3.9117
80	3.0983E-4	3.6041	2.0978E-4	3.9264
160	8.0474E-5	3.8501	5.4798E-5	3.8283
320	1.9307E-5	4.1681	1.3915E-5	3.9379

Table 3.4: A grid refinement analysis in L^2, energy, and H^1 norms of the finite element methods using two different meshes for Case 2.

$C = 0.1,\ b = 1000$						
locally modified mesh						
N	e_0	ratio	e_a	ratio	e_1	ratio
20	2.1762E-3		3.3751E-2		1.6358E-2	
40	6.1872E-4	3.5173	1.1812E-2	2.8574	5.4530E-3	2.9998
80	1.7183E-4	3.6007	4.8557E-3	2.4326	2.1660E-3	2.5176
160	4.3527E-5	3.9477	1.8397E-3	2.6394	8.1848E-4	2.6464
320	1.1430E-5	3.8080	5.5541E-4	3.3123	2.6867E-4	3.0464
locally enriched mesh						
N	e_0	ratio	e_a	ratio	e_1	ratio
20	2.1272E-3		4.8196E-2		1.9641E-2	
40	5.5894E-4	3.8058	1.7940E-2	2.6865	6.8174E-3	2.8810
80	1.4401E-4	3.8813	6.8224E-3	2.6296	2.5637E-3	2.6592
160	3.6752E-5	3.9183	2.5369E-3	2.6892	9.4370E-4	2.7166
320	9.3078E-6	3.9486	8.7973E-4	2.8837	3.3714E-4	2.7991

Chapter 4

The FEM for elasticity problems with interfaces

For convenience, we re-write the model problem again (see Section 1.2.2):

$$\begin{aligned} -\nabla\cdot\sigma &= \mathbf{f} \quad \text{in } \Omega^+\bigcup\Omega^- && (4.1)\\ [\mathbf{u}]_\Gamma &= 0, && (4.2)\\ [\sigma\mathbf{n}]_\Gamma &= \mathbf{q}, && (4.3)\\ \mathbf{u}|_{\partial\Omega} &= \mathbf{u}_0, && (4.4) \end{aligned}$$

where $\mathbf{u} = (u_1, u_2)$ is the unknown vector function.

4.1 The weak formulation

First we apply the standard Galerkin finite element method to a homogeneous Dirichlet boundary value problem to derive the weak form. We multiply both sides of the equation (4.1) by a test vector function $\mathbf{v} = (v_1, v_2)$, $v_1, v_2 \in H_0^1(\Omega)$ and integrate over the domain Ω^+ and Ω^- to obtain

$$-\iint_{\Omega^+}\left\{\left(\frac{\partial\sigma_{11}}{\partial x}+\frac{\partial\sigma_{12}}{\partial y}\right)v_1+\left(\frac{\partial\sigma_{21}}{\partial x}+\frac{\partial\sigma_{22}}{\partial y}\right)v_2\right\}dxdy = \iint_{\Omega^+}(f_1v_1+f_2v_2)dxdy. \quad (4.5)$$

$$-\iint_{\Omega^-}\left\{\left(\frac{\partial\sigma_{11}}{\partial x}+\frac{\partial\sigma_{12}}{\partial y}\right)v_1+\left(\frac{\partial\sigma_{21}}{\partial x}+\frac{\partial\sigma_{22}}{\partial y}\right)v_2\right\}dxdy = \iint_{\Omega^-}(f_1v_1+f_2v_2)dxdy. \quad (4.6)$$

Then integrating each term by parts and rearranging on the above equations and using $v_1|_{\partial\Omega} = 0, v_2|_{\partial\Omega} = 0$, we get

$$\int_\Gamma \Big\{(\sigma_{11}n_1 + \sigma_{12}n_2)v_1 \ + (\sigma_{12}n_1 + \sigma_{22}n_2)v_2\Big\}\, ds$$
$$+ \iint_{\Omega^+} \left\{\sigma_{11}\frac{\partial v_1}{\partial x} + \sigma_{12}\left(\frac{\partial v_2}{\partial x} + \frac{\partial v_1}{\partial y}\right) + \sigma_{22}\frac{\partial v_2}{\partial y}\right\} dxdy = \iint_{\Omega^+} (f_1v_1 + f_2v_2)dxdy, \tag{4.7}$$

and

$$-\int_\Gamma \Big\{(\sigma_{11}n_1 + \sigma_{12}n_2)v_1 \ + (\sigma_{12}n_1 + \sigma_{22}n_2)v_2\Big\}\, ds$$
$$+ \iint_{\Omega^-} \left\{\sigma_{11}\frac{\partial v_1}{\partial x} + \sigma_{12}\left(\frac{\partial v_2}{\partial x} + \frac{\partial v_1}{\partial y}\right) + \sigma_{22}\frac{\partial v_2}{\partial y}\right\} dxdy = \iint_{\Omega^-} (f_1v_1 + f_2v_2)dxdy, \tag{4.8}$$

where $\mathbf{n} = (n_1, n_2)$ is the unit normal direction of the interface Γ pointing outward. Adding (4.7) to (4.8), we obtain

$$\int_\Gamma \Big\{q_1v_1 \ + q_2v_2\Big\}\, ds + \iint_\Omega \left\{\sigma_{11}\frac{\partial v_1}{\partial x} + \sigma_{12}\left(\frac{\partial v_2}{\partial x} + \frac{\partial v_1}{\partial y}\right) + \sigma_{22}\frac{\partial v_2}{\partial y}\right\} dxdy$$
$$= \iint_\Omega (f_1v_1 + f_2v_2)dxdy \tag{4.9}$$

using the fact that $[\mathbf{u}]_\Gamma = 0$ and $[\sigma\mathbf{n}]_\Gamma = \mathbf{q}$. Thus we have arrived at the weak form:

$$\iint_\Omega \left\{\sigma_{11}\frac{\partial v_1}{\partial x} + \sigma_{12}\left(\frac{\partial v_2}{\partial x} + \frac{\partial v_1}{\partial y}\right) + \sigma_{22}\frac{\partial v_2}{\partial y}\right\} dxdy$$
$$= \iint_\Omega (f_1v_1 + f_2v_2)dxdy - \int_\Gamma \{q_1v_1 + q_2v_2\}\, ds. \tag{4.10}$$

The problem then is to find $\mathbf{u} \in H^1(\Omega) \times H^1(\Omega)$ such that the weak form above is true for all $\mathbf{v} \in H_0^1(\Omega) \times H_0^1(\Omega)$.

If we denote

$$\sigma = \begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{pmatrix}, \qquad \varepsilon(\mathbf{v}) = \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{12} \end{pmatrix} = \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \gamma_{12} \end{pmatrix} = \begin{pmatrix} \frac{\partial v_1}{\partial x} \\ \frac{\partial v_2}{\partial y} \\ \frac{\partial v_1}{\partial y} + \frac{\partial v_2}{\partial x} \end{pmatrix} \tag{4.11}$$

and

$$\mathbf{f} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}, \qquad \mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, \qquad \mathbf{q} = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}, \tag{4.12}$$

then the weak form can be written as

$$\iint_\Omega \sigma^T\varepsilon(\mathbf{v})dxdy \ = \ \iint_\Omega \mathbf{f}^T\mathbf{v}dxdy - \int_\Gamma \mathbf{q}^T\mathbf{v}ds. \tag{4.13}$$

By (1.8), we have

$$\sigma = D\varepsilon(\mathbf{u})$$

where

$$D = \begin{pmatrix} \lambda+2\mu & \lambda & 0 \\ \lambda & \lambda+2\mu & 0 \\ 0 & 0 & \mu \end{pmatrix}, \qquad \mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.$$

So (4.13) can also be written as

$$\iint_\Omega \varepsilon(\mathbf{u})^T D\varepsilon(\mathbf{v})dxdy \quad = \quad \iint_\Omega \mathbf{f}^T\mathbf{v}dxdy - \int_\Gamma \mathbf{q}^T\mathbf{v}ds. \tag{4.14}$$

or

$$\begin{aligned} &\iint_\Omega \{(\lambda+2\mu)\varepsilon_{11}(\mathbf{u})\varepsilon_{11}(\mathbf{v}) + \lambda\varepsilon_{22}(\mathbf{u})\varepsilon_{11}(\mathbf{v}) \\ &\quad +\lambda\varepsilon_{11}(\mathbf{u})\varepsilon_{22}(\mathbf{v}) + (\lambda+2\mu)\varepsilon_{22}(\mathbf{u})\varepsilon_{22}(\mathbf{v}) + \mu\gamma_{12}(\mathbf{u})\gamma_{12}(\mathbf{v})\}dxdy \\ &\quad = \iint_\Omega (f_1v_1 + f_2v_2)dxdy - \int_\Gamma \{q_1v_1 + q_2v_2\}\, ds. \end{aligned} \tag{4.15}$$

4.2 Generation of FE source code by FEPG

FEPG (Finite Element Program Generator) [17] is a very powerful tool to generate the FE source code. Here we will give some template files for writing the GCN, GIO and PDE files.

4.2.1 Writing GCN, GIO files

We can use the same GCN, GIO as in Section 3.4. The filenames of GCN and GIO files must be the same. We list them below. The comments follow '\'.

GIO file "le.gio":

```
intf          \ Define the name of PDE file we will use.

#elemtype t3  \ Define the element shape (t for triangular elements)
              \and node number (3) of each element.
2dxy          \ Define the coordinate type.
              \This is 2d Cartesian coordinates.
```

GCN file "le.gcn":

```
defi            \Define solution algorithm for
                \every decoupled physical field.
a ell &         \The field "a" associates with the first
                \PDE name in the GIO file.
                \"ell" is the algorithm for solving the
                \elliptic equation provided by FEPG.

STARTilu a      \Initialize the solver which uses the conjugate
                \gradient method with ilu as pre-conditioner.
SOLVilu a       \Solve the linear system using the conjugate
                \gradient method with ilu as pre-conditioner.
```

We can select a different solver by replacing `ilu` with `sin` (symmetric direct method), `sor` (SOR iterative method) etc.

