Abstract

Chen Guo. Immersed Interface Method for Biharmonic equations on irregular domain and its applications. (Under the direction of Zhilin Li.)

This thesis presents a fast algorithm for solving two-dimensional biharmonic equations on irregular domains. To avoid mesh generation difficulties associated with unstructured, body fitted grid, the irregular domain is embedded into a uniform Cartesian grid. The biharmonic equation is decomposed into two coupled Poisson equations.

The solution of the coupled Poisson system depends on the Laplacian Δu on the boundary. We use a weighted least squares interpolation to approximate the Laplacian on the boundary from inside of the region. The accuracy of the interpolation scheme turns out to be a crucial step in solving the biharmonic problem for our algorithm.

The resulting linear system involves both the solution and the Laplacian on the boundary. In order to take advantage of fast Poisson solvers, we use Generalized Minimum Residue method to solve for Δu, and use the Immersed Interface Method to solve the coupled Poisson problem. Putting all these techniques together, we get a second order fast algorithm for solving biharmonic equations on irregular domains. Numerical analysis show the algorithm is very stable and the number of iterations of our method seems to be independent of the mesh size.

We also investigate some applications of the proposed fast algorithm for incompressible Stokes equations and the biharmonic equation with a linear parameter.

IMMERSED INTERFACE METHOD FOR BIHARMONIC EQUATIONS ON IRREGULAR DOMAIN AND ITS APPLICATIONS

BY
Guo Chen

A DISSERTATION SUBMITTED TO THE GRADUATE FACULTY OF
NORTH CAROLINA STATE UNIVERSITY
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF MATHEMATICS

RALEIGH
December 18, 2003

APPROVED BY:

CHAIR OF ADVISORY COMMITTEE	

To my wife and my parents
for their endless love.

Biography

Guo Chen is from Nanxian, Hunan province, the People's Republic of China. He spent four years for his undergraduate study at Nankai University, Tianjin and received his Bachelor of Science degree in 1996. After graduation, he joined the graduate program in Beijing University and received his Master of Science degree in computational mathematics three years later. In the fall of 1999, he entered the Ph.D program in Applied Mathematics with a concentration in Computational Mathematics at North Carolina State University in Raleigh.

Acknowledgements

I would like to express my sincerest appreciation to my research advisor, Dr. Zhilin Li, for his patience and support throughout my career as a graduate student. Our innumerable times discussing and questioning the problems and cheering any achievements together will leave a lasting impression. Most importantly, Zhilin's clarity and insight into problems are inspiring. I look forward to collaborating with him on future research and expect constant support from him.

I would also like to thank the other members of my advisory committee: Dr. Ito, Dr. Lin, Dr. Lubkin. They have been very helpful during my four years in the department. Whether it was through rigorous coursework or suggestions of interesting problems to look at, I am appreciative to all their help.

I would also like to thank the faculty, students and staff at the Department of Mathematics, North Carolina University for providing a stimulating environments. I especially want to thank my officemates, past and present, for their friendship throughout the time we have been students here.

My wife Hailin and my parents Jinsheng Chen and Lihua Yuan have been wonderfully supportive of my work. I can not thank them enough.

Table of Contents

List of Figures **vii**

List of Tables **ix**

1 Introduction **1**
1.1 Some Application Examples 1
1.2 Overview of numerical methods for the biharmonic equations 4
1.3 Our strategy 8
1.3.1 The sketch of our method 8
1.3.2 Decomposing the system 8
1.3.3 Uniform Cartesian grid 10
1.3.4 Represent the boundary 10
1.4 Outline of the thesis 11

2 The Immersed Interface method for biharmonic equations **13**
2.1 Coupled System 14
2.2 The discrete approximation 16
2.2.1 Cartesian grid and level set function 16
2.2.2 The finite difference scheme and the treatment of boundary 17
2.3 The algorithm description 22
2.3.1 Matrix representation 22
2.3.2 The weighted least squares interpolation 24
2.3.3 Selecting grid points for interpolation 27

3 The algorithm description **33**
3.1 Schur complement system and the GMRES method 33
3.2 Some implementation details 35
3.3 Numerical results 36
3.3.1 Algorithm efficiency analysis 45
3.3.2 Summary of the computation scheme 48

4 Applications of the biharmonic solver **51**
4.1 Solving incompressible Stokes equation on an irregular domain 51
4.2 Solving the biharmonic equation with a linear parameter 54

5 Conclusions and future work **60**
5.1 Thesis conclusions 60
5.2 Future work 61

List of References **62**

List of Figures

1.1 A diagram of the problem: a biharmonic equation on an irregular domain. . 2
1.2 A classical 13 point finite difference stencil. 5
1.3 A 9-point stencil. 7

2.1 A diagram of extending $u(x, y)$ to $[a, b] \times [c, d]$ 15
2.2 An uniform cartesian grid and the immersed boundary 17
2.3 An irregular grid point and its projection on the boundary. 19
2.4 Comparison of two typical delta functions. Left: the hat function. Right: Peskin's delta function 25
2.5 Selected grid points for interpolation: Peskin's approach 28
2.6 Selected grid points for interpolation: the one-side approach 29
2.7 Selected grid points for interpolation when (a) $ctg(\theta) \in [0, \frac{1}{10})$ and (b) $ctg(\theta) \in [\frac{1}{10}, \frac{1}{5})$ 30
2.8 Selected grid points for interpolation when (a) $ctg(\theta) \in [\frac{1}{5}, \frac{1}{4}]$ and (b) $ctg(\theta) \in [\frac{1}{4}, \frac{1}{3}]$ 30
2.9 Selected grid points for interpolation when (a) $ctg(\theta) \in [\frac{1}{3}, \frac{1}{2}]$ and (b) $ctg(\theta) \in [\frac{1}{2}, \frac{4}{7}]$ 31
2.10 Selected grid points for interpolation when (a) $ctg(\theta) \in [\frac{4}{7}, \frac{4}{5}]$ and (b) $ctg(\theta) \in [\frac{4}{5}, 1]$ 31

3.1 The exact solution for example (3.1). 38
3.2 Error of the computed solution for example (3.1) when n=m=64. 38
3.3 Error of the computed solution for example (3.1) when n=m=128. 39
3.4 The linear regression analysis on the errors for example (3.1). 39
3.5 The exact solution for example (3.2). 40
3.6 Error of the computed solution for example (3.2) with 64×32 mesh. 41
3.7 Error of the computed solution for example (3.2) with 128×64 mesh. . . . 41
3.8 The linear regression analysis on the errors for example (3.2). 42
3.9 The exact solution for example (3.3). 43
3.10 Error of the computed solution for example (3.3) when n=m=64. 44
3.11 Error of the computed solution for example (3.3) when n=m=128. 44
3.12 The linear regression analysis on the errors for example (3.3). 45
3.13 The exact solution for example (3.4). 46

3.14 Error of the computed solution for example (3.4) when n=m=64. 47
3.15 Error of the computed solution for example (3.4) when n=m=128. 47

4.1 The exact velocity field for example 4.10–4.12. 53
4.2 Plot of the exact stream function ψ for example 4.10–4.12. 54
4.3 Plot of the exact solution **u** for example 4.10–4.12. 55
4.4 The computed velocity field for example 4.10–4.12 when n=m=64. 56
4.5 The computed velocity field for example 4.10–4.12 when n=m=128. 56
4.6 The exact solution for example 4.21–4.22 58
4.7 The error