4.2.2 Writing PDE file

Next, we will show the PDE (FBC) files based on the weak form (4.14) or (4.15). We notice that there are both volume integrals and a boundary integral in the weak form. We need to write a PDE file for volume integrals and an FBC file for the boundary integral. If there is no boundary integral in the weak form, then we do not need to write the FBC file. The name of the PDE (FBC) file is given in the first part of the GIO file. We name it "intf.pde" ("intf.fbc").

Here is the file "intf.pde". It uses the same rules as in Section 3.4. We will use "`pl, pm, f1, f2`" to denote λ, μ, f_1, f_2 respectively. FEPG uses F77 naming conventions. This means that all variables starting with the letters i-n are integers and all others are real. To make it easy to write the "stif" segment, we introduce some auxiliary functions "`exx, eyy, exy`" which will be defined in the "func" segment. In that segment, "`exx, eyy`" and "`exy`" are define as $\varepsilon_{11}, \varepsilon_{22}$ and γ_{12} respectively.

PDE file "intf.pde":

```
defi
disp u1 u2            \Define the name of the unknown function.
coor x y              \Define the coordinate system.
func exx eyy exy      \Give the names of some auxiliary functions
                      \which will be defined in "func" segment.
shap %1 %2            \Define the shape function.
                      \The parameters "%1, %2" are retrieved from
                      \"#elemtype t3" in GIO file.
gaus %3               \Define type of integration
```

```
                      \(nodal integral or Gaussian integral).
                      \"%3" is also retrieved from
                      \"#elemtype t3" in GIO file.
                      \For Gaussian integral, it gives
                      \the number of integral points.
mate pl pm f1 f2 1.0;1.0;0.0;0.0;
                      \Define the material parameters "pl, pm, f1, f2"
                      \which will be used in weak form and their
                      \default values.
                      \The values can be modified in the following
                      \"stif" segment.

func                  \Define auxiliary functions.
exx=+[u1/x]

eyy=+[u2/y]

exy=+[u1/y]+[u2/x]

stif                      \Define stiffness matrix based on weak form.
                          \Insert fortran code following the "$c6"
                          \to define the material parameters.
                          \"$c6" means beginning with 6 spaces
                          \which is the fortran 77 convention.
$c6 if(imate.eq.1) then   \Inside the interface.
$c6   pl=1d0              \Give the values of the material parameters.
$c6   pm=1d0
$c6   f1=-8d0
$c6   f2=-8d0
$c6 else                  \Outside the interface.
$c6   pl=1d2
$c6   pm=1d2
$c6   f1=-8d0
$c6   f2=-8d0
$c6 endif
$c6 factvol = pl+2.0*pm
dist=+[exx;exx]*factvol+[exx;eyy]*pl
+[eyy;exx]*pl+[eyy;eyy]*factvol+[exy;exy]*pm
                         \Correspond to volume integrals
                         \at the Left Hand Side of the weak form.

load=+[u1]*f1+[u2]*f2    \Correspond to volume integrals
                         \at the Right Hand Side of the weak form.
```

```
end
```

"`disp u1 u2`" gives the primary variables. Here "(`u1, u2`)" corresponds to $\mathbf{u} = (u_1, u_2)$.

The keyword "`dist`" in stif segment is to define the volume integrals in the left hand side of the weak form which contains the unknown function. The first part "`+[exx;exx]*factvol`" corresponds to

$$\iint_\Omega (\lambda + 2\mu)\varepsilon_{11}(\mathbf{u})\varepsilon_{11}(\mathbf{v})dxdy$$

in the weak form (4.15). In the expression, the "`[ ; ]`" corresponds to volume integral. The "`exx`" preceding the semicolon '`;`' corresponds to $\varepsilon(\mathbf{u})$. The "`exx`" following the semicolon '`;`' corresponds to $\varepsilon(\mathbf{v})$. The "`factvol`" is defined as $\lambda + 2\mu$. The other parts have similar meanings.

The volume integral in the right hand side of the weak form (4.15) is

$$\iint_\Omega (f_1 v_1 + f_2 v_2)dxdy.$$

We write it in the FEPG language as

```
load=+[u1]*f1+[u2]*f2
```

Here, "(`u1, u2`)" corresponds to the test function $\mathbf{v} = (v_1, v_2)$. The "`f1, f2`" corresponds to f_1, f_2 respectively. The "`[ ]`" corresponds to the volume integral.

4.2.3 Writing FBC file

What is left in the weak form (4.15) is the boundary integral

$$-\int_\Gamma \{q_1 v_1 + q_2 v_2\}\, ds. \tag{4.16}$$

Here we assume

$$q_1 = \frac{2(3x^2 + 2xy + y^2)}{\sqrt{x^2 + y^2}}, \qquad q_2 = \frac{2(3y^2 + 2xy + x^2)}{\sqrt{x^2 + y^2}}.$$

We need to introduce a local coordinate system first.

The triangulation in Chapter 2 also gives the discretization of the interface. The interface Γ is approximated with a series of line elements $\{l_1, l_2, \ldots, l_{n_i}\}$ where n_i is the total number of line elements. In general, we approximate (4.16) with

$$-\sum_{i=1}^{n_i} \int_{l_i} \{q_1 v_1 + q_2 v_2\}\, ds. \tag{4.17}$$

For a typical line element l_i with nodes $1, 2$ (see Fig 4.1), we define the local coordinate system centered at node 1 as

$$\begin{cases} \xi = (x - x_1)\cos\theta + (y - y_1)\sin\theta, \\ \eta = -(x - x_1)\sin\theta + (y - y_1)\cos\theta, \end{cases} \tag{4.18}$$

where θ is the angle between the x-axis and $\overrightarrow{12}$ (see Fig. 4.1), and (x_1, y_1) is the coordinates of node 1.

If we set

$$T = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \tag{4.19}$$

$$\boldsymbol{\xi} = \begin{pmatrix} \xi \\ \eta \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} x - x_1 \\ y - y1 \end{pmatrix}, \tag{4.20}$$

then

$$\boldsymbol{\xi} = T\mathbf{x}.$$

T is an orthogonal matrix and often called the transformation matrix.

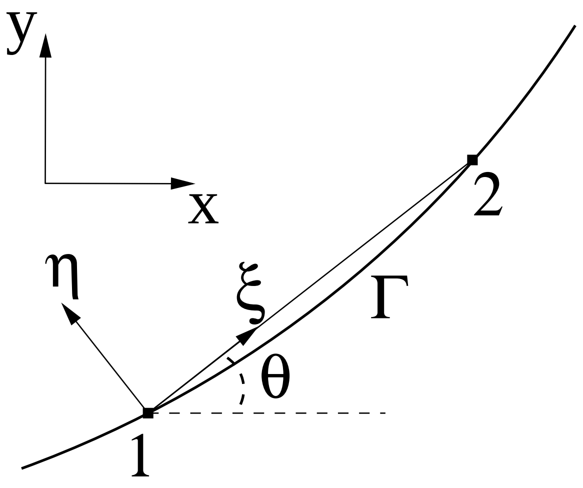


Figure 4.1: Local coordinate system for a typical line element l_i with nodes $1, 2$.

To write the system in an FBC file, we need to represent

$$\mathbf{q} = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}$$

in the local coordinate system. If we denote it as $\mathbf{q}'$, then

$$\mathbf{q}' = T\mathbf{q}$$

There is no direct way to input $\mathbf{q}$ given in the global coordinate system into the FEPG system. We will do this indirectly by revising the generated Fortran source code. The FEPG system provides the transformation matrix T. We need to pass this matrix to the subroutine “all2” which is generated from the FBC file below.

FBC file “intf.fbc”:

```
defi
disp u1,u2
coor x                  \local coordinates
shap %1 %2
gaus %3
coef gx,gy              \global coordinates
mate q1 q2 0d0;0d0;

stif
$c6 pl=1d0              \Define vector q.
$c6 pm=1d0
$c6 el=dsqrt(gx*gx+gy*gy)
$c6 gq1 =-2d0*((pl+2d0*pm)*gx*gx+pm*gy*gy+(pl+pm)*gx*gy)/el
$c6 gq2 =-2d0*((pl+2d0*pm)*gy*gy+pm*gx*gx+(pl+pm)*gx*gy)/el
$c6
$c6 q1=t(1,1)*gq1+t(1,2)*gq2    \Transformation of vector q.
$c6 q2=t(2,1)*gq1+t(2,2)*gq2
dist=[u1;u1]*0.0

load=+[u1]*q1+[u2]*q2

end
```

The format of FBC files is nearly the same as that for PDE files. Since the elements treated in FBC files are boundary elements, the dimension is one order lower than that for volume elements. The “`dist=+[u1;u1]*0d0`” means there is no boundary integral in the left hand side of the weak form. The “`load=+[u1]*q1+[u2]*q2`” corresponds to the boundary integral (4.16).

4.2.4 Generation of FE source code

After preparing the “le.gcn”, “le.gio” , “intf.pde” and “intf.fbc” files, we can just run “gio le” under the FEPG environment to generate all the Fortran 77 source code.

In Section 4.2.3, we need to use the transformation matrix T which is not defined in the FBC file. We need to modify the Fortran 77 source code to make it work. We change the flow of subroutine “agl2” by forming the matrix T before calling the subroutine “all2” and pass the matrix T as a parameter for the subroutine “all2”.

Then we can insert some Fortran code to customize the output. FEPG also integrates a pre/post processor GID [5] into its system. It provides a user-friendly graphical user interface for geometrical modeling, data input and visualization of results for all types of numerical simulation programs.

4.3 Numerical results

We present numerical experiments for the model problem (1.15)-(1.18) using the proposed locally modified mesh method. We use the standard piecewise linear Galerkin finite element method. Finite Element Program Generator (FEPG) [17] is used to generate the stiffness matrix and the load vector. We use the incomplete LU decomposition (ILU) along with the conjugate gradient method to solve the resulting linear system of equations.

Most of the computations are done on an Intel Core 2 Duo 2.0GHz processor with 2GB of RAM. For a 320×320 mesh, it takes about 0.2 seconds to generate the mesh. Most of the simulations are done within seconds.

We present some examples (Examples 1, 2) with known exact solutions so that we can validate the computer codes and check the convergence rate. The interface Γ is the circle $x^2 + y^2 = 1/4$ within the computational domain $\Omega = (-1, 1) \times (-1, 1)$. For a 320×320 mesh, it takes 53 seconds to solve the resulting linear system using the ILU. In Example 3, we consider a more realistic problem with a non-convex general interface. In this example, we do not have an analytic solution. It takes about 3 minutes to solve the resulting linear system using the ILU.

Let $\tilde{u}$ be the finite element solution obtained from our method and

$$\tilde{u}(x,y) = \sum_{ij} U_{ij}\varphi_{ij}(x,y)$$

where φ_{ij} is the piecewise linear basis function centered at point (x_i, y_j). We define e_∞, e_1 as the errors in the L^∞ and H^1 norms, respectively, thus

$$e_\infty = \max_{i,j} |u(x_i, y_j) - U_{ij}|, \tag{4.21}$$

$$e_1 = \sqrt{\iint_\Omega \left\{(u-\tilde{u})^2 + (u_x - \tilde{u}_x)^2 + \ (u_y - \tilde{u}_y)^2\right\} d\Omega}. \tag{4.22}$$

We also scale e_∞, e_1 as follows to get the relative errors,

$$r_\infty = \frac{e_\infty}{\max\limits_{i,j} |u(x_i, y_j)|}, \tag{4.23}$$

$$r_1 = \frac{e_1}{\max\limits_{i,j} |u(x_i, y_j)|}. \tag{4.24}$$

We will use this scaled error measurement only for Example 2.

Example 1 *The first example is taken from [8]. The parameters are $\lambda^- = \mu^- = 1$, and $\lambda^+ = \mu^+ = 100$. The body force term $\mathbf{f} = (f_1, f_2)$ and the Dirichlet boundary condition $\mathbf{u}_0 = (u_{0_1}, u_{0_2})$ are given from the exact solution $\mathbf{u} = (u_1, u_2)$:*

$$u_1 = u_2 = \begin{cases} r^2, & r \le R; \\ \dfrac{r^2}{100} + (1 - \dfrac{1}{100})R^2, & \textit{otherwise.} \end{cases}$$

where $R = \frac{1}{2}$ and $r = \sqrt{x^2 + y^2}$.

Note that the solution $\mathbf{u}$ is continuous across the interface $r = 1/2$ and $[\sigma\mathbf{n}]|_\Gamma = 0$, i.e., the exact solution satisfies the homogeneous jump conditions.

In Table 4.1 we show the result of a grid refinement analysis of the finite element solutions using the locally modified meshes. In the table, the first column N is the number of intervals in the x and the y directions; the second column is the error in the maximum norm. The third column is the ratio of the two consecutive errors. The ratio approaches number four for quadratic convergence and number two for linear convergence. The fourth column is the error in the H^1 norm. The other columns in this table and other tables have the similar meanings. The finite element methods are both roughly second order accurate in the maximum norm and first order accurate in the H^1 norm as expected.

Table 4.1: A grid refinement analysis in the maximum and H^1 norms of the finite element method using the locally modified meshes for Example 1.

N	e_∞	ratio	e_1	ratio
20	2.4271E-03		1.1957E-02	
40	8.8263E-04	2.7498	4.4212E-03	2.7045
80	2.5543E-04	3.4555	1.7337E-03	2.5502
160	5.7255E-05	4.4613	6.3617E-04	2.7252
320	1.4157E-05	4.0443	2.2759E-04	2.7952

Example 2 *We set $\lambda^- = \mu^- = 1$ and $\lambda^+ = \mu^+ = b$ for some $b > 0$. The body force term $\mathbf{f} = (f_1, f_2)$ and the Dirichlet boundary condition $\mathbf{u}_0 = (u_{0_1}, u_{0_2})$ are derived from the exact solution $\mathbf{u} = (u_1, u_2)$:*

$$u_1 = u_2 = \begin{cases} r^2 + \log(1 + r^2), & r \le R; \\ \dfrac{r^2}{b/2} + \left(1 - \dfrac{1}{b/2}\right) R^2 + \dfrac{\log(1 + r^2)}{b} + \left(1 - \dfrac{1}{b}\right) \log(1 + R^2), & \textit{otherwise.} \end{cases}$$

where again $R = \frac{1}{2}$ and $r = \sqrt{x^2 + y^2}$.

For any b, the solution $\mathbf{u}$ is continuous across the interface $r = 1/2$ and $[\sigma \mathbf{n}]|_\Gamma = \mathbf{q} = (q_1, q_2)$,

$$q_1 = \frac{2(3x^2 + 2xy + y^2)}{\sqrt{x^2 + y^2}}, \qquad q_2 = \frac{2(3y^2 + 2xy + x^2)}{\sqrt{x^2 + y^2}}.$$

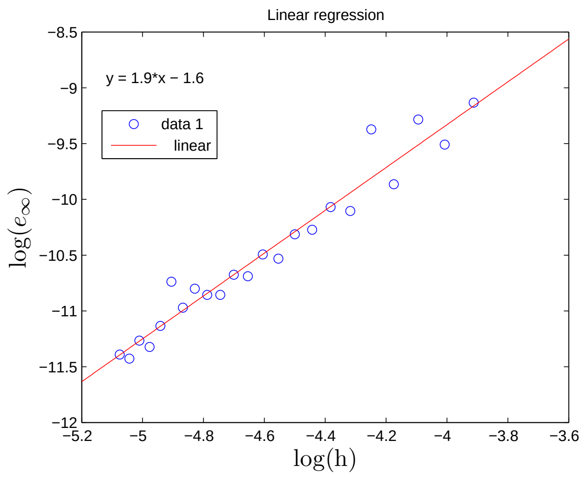
In Table 4.2 we show the result of a grid refinement analysis of our method. We can see once again that we have first order accuracy in the H^1 norm, and second order accuracy in L^∞ norm as we would expect. The results also show that our method can handle small or large jumps in the physical parameters and in the solution. The convergence order is almost independent of the jumps in the coefficients for the relative error. Note that the magnitude of the solution is getting larger as the parameter $b = \lambda^+ = \mu^+$ is getting smaller, or vice versa. Nevertheless, our method can handle these situations very well.

In Figure 4.2, we show a linear regression analysis for $\lambda^- = \mu^- = 1$, $\lambda^+ = \mu^+ = 0.1$ for the finite element solution using the locally modified mesh. From the slopes, we can observe that the convergence order is 1.9 with the mesh size varying from $N = 100$ to $N = 320$ according to $N = 100 + 10k$, $k = 0, \ldots, 22$ and 2 with the mesh varying according to $N = 100 + 20k$, $k = 0, \ldots, 11$.

Table 4.2: A grid refinement analysis in the maximum and H^1 norms of the finite element method using the locally modified meshes for Example 2 with different jump ratio.

	$\lambda^+ = \mu^+ = 10$			
N	r_∞	ratio	r_1	ratio
20	3.7083E-3		2.3834E-2	
40	1.4063E-3	2.6369	8.6210E-3	2.7646
80	4.1231E-4	3.4108	3.2720E-3	2.6348
160	9.2478E-5	4.4585	1.1867E-3	2.7573
320	2.3622E-5	3.9149	4.2083E-4	2.8198
	$\lambda^+ = \mu^+ = 1000$			
N	r_∞	ratio	r_1	ratio
20	9.2119E-3		4.4262E-2	
40	3.3598E-3	2.7418	1.6187E-2	2.7344
80	9.8181E-4	3.4220	6.2464E-3	2.5915
160	2.2333E-4	4.3962	2.2636E-3	2.7596
320	5.4229E-5	4.1183	8.0108E-4	2.8257
	$\lambda^+ = \mu^+ = 0.1$			
N	r_∞	ratio	r_1	ratio
20	1.8451E-3		1.0097E-2	
40	5.3678E-4	3.4373	3.6190E-3	2.7901
80	1.5390E-4	3.4878	1.3593E-3	2.6625
160	4.2369E-5	3.6324	4.1334E-4	3.2885
320	1.1251E-5	3.7658	1.4863E-4	2.7810
	$\lambda^+ = \mu^+ = 0.001$			
N	r_∞	ratio	r_1	ratio
20	1.9727E-3		1.0169E-2	
40	5.8714E-4	3.3598	3.6591E-3	2.7792
80	1.6904E-4	3.4734	1.3775E-3	2.6564
160	4.6768E-5	3.6144	4.1916E-4	3.2863
320	1.2460E-5	3.7535	1.5067E-4	2.7820

(a).



(b).

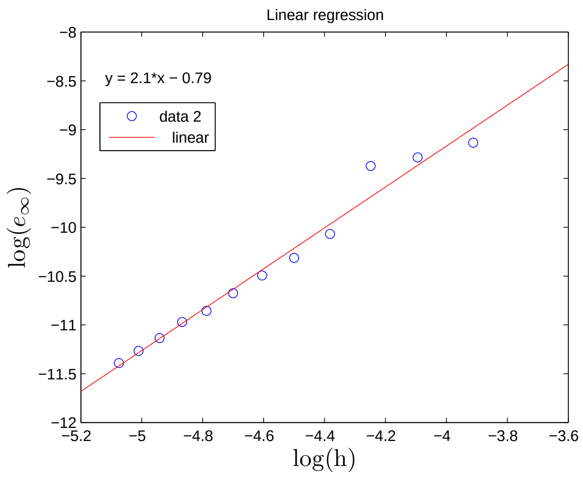


Figure 4.2: A linear regression analysis for $\lambda^- = \mu^- = 1$, $\lambda^+ = \mu^+ = 0.1$ in log-log scale (a) with the mesh varying according to $N = 100 + 10k$, $k = 0, \ldots, 22$. The slope(convergence order) is 1.9; (b) with the mesh varying according to $N = 100 + 20k$, $k = 0, \ldots, 22$. The slope is 2.1.

Example 3 *We consider a plate consisting of two materials with a non-circular interface in a state of plane strain. The interface is given by $r = 0.5 + 0.2 \sin\theta$ in polar coordinates. The plate is $2m \times 2m$ with the bottom fixed and $1e7$ N/m force applied on the top; see Fig. 1.2. The Young's modulus and Poisson's ratio for the inner and outer materials are $E^- = 1e9$ Pa, $\nu^- = 0.3$, and $E^+ = 1e10$ Pa, $\nu^+ = 0.3$ respectively. We wish to compute the displacement distribution of the plate.*

The purpose of this example is to show that our method can handle complicated geometry and large coefficients. Under the pressure, the plate will undergo a compression in the y-direction and will react by stretching in the x-direction in order to reduce the change in the volume.

We set the origin at the center of the plate and use a 320×320 mesh to do the simulation. We use

$$\mu = \frac{E}{2(1+\nu)}, \qquad \lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \tag{4.25}$$

to compute the Lamé coefficients μ^-, λ^- and μ^+, λ^+ for this is a plane strain problem.

In Fig. 4.4, we show a 40×40 mesh for an illustration. In Fig. 4.5, we show the contour plot for the displacement distribution. The results are symmetric along the y-axis for the x-component u_1 of displacement and symmetric along the x-axis for the y-component u_2 of

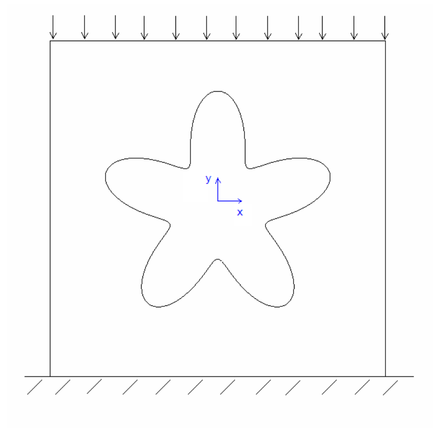


Figure 4.3: A rectangular plate with a non-circular interface $r = 0.5 + 0.2\sin\theta$. A uniform force is applied at the top of the boundary.

displacement. In Fig. 4.6, we show the positions of the original and deformed plates.

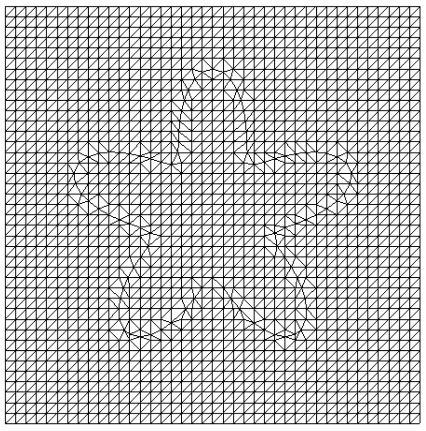
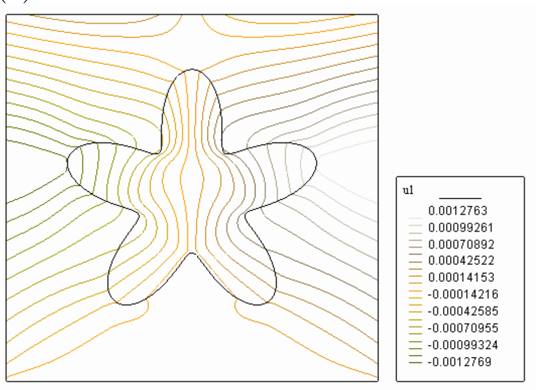


Figure 4.4: A locally modified mesh for a non-circular interface.

We also consider nearly incompressible cases for Example 3 with different Poisson's ratios ν. Notice that in (4.25), if ν is close to 0.5, then λ goes to infinity, or so called the nearly incompressible case. We first set $\nu^- = \nu^+ = 0.4, 0.42, 0.44, 0.45,$ and 0.46. It takes about $191, 215, 231, 700,$ and 4000 seconds respectively to converge using the `ilu` solver. It takes much more time for larger Poisson's ratios since the matrix become worse. So we use the `sin` (symmetric direct method) solver to compute the solution for the case $\nu^- = \nu^+ = 0.49$. The CPU time is about 450 seconds. We show the deformation in Fig. 4.7.

(a).



(b).

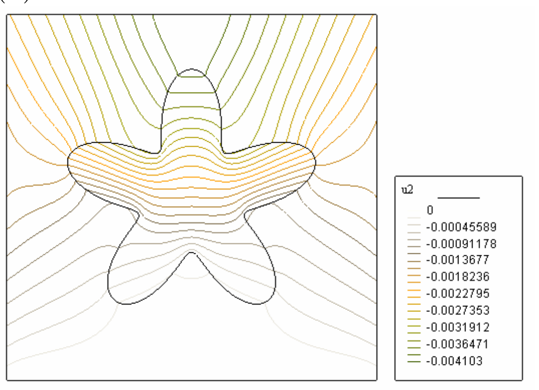


Figure 4.5: Contour plots of the displacement distribution $\mathbf{u} = (u_1, u_2)$. (a) the contour plot of u_1; (b) the contour plot of u_2.

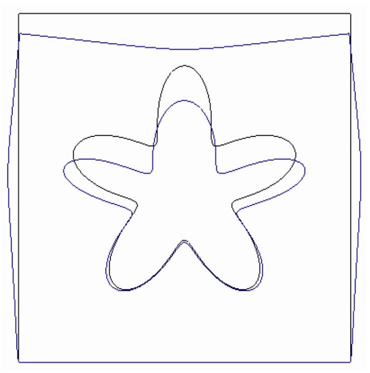


Figure 4.6: The original and deformed plates for case $\nu^- = \nu^+ = 0.3$. Note that the displacement is very small. To make it visible, we have rescaled the displacement 50 times larger in the plot. The result agrees with physical intuition and reasoning.

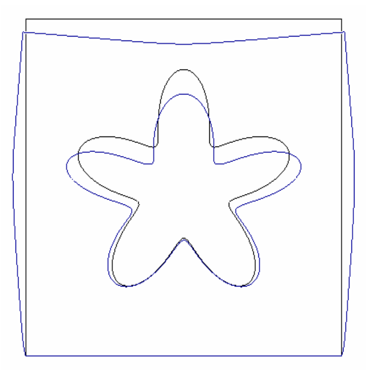


Figure 4.7: The original and deformed plates for case $\nu^- = \nu^+ = 0.49$. Note that the displacement is very small. To make it visible, we have rescaled the displacement 50 times larger in the plot. The result agrees with physical intuition and reasoning.

Chapter 5

A new second order finite difference method for interface problems

In this chapter, we will present a new second order finite difference method for interface problems. Our model problem is as follows:

$$\begin{aligned} \nabla \cdot (\beta \, \nabla u) &= f(\vec{x}), \quad \vec{x} = (x, y) \in \Omega \setminus \Gamma && (5.1) \\ [u]_\Gamma &= 0, && (5.2) \\ \left[\beta \frac{\partial u}{\partial n} \right]_\Gamma &= C, && (5.3) \\ u|_{\partial\Omega} &= u_0(\vec{x}), && (5.4) \end{aligned}$$

where $\Omega = \Omega^+ \bigcup \Omega^-$, $\Gamma \in C^2$ is an interface between the subdomains Ω^+ and Ω^-; see Fig. 5.1 for an illustration. The diffusion coefficient β can have a finite jump across the interface Γ. We assume that $\beta \geq \beta_{min} > 0$, and f is a bounded function, u_0 is a given function, and C is a constant. "$[\cdot]$" denotes the jump of the function which is defined as the difference of the limiting values from the outside of the interface to the inside, and n is the unit normal direction of the interface Γ pointing outward.

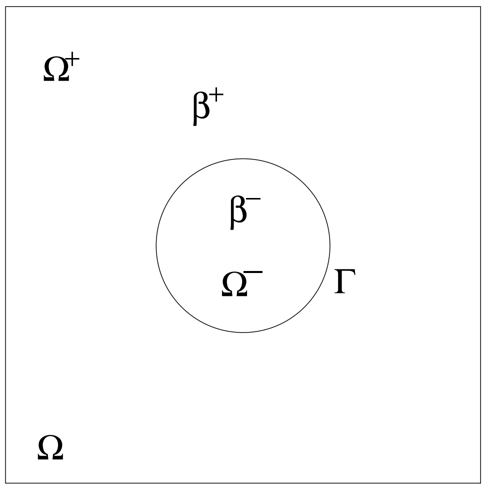


Figure 5.1: A rectangular domain Ω with an immersed interface Γ.

5.1 The idea of the new method

For non-interface problems, we know that the standard 5-point finite difference scheme based on a Cartesian grid will produce second order results. But for interface problems, this method will lose accuracy when the interface crosses the 5-point stencil. We call these grid points irregular grid points. The local truncation errors are no longer h^2 for the standard finite difference methods where h is the step size of the Cartesian grid. We want to take advantage of the simplicity of the standard finite difference methods. So we introduce some extra variables which we call ghost values to make the 5-point stencil work. The price of this is that we need to find an equation for each ghost value. We notice that there are two interface conditions $[u]_\Gamma = 0$, $\left[\beta \frac{\partial u}{\partial n}\right]_\Gamma = C$. We need to carefully choose one of the interface conditions for each ghost value.

5.2 A new finite difference discretization

Again, we will use a Cartesian grid. Let (x_i, y_j), $0 \le i \le M,\ 0 \le j \le N$ be the grid points. The step sizes are the same denoted as h in both the x and the y directions. Those grid points marked by circles or boxes, in Fig 5.2, are irregular grid points for which the interface crosses the standard 5-point finite difference stencil. Other grid points are called regular grid points.

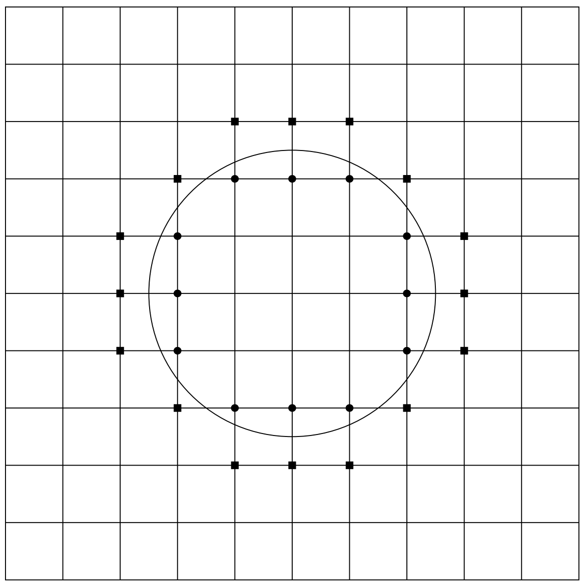


Figure 5.2: A mesh with irregular grid points marked.

5.2.1 The finite difference scheme for regular grid points

Let $U_{i,j}$ be the approximation to the solution $u(x,y)$ of (5.1)-(5.4) at a point (x_i, y_j). For regular points with index (i,j), we use the standard centered 5-point finite difference scheme,

$$\begin{aligned}\frac{1}{h}\Bigg\{&\left(\beta_{i+1/2,j}\frac{(U_{i+1,j}-U_{i,j})}{h}-\beta_{i-1/2,j}\frac{(U_{i,j}-U_{i-1,j})}{h}\right)\\&+\left(\beta_{i,j+1/2}\frac{(U_{i,j+1}-U_{i,j})}{h}-\beta_{i,j-1/2}\frac{(U_{i,j}-U_{i,j-1})}{h}\right)\Bigg\}=f_{ij},\end{aligned}\tag{5.5}$$

where

$$\beta_{i+1/2,j}=\beta\left(x_i+\frac{h}{2},y_j\right),\quad \beta_{i,j+1/2}=\beta\left(x_i,y_j+\frac{h}{2}\right),$$

and so on. The local truncation errors are $O(h^2)$ at these regular points.

If we rearrange (5.5), we have

$$\begin{aligned}\frac{1}{h^2}\{&\beta_{i-1/2,j}U_{i-1,j}+\beta_{i+1/2,j}U_{i+1,j}+\beta_{i,j-1/2}U_{i,j-1}+\beta_{i,j+1/2}U_{i,j+1}\\&-(\beta_{i-1/2,j}+\beta_{i+1/2,j}+\beta_{i,j-1/2}+\beta_{i,j+1/2})U_{i,j}\}=f_{i,j}.\end{aligned}\tag{5.6}$$

5.2.2 The finite difference scheme for irregular grid points

If (x_i, y_j) is an irregular grid point, i.e., the grid points in the centered 5-point stencil are from both sides of the interface, then we will introduce some ghost values to make the 5-point stencil work properly. Without loss of generality, we assume that an irregular point (x_i, y_j) lies outside the interface, or in "+" side; see Fig. 5.3. Because the grid point

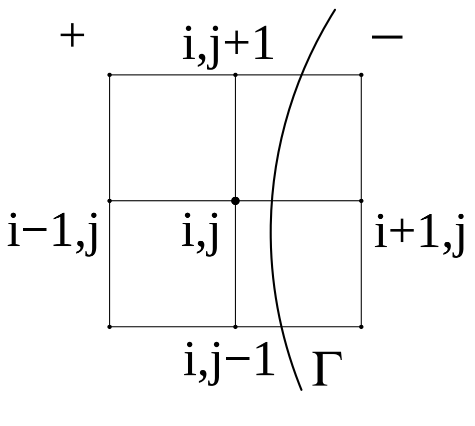


Figure 5.3: A typical irregular grid point (i, j).

(x_{i+1}, y_j) is not on the same side of the interface as the grid point (x_i, y_j), we need a ghost value $U^g{}_k$ at that point (x_{i+1}, y_j), where there is a one-to-one mapping between the index $(i + 1, j)$ and k. So we have an equation for the irregular grid point (x_i, y_j),

$$\begin{aligned}\frac{1}{h^2}\Big\{&\beta_{i-1/2,j}U_{i-1,j} + \beta^*_{i+1/2,j}U^g{}_k + \beta_{i,j-1/2}U_{i,j-1} + \beta_{i,j+1/2}U_{i,j+1}\\ &-(\beta_{i-1/2,j} + \beta^*_{i+1/2,j} + \beta_{i,j-1/2} + \beta_{i,j+1/2})U_{i,j}\Big\} = f_{i,j} \qquad (5.7)\end{aligned}$$

Note that we replace $U_{i+1,j}$ with $U^g{}_k$ and $\beta_{i+1/2,j}$ with $\beta^*_{i+1/2,j}$ in (5.6). If the grid point $(x_{i+1/2}, y_j)$ is on the same side as the point (x_i, y_j), then $\beta^*_{i+1/2,j} = \beta_{i+1/2,j}$. Otherwise we evaluate the extension of $\beta(x, y)$ from outside the interface, where the grid point (x_i, y_j) lies, at the point $(x_i + h/2, y_j)$.

Now we have obtained all the equations for all grid points. We can write (5.6)-(5.7) in a matrix-vector form

$$A\,U + B\,U^g = F \qquad (5.8)$$

where U, U^g and F are the vectors formed by $\{U_{i,j}\}, \{U^g{}_k\}$, and $\{f_{i,j}\}$, respectively.

5.2.3 The least squares interpolation for ghost values

Now we need to set up an equation for $U^g{}_k$ (the ghost value for grid point (x_{i+1}, y_j)) to complete the linear system. Recall that we have not used the interface conditions

$$[u]_\Gamma = 0, \qquad \left[\beta\frac{\partial u}{\partial n}\right]_\Gamma = C.$$

One of them will be chosen to be satisfied at its orthogonal projection point of (x_{i+1}, y_j) on the interface based on the values of β^+ and β^- at that projection point. Since we need to deal with the flux jump condition given in the normal direction, we will use the local coordinates in the normal and tangential directions.

Let (x^*, y^*) be the orthogonal projection point of the grid point (x_{i+1}, y_j) on the interface. Then the local coordinate system is defined as

$$\begin{cases} \xi = (x - x^*)\cos\theta + (y - y^*)\sin\theta, \\ \eta = -(x - x^*)\sin\theta + (y - y^*)\cos\theta, \end{cases} \tag{5.9}$$

where θ is the angle between the x-axis and the normal direction n of the interface Γ pointing outward. See Fig. 5.4 for an illustration.

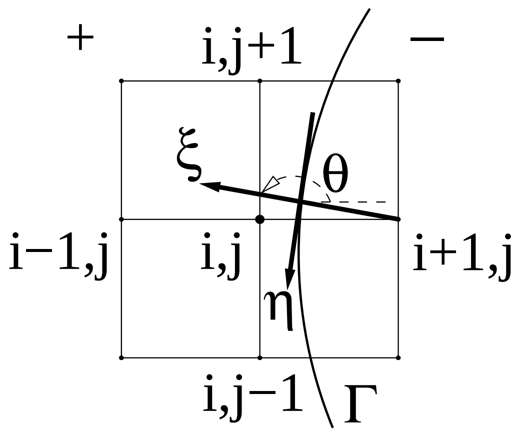


Figure 5.4: The local coordinate system in the normal and tangential directions. The "+" side lies outside the interface.

With the local coordinate system, we can set up the equation for the ghost value ${U^g}_k$.

Let β^+, β^- be the limiting values of $\beta(x, y)$ at the projection point (x^*, y^*) from outside and inside the interface, respectively. The grid point (x_{i+1}, y_j) lies the "−" side. We need a ghost value from the "+" side to make the 5-point stencil work at grid point (x_i, y_j). If $\beta^+ > \beta^-$, then we choose $\left[\beta\frac{\partial u}{\partial n}\right]_\Gamma = C$. Otherwise we choose $[u]_\Gamma = 0$. We will explain this below in the remark. We consider two different cases.

Case 1, $\beta^+ > \beta^-$:

We have

$$\beta^+ \frac{\partial u}{\partial n}^+ - \beta^- \frac{\partial u}{\partial n}^- = C.$$

We use ${U^g}_k$, the ghost value at (x_{i+1}, y_j), and $U_{i_p^+, j_p^+}$, $p = 2, \ldots, n_g$ to carry out the second order least squares interpolation for u_n^+, where the grid points $(x_{i_p^+}, y_{j_p^+})$, $p = 2, \ldots, n_g$ are in the neighborhood of the projection point (x^*, y^*) and lie outside the interface.

We use $U_{i_p^-, j_p^-}, p = 1, \ldots, n_g$ for the second order least squares interpolation for u_n^-, where the grid points $(x_{i_p^-}, y_{j_p^-}), p = 1, \ldots, n_g$ are in the neighborhood of the projection point (x^*, y^*) and lie inside the interface.

So we have an equation for the ghost value ${U^g}_k$,

$$\beta^+ \left(\gamma_1^+ {U^g}_k + \sum_{p=2}^{n_g} \gamma_p^+ U_{i_p^+, j_p^+} \right) - \beta^- \left(\sum_{p=1}^{n_g} \gamma_p^- U_{i_p^-, j_p^-} \right) = C. \tag{5.10}$$

Remark: We want the magnitude of the coefficient of ${U^g}_k$ as large as possible. So we choose $\left[\beta \frac{\partial u}{\partial n}\right]_\Gamma = C$ when $\beta^+ > \beta^-$. Another consideration is to control the local truncation error. If we divide the above equation by β^+, then the local truncation error is $O(h^2) - \frac{\beta^-}{\beta^+} O(h^2)$. Since $\frac{\beta^-}{\beta^+} < 1$ when $\beta^+ > \beta^-$, the local truncation error is well controlled by the second order least squares interpolation error.

Case 2, $\beta^+ <= \beta^-$:

We have

$$u^+ - u^- = 0.$$

We use ${U^g}_k$, the ghost value at (x_{i+1}, y_j), and $U_{i_p^+, j_p^+}$, $p = 2, \ldots, n_g$ to carry out the second order least squares interpolation for u^+, where the grid points $(x_{i_p^+}, y_{j_p^+})$, $p = 2, \ldots, n_g$ are in the neighborhood of the projection point (x^*, y^*) and lie outside the interface.

We use $U_{i_p^-, j_p^-}$, $p = 1, \ldots, n_g$ for the second order least squares interpolation for u^-, where the grid points $(x_{i_p^-}, y_{j_p^-})$, $p = 1, \ldots, n_g$ are in the neighborhood of the projection point (x^*, y^*) and lie inside the interface.

So we have an equation for the ghost value ${U^g}_k$,

$$\left(\gamma_1^+ U_{gk} + \sum_{p=2}^{n_g} \gamma_p^+ U_{i_p^+, j_p^+} \right) - \left(\sum_{p=1}^{n_g} \gamma_p^- U_{i_p^-, j_p^-} \right) = 0. \tag{5.11}$$

Next, we will show how to use ${U^g}_k$, $U_{i_p^+, j_p^+}$, $p = 2, \ldots, n_g$ to carry out the second order least squares interpolation for u_n^+ and $U_{i_p^-, j_p^-}$, $p = 1, \ldots, n_g$ for u_n^-.

The interpolation scheme for approximating u_n^+ is

$$u_n^+ \simeq \gamma_1^+ U^g{}_k + \sum_{p=2}^{n_g} \gamma_p^+ U_{i_p^+, j_p^+}. \tag{5.12}$$

Using the local coordinate system (5.9) centered at the projection point (x^*, y^*), we can denote the local coordinates of (x_{i+1}, y_j) as (ξ_0, η_0), $(x_{i_p^+}, y_{j_p^+})$ as (ξ_p, η_p). We have the following from the Taylor expansion at (x^*, y^*) or $(0,0)$ in the local coordinates:

$$\begin{aligned} U^g{}_k &\simeq u^+(x_{i+1}, y_j) = u^+(\xi_0, \eta_0) \\ &= u^+ + \xi_0 u_\xi^+ + \eta_0 u_\eta^+ + \frac{1}{2}\xi_0^2 u_{\xi\xi}^+ + \xi_0\eta_0 u_{\xi\eta}^+ + \frac{1}{2}\eta_0^2 u_{\eta\eta}^+ + O(h^3), \qquad (5.13) \\ U_{i_p^+, j_p^+} &\simeq u(x_{i_p^+}, y_{j_p^+}) = u(\xi_p, \eta_p) \\ &= u^+ + \xi_p u_\xi^+ + \eta_p u_\eta^+ + \frac{1}{2}\xi_p^2 u_{\xi\xi}^+ + \xi_p\eta_p u_{\xi\eta}^+ + \frac{1}{2}\eta_p^2 u_{\eta\eta}^+ + O(h^3), \qquad (5.14) \end{aligned}$$

where $u^+, u_\xi^+, \ldots, u_{\eta\eta}^+$ are evaluated at the local coordinates $(0,0)$ or (x^*, y^*) in the original coordinate system. Below we will use $U_{i_1^+, j_1^+}$ to denote $U^g{}_k$. Then we can just focus on (5.14).

In the local coordinate system, the equation (5.1) can be written as

$$\beta^+(u_{\xi\xi}^+ + u_{\eta\eta}^+) + \beta_\xi^+ u_\xi^+ + \beta_\eta^+ u_\eta^+ = f^+.$$

Therefore, we have

$$u_{\xi\xi}^+ = \frac{f^+}{\beta^+} - u_{\eta\eta}^+ - \frac{\beta_\xi^+}{\beta^+} u_\xi^+ - \frac{\beta_\eta^+}{\beta^+} u_\eta^+. \tag{5.15}$$

Plugging (5.15) into (5.14), we get

$$\begin{aligned} U_{i_p^+, j_p^+} &\simeq u(x_{i_p^+}, y_{j_p^+}) = u(\xi_p, \eta_p) \\ &= u^+ + (\xi_p - \frac{1}{2}\xi_p^2 \frac{\beta_\xi^+}{\beta^+}) u_\xi^+ + (\eta_p - \frac{1}{2}\xi_p^2 \frac{\beta_\eta^+}{\beta^+}) u_\eta^+ \\ &\quad + \xi_p\eta_p u_{\xi\eta}^+ + (\frac{1}{2}\eta_p^2 - \frac{1}{2}\xi_p^2) u_{\eta\eta}^+ + \frac{1}{2}\xi_p^2 \frac{f^+}{\beta^+} + O(h^3). \qquad (5.16) \end{aligned}$$

To minimize the interpolation error, we should set the following linear system of equations for the coefficients $\{\gamma_p^+\}$:

$$\begin{aligned}\sum_{p=1}^{n_g}\gamma_p^+ &= 0\\ \sum_{p=1}^{n_g}\gamma_p^+(\xi_p-\frac{1}{2}\xi_p^2\frac{\beta_\xi^+}{\beta^+}) &= 1\\ \sum_{p=1}^{n_g}\gamma_p^+(\eta_p-\frac{1}{2}\xi_p^2\frac{\beta_\eta^+}{\beta^+}) &= 0\\ \sum_{p=1}^{n_g}\gamma_p^+\xi_p\eta_p &= 0\\ \sum_{p=1}^{n_g}\gamma_p^+(\frac{1}{2}\eta_p^2-\frac{1}{2}\xi_p^2) &= 0.\end{aligned} \tag{5.17}$$

Note that we have five equations in the above system. We should set $n_g \geq 5$ in order to have a consistent system. In our numerical tests, we set $n_g = 8$ or 9. Thus (5.17) is an underdetermined system of linear equations that has an infinite number of solutions in general. We solve (5.17) using the singular value decomposition (SVD). The SVD subroutine is taken from Linpack. The SVD solution has the smallest 2-norm among all feasible solutions,

$$\sum_{p=1}^{n_g}(\gamma_p^{+*})^2 = \min_{\gamma_p^+}\sum_{p=1}^{n_g}(\gamma_p^+)^2, \quad \text{subject to (5.17).}$$

We will drop the superscript '*' for simplicity. Note that we have an extra constant term $\frac{1}{2}\xi_p^2\frac{f^+}{\beta^+}$ in (5.16), so we need to modify (5.12) to

$$u_n^+ + \sum_{p=1}^{n_g}\gamma_p^+\frac{1}{2}\xi_p^2\frac{f^+}{\beta^+} \simeq \gamma_1^+ {U^g}_k + \sum_{p=2}^{n_g}\gamma_p^+ U_{i_p^+,j_p^+}. \tag{5.18}$$

Similarly, for

$$u_n^- \simeq \sum_{p=1}^{n_g}\gamma_p^- U_{i_p^-,j_p^-}, \tag{5.19}$$

we denote the local coordinates of $(x_{i_p^-}, y_{j_p^-})$ as $(\xi_{n_g+p}, \eta_{n_g+p})$ and expand $U_{i_p^-,j_p^-}, p = 1, n_g$ from the "$-$" side. Following the above procedure, by changing all the "+" superscripts to "$-$" superscripts and (ξ_p, η_p) to $(\xi_{n_g+p}, \eta_{n_g+p})$ in the linear system (5.17), we can determine the coefficients γ_p^- and get

$$u_n^- + \sum_{p=1}^{n_g} \gamma_p^- \frac{1}{2}\xi_{n_g+p}^2 \frac{f^-}{\beta^-} \simeq \sum_{p=1}^{n_g} \gamma_p^- U_{i_p^-,j_p^-}. \tag{5.20}$$

Instead of (5.10), we obtain an equation for the ghost variable ${U^g}_k$ under the condition $\beta^+ > \beta^-$:

$$\begin{aligned} &\beta^+ \left(\gamma_1^+ {U^g}_k + \sum_{p=2}^{n_g} \gamma_p^+ U_{i_p^+,j_p^+} \right) - \beta^- \left(\sum_{p=1}^{n_g} \gamma_p^- U_{i_p^-,j_p^-} \right) \\ &= C + \beta^+ \left(\sum_{p=1}^{n_g} \gamma_p^+ \frac{1}{2}\xi_p^2 \frac{f^+}{\beta^+} \right) - \beta^- \left(\sum_{p=1}^{n_g} \gamma_p^- \frac{1}{2}\xi_{n_g+p}^2 \frac{f^-}{\beta^-} \right). \end{aligned} \tag{5.21}$$

For the case when $\beta^+ \le \beta^-$, we will obtain a similar equation

$$\begin{aligned} &\left(\gamma_1^+ {U^g}_k + \sum_{p=2}^{n_g} \gamma_p^+ U_{i_p^+,j_p^+} \right) - \left(\sum_{p=1}^{n_g} \gamma_p^- U_{i_p^-,j_p^-} \right) \\ &= \left(\sum_{p=1}^{n_g} \gamma_p^+ \frac{1}{2}\xi_p^2 \frac{f^+}{\beta^+} \right) - \left(\sum_{p=1}^{n_g} \gamma_p^- \frac{1}{2}\xi_{n_g+p}^2 \frac{f^-}{\beta^-} \right). \end{aligned} \tag{5.22}$$

The difference from the above derivation is the right hand side vector in the linear system (5.17). Here we need to approximate u^+ and u^-. The right hand side vector should be the column vector $(1,0,0,0,0)^T$ instead of the vector $(0,1,0,0,0)^T$.

Finally, summarizing the procedure above, we can come up with

$$C\,U + D\,U^g = F^g. \tag{5.23}$$

By combining (5.8) and (5.23), we obtain a complete linear system

$$\begin{cases} A\,U + B\,U^g &= F \\ C\,U + D\,U^g &= F^g, \end{cases} \tag{5.24}$$

where A is a symmetric negative definite matrix and D is a diagonal matrix by our construction.

5.3 The Schur complement

If we multiply the top equation in (5.24) by $C\ A^{-1}$ and then subtracting from the bottom equation, we obtain

$$(D - C\ A^{-1}B)U^g = F^g - C\ A^{-1}F. \tag{5.25}$$

The matrix $D - C\ A^{-1}B$ is called the Schur complement of the block A. It is an $n_g \times n_g$ matrix for U^g, a much smaller linear system compared with the one for U. Since D is a diagonal matrix, we will use it as a right preconditioner. We solve

$$(I - C\ A^{-1}BD^{-1})y = F^g - C\ A^{-1}F \quad \text{with} \ \ U^g = D^{-1}y. \tag{5.26}$$

Since the matrix $I - C\ A^{-1}BD^{-1}$ is nonsymmetric in most cases, we will solve (5.26) by the GMRES iterative method [23].

The GMRES method requires only matrix-vector multiplication. Given a vector y_i, we compute $y_i^1 = D^{-1}y_i$, $y_i^2 = By_i^1$, $y_i^3 = A^{-1}y_i^2$, and $y_i^4 = Cy_i^3$, then subtract y_i^4 from y_i, to obtain

$$y_{i+1} = y_i - y_i^4.$$

There are many ways to compute $y_i^3 = A^{-1}y_i^2$. In general, we need to solve the linear system

$$Ay_i^3 = y_i^2 \tag{5.27}$$

for y_i^3. The matrix A has many nice properties:

1. invertible
2. symmetric negative definite
3. diagonally dominant
4. reducible

   If we order all the outer grid points before the inner grid points, we have

$$PAP^T = \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix}, \tag{5.28}$$

   where A_1, A_2 are irreducible and have the Properties 1-3, and P is some permutation matrix.

Based on these properties, we can use a direct method such as the LU decomposition or iterative methods such as the preconditioned conjugate gradient method, or multigrid method [3] based on the matrix A. Or we can solve two smaller systems using the LU decomposition, or the preconditioned conjugate gradient method based on (5.28).

In this chapter, we will use a blackbox multigrid solver DMG9V [6] to solve (5.27). This solver is developed for the scheme that involves the 9-point stencil. We use it for convenience.

5.4 Numerical results

We present numerical experiments using the proposed finite difference method.

We present an example used in Chapter 3.5 with a known exact solution from [15] so that we can obtain the error and the convergence rate. The interface Γ is the circle $x^2 + y^2 = 1/4$ within the computational domain $\Omega = (-1,1) \times (-1,1)$. The equation is

$$-(\beta u_x)_x - (\beta u_y)_y = f(x,y) - C \int_\Gamma \delta(\vec{x} - \vec{X}(s))ds, \tag{5.29}$$

with

$$f(x,y) = -8(x^2 + y^2) - 4, \tag{5.30}$$

and

$$\beta(x,y) = \begin{cases} x^2 + y^2 + 1, & x^2 + y^2 \le \frac{1}{4}, \\ b, & x^2 + y^2 > \frac{1}{4}. \end{cases} \tag{5.31}$$

The Dirichlet boundary condition is given by the exact solution

$$u(x,y) = \begin{cases} r^2, & \text{if} \quad r \le \frac{1}{2}, \\ (1 - \frac{1}{8b} - \frac{1}{b})/4 + (\frac{r^4}{2} + r^2)/b + C\log(2r)/b/2, & \text{if} \quad r > \frac{1}{2}. \end{cases} \tag{5.32}$$

where $r = \sqrt{x^2 + y^2}$. For any b and C, the solution u is continuous across the interface $r = 1/2$ and $[\beta u_n]|_\Gamma = C$.

We first consider two cases with $b = 10$ and $b = 1000$ with $C = 0.1$. In both cases β is discontinuous.

The maximum error over all grid points is defined as

$$e_\infty = \max_{i,j} |u(x_i, y_j) - U_{ij}|, \tag{5.33}$$

and we scale e_∞ as follows to get the relative error,

$$r_\infty = \frac{e_\infty}{\max\limits_{i,j} |u(x_i, y_j)|}. \tag{5.34}$$

We set the max number of the GMRES iterations to be 100.

Case 1, $C = 0.1,\ b = 10$**.**

In Table 5.1 we show the result of a grid refinement analysis of the finite difference solutions. In the table, the second row contains the GMRES error tolerances to reduce the norm of the residual by the factors of 10^{-5} and 10^{-7}. The first column N is the number of intervals in the x and the y directions; the second column is the error in the maximum norm. The third column is the ratio of the two consecutive errors. The ratio approaches number four for quadratic convergence and number two for linear convergence. The fourth column is the number k of GMRES iterations. The other columns in this table and other tables have the similar meanings.

The finite difference method is roughly second order accurate in the maximum norm for suitable GMRES error tolerance as expected.

Table 5.1: A grid refinement analysis in the maximum norm of the finite difference method for Case 1.

$C = 0.1,\ b = 10$						
	GMRES error $1e{-5}$			GMRES error $1e{-7}$		
N	e_∞	ratio	k	e_∞	ratio	k
20	4.2998E-3		9	4.2957E-3		13
40	4.1346E-4	10.3996	13	4.1223E-4	10.4206	17
80	1.1218E-4	3.6857	17	1.0975E-4	3.7561	24
160	2.8796E-5	3.8957	24	2.7949E-5	3.9268	36
320	7.1458E-6	4.0298	33	6.7241E-6	4.1565	48

Case 2, $C = 0.1,\ b = 1000$**.**

The coefficient β has a bigger jump compared with Case 1. We show the same grid refinement results in Table 5.2. We need a smaller GMRES error tolerance to get the second order convergence rate.

Table 5.2: A grid refinement analysis in the maximum norm of the finite difference method for Case 2.

$C = 0.1,\ b = 1000$						
	GMRES error $1e-5$			GMRES error $1e-7$		
N	e_∞	ratio	k	e_∞	ratio	k
20	1.4585E-3		8	1.4362E-3		12
40	2.1754E-4	6.7045	12	2.1103E-4	6.8057	16
80	1.2089E-4	1.7995	16	4.2766E-5	4.9345	23
160	4.1394E-5	2.9205	23	9.8058E-6	4.3613	54
320	1.2865E-4	0.3218	31	2.3460E-6	4.1798	44

5.5 The new method with a good guess

For both cases in previous subsection, we have verified that our method is a second order method. Now we need to control the number of GMRES iterations. The idea is to use a good guess obtained by some simple methods such as harmonic averaging [1, 26, 27], or Peskin's immersed boundary method [20, 21]. These methods are generally first order methods. To illustrate the idea, we consider a simple case when $C = 0$ for which we can use *harmonic averaging* to get a good guess. For the harmonic averaging, we use (5.5) for all grid points and redefine the $\beta_{i+\frac{1}{2},j}$, $\beta_{i,j+\frac{1}{2}}$ as below:

$$\beta_{i+\frac{1}{2},j} = \begin{cases} \beta(x_i + h/2, y_j), & \beta \text{ is smooth in } [x_i, x_{i+1}]; \\ \left(\dfrac{1}{h}\displaystyle\int_{x_i}^{x_{i+1}} \beta^{-1}(x, y_j)dx\right)^{-1}, & \text{otherwise.} \end{cases}$$

$$\beta_{i,j+\frac{1}{2}} = \begin{cases} \beta(x_i, y_j + h/2), & \beta \text{ is smooth in } [y_j, y_{j+1}]; \\ \left(\dfrac{1}{h}\displaystyle\int_{y_j}^{y_{j+1}} \beta^{-1}(x_i, y)dx\right)^{-1}, & \text{otherwise.} \end{cases}$$

After we obtain U_0 from the harmonic averaging, we can use the second equation in (5.24) to obtain the initial guess

$$y_0 = D\ U_0^g\ = F^g - C\ U_0$$

for the linear system (5.26).

We consider two cases for $b = 10$, $b = 1000$ with $C = 0$. We set the GMRES error tolerance to be 10^{-5} or 10^{-7}. We show the results from our method with zeros as the initial guess, the harmonic averaging and our method with the harmonic averaging as the initial guess in Table 5.3 and Table 5.4. From Table 5.3, we see that our method is second order accurate and the harmonic averaging is first order accurate. The method with the harmonic averaging as the initial guess requires fewer GMRES iterations than the method with zeros as the initial guess. From Table 5.4, the harmonic averaging is nearly second order accurate. Our method is second order accurate with much less error. The method with the harmonic averaging as the initial guess is much better than the method with zeros. So we will use it for the other examples.

Table 5.3: A grid refinement analysis in the maximum norm of the finite difference method for $C = 0,\ b = 10$.

$C = 0,\ b = 10$, GMRES error $1e{-}5$								
	with initial zeros			HA		with initial HA		
N	e_∞	ratio	k	e_∞	ratio	e_∞	ratio	k
20	4.3430E-3		9	4.6449E-3		4.3405E-3		7
40	4.5682E-4	9.5070	13	2.4262E-3	1.9145	4.5484E-4	9.5429	8
80	1.2764E-4	3.5790	17	1.0861E-3	2.2339	1.2393E-4	3.6701	10
160	3.3115E-5	3.8544	24	4.8931E-4	2.2197	3.1066E-5	3.9892	12
320	8.2476E-6	4.0151	33	2.2840E-4	2.1423	6.6660E-6	4.6604	14

Table 5.4: A grid refinement analysis in the maximum norm of the finite difference method for $C = 0,\ b = 1000$.

$C = 0,\ b = 1000$, GMRES error $1e{-}7$								
	with initial zeros			HA		with initial HA		
N	e_∞	ratio	k	e_∞	ratio	e_∞	ratio	k
20	1.4363E-3		12	3.2352E-3		1.4373E-3		9
40	2.1132E-4	6.7968	16	1.3592E-3	2.3802	2.1183E-4	6.7852	11
80	4.3038E-5	4.9101	68	4.2586E-4	3.1917	4.2829E-5	4.9459	11
160	9.9196E-6	4.3387	33	1.2139E-4	3.5082	9.5899E-6	4.4661	13
320	2.3288E-6	4.2595	100	3.3165E-5	3.6602	2.3164E-6	4.1400	16

We also compare harmonic averaging with our method with harmonic averaging as the initial guess for the cases $C = 0,\ b = 0.1$ and $C = 0,\ b = 0.001$. We set the GMRES error tolerance to be 10^{-4} for these cases and use the relative error r_∞ to show the results. In Table 5.5 and Table 5.6, we also observe a second order pattern for our method with harmonic averaging as the initial guess.

Table 5.5: A grid refinement analysis in the maximum norm of the finite difference method for $C = 0,\ b = 0.1$.

$C = 0,\ b = 0.1$, GMRES error $1e{-}4$					
	HA		with initial HA		
N	r_∞	ratio	r_∞	ratio	k
21	2.4260E-3		1.2353E-02		8
41	1.9517E-3	1.2430	2.3693E-03	5.2138	11
81	1.1450E-3	1.7045	5.6211E-04	4.2150	14
161	6.1808E-4	1.8525	1.3889E-04	4.0472	21
321	3.2563E-4	1.8981	3.1969E-05	4.3446	29

Table 5.6: A grid refinement analysis in the maximum norm of the finite difference method for $C = 0,\ b = 0.001$.

$C = 0,\ b = 0.001$, GMRES error $1e{-}4$					
	HA		with initial HA		
N	r_∞	ratio	r_∞	ratio	k
21	2.2187E-3		1.2025E-2		9
41	1.7974E-3	1.2344	2.2838E-3	5.2653	13
81	1.0469E-3	1.7169	5.3352E-4	4.2806	16
161	5.6346E-4	1.8580	1.3036E-4	4.0927	26
321	2.9688E-4	1.8979	2.6186E-5	4.9782	38

Harmonic averaging can not give a good initial guess for problems where $C \neq 0$. We need to use other methods such as Peskin's immersed boundary method to obtain a good initial guess. A more natural way is to use our method on a coarse grid which consists of the even-numbered grid points of the fine grid (original grid) with stepsize h to obtain a better initial guess. If we obtain $(U^g)^{2h}$, U^{2h} from a coarse grid, then we can use a second order least squares interpolation to get a better initial guess $(U^g)^h$ for the fine grid. We show the results for the cases $C = 0.1,\ b = 0.1$ and $C = 0.1,\ b = 0.001$. We set the GMRES error tolerance to be 10^{-5} for these cases and use the relative error r_∞ to show the results. In Table 5.7, we also observe a second order pattern for our method with the coarse grid solution as the initial guess. For the case $C = 0.1$ and $b = 0.001$, on an Intel(R) Xeon(R) 3.0GHz processor with 32GB of RAM, the CPU time of our method with the coarse grid solution as the initial guess with a 320×320 grid is 7 seconds.

Table 5.7: A grid refinement analysis in the maximum norm of the finite difference method with the coarse grid solution as the initial guess.

GMRES error $1e{-}5$						
	$C = 0.1,\ b = 0.1$			$C = 0.1,\ b = 0.001$		
N	r_∞	ratio	k	r_∞	ratio	k
41	2.0259E-3		15	1.9470E-3		17
81	4.6540E-4	4.3531	20	4.4413E-4	4.3839	22
161	1.1314E-4	4.1134	26	1.0814E-4	4.1070	31
321	2.5047E-5	4.5171	34	2.3518E-5	4.5982	41

Chapter 6

Conclusions and future work

In this thesis, we have proposed finite element methods for solving elliptic and elasticity problems with interfaces using locally modified meshes. The locally modified mesh is easy to generate from a Cartesian mesh in which some nodal points near the interface are moved on the interface. For elliptic problems with interfaces, the finite element method using the modified mesh is compared with another modified mesh, locally enriched triangulations also based on a Cartesian mesh, in which new nodal points are added on the intersection between grid lines and the interface. The locally enriched triangulation is also easy to generate. Based on our numerical experiments both meshes lead to second order accurate solution for elliptic problems with interfaces. With the adaptation, the triangles are quasi-uniform and the mesh maintains the Cartesian structure. In particular, the Cartesian structure is useful when storing matrices and developing iterative solvers. Neither of these properties hold for the locally enriched meshes.

We have also proposed a new second order finite difference method for solving elliptic problems with interfaces which can be easily extended to elasticity problems with interfaces. By introducing some ghost values where the 5-point stencil contains the interface, we can use the standard finite difference scheme at all the grid points. With ghost values, we use the interface conditions to set up the equations for the ghost values. We test our method with different initial guess and find that our method with the coarse grid solution as the initial guess is a good choice for interface problems.

There are a lot can be done in future work:

1. Apply the finite element methods with a locally modified mesh to the moving interface

problems.

2. Extend the finite element methods to three dimensional (3D) problems. The interfaces are surfaces in 3D instead of curves in 2D. To form a locally modified mesh, starting form a Cartesian mesh, we propose to move the grid points near the interface onto it. This will lead to a hexahedral grid. We will develop a systematical way to divide each of the hexahedrons into six tetrahedrons.

3. Use a two-grid correction or even a multigrid method to reduce the number of the GMRES iterations.

Bibliography

[1] J. B. Bell, C. N. Dawson, and G. R. Shubin. An unsplit, higher order Godunov method for scalar conservation laws in multiple dimensions. *J. Comput. Phys.*, 74:1–24, 1988.

[2] C. Börgers. A triangulation algorithm for fast elliptic solvers based on domain imbedding. *SIAM J. Numer. Anal.*, 27:1187–1196, 1990.

[3] W.L. Briggs, V.E. Henson, and S.F. McCormick. *A Multigrid tutorial 2nd ed.* SIAM, Philadelphia, 2000.

[4] P.G. Ciarlet. *The Finite Element Method for Elliptic Problems.* North-Holland Publishing Co., Amsterdam, 1978.

[5] CIMNE. Gid. http://gid.cimne.upc.es.

[6] D. De Zeeuw. Matrix-dependent prolongations and restrictions in a blackbox multigrid solver. *J. Comput. Appl. Math.*, 33:1–27, 1990.

[7] K. Feng and Z. Shi. *Mathematical Theory of Elastic Structures.* Science Press, 1996.

[8] Y. Gong. Immersed-interface finite-element methods for elliptic and elasticity interface problems. North Carolina State University, 2007.

[9] Y. Gong, B. Li, and Z. Li. Immersed-interface finite-element methods for elliptic interface problemswith non-homogeneous jump conditions. *SIAM J. Numer. Anal.*, 46:472–495, 2008.

[10] K. Ito, Z. Qiao, and J. Toivanen. A domain decomposition solver for acoustic scattering by elastic objects in layered media. *J. Comput. Phs.*, 227:8685–8698, 2008.

[11] K. Ito and J. Toivanen. Preconditioned iterative methods on sparse subspaces. *Appl. Math. Lett.*, 19:1191–1197, 2006.

[12] K. Ito and J. Toivanen. A fast iterative solver for scattering by elastic objects in layered media. *Appl. Numer. Math*, 57:811–820, 2007.

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

[14] Z. Li. An overview of the immersed interface method and its applications. *Taiwanese J. Mathematics*, 7:1–49, 2003.

[15] 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.

[16] Z. Li, T. Lin, and X. Wu. New Cartesian grid methods for interface problem using finite element formulation. *Numer. Math.*, 96:61–98, 2003.

[17] Beijing Fegensoft Co. Ltd. Finite Element Program Generator. http://www.fegensoft.com.

[18] N. I. Muskhelishvili. *Some Basic Problems of the Mathematical Theory of Elasticity.* Groningen, P. Noordhoff, 1963.

[19] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces.* Springer, New York, 2002.

[20] C. S. Peskin. Numerical analysis of blood flow in the heart. *J. Comput. Phys.*, 25:220–252, 1977.

[21] C. S. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[22] W. Proskurowski and O.B. Widlund. A finite element capacitance matrix method for the neumann problem for laplace's equation. *SIAM J. Sci. Statist. Comput.*, 1:410–425, 1980.

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

[24] M.H. Sadd. *Elasticity: Theory, Application and Numerics.* Amsterdam: Elsevier Butterworth Heinemann, 2005.

[25] J. A. Sethian. *Level Set Methods and Fast Marching Methods.* Cambridge University Press, 2nd edition, 1999.

[26] G. R. Shubin and J. B. Bell. An analysis of the grid orientation effect in numerical simulation of miscible displacement. *Comp. Meth. Appl. Mech. Eng.*, 47:47–71, 1984.

[27] A. N. Tikhonov and A. A. Samarskii. Homogeneous difference schemes. *USSR Comput. Math. and Math. Phys.*, 1:5–67, 1962.

[28] S. P. Timoshenko and J. N. Goodier. *Theory of Elasticity.* McGraw-Hill, New York, 1985.

[29] Wikipedia. Finite element method. http://en.wikipedia.org/wiki/Finite_element_method.

[30] Wikipedia. Level set method. http://en.wikipedia.org/wiki/Level_set_method.

[31] H. Xie, K. Ito, Z. Li, and J. Toivanen. A finite element method for interface problems with locally modified triangulation. *Contemporary Mathematics*, 466:179–190, 2008.

[32] X. Yang. Immersed interface method for elasticity problems with interfaces. North Carolina State University, 2004.

[33] D. Yu. A system of plane elasticity canonical integral equations and its application. *J. Comp. Math.*, 4:200–211, 1986.

[34] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *The Finite Element Method: Its Basis And Fundamentals, 6th ed.* Elsevier, Oxford, 2005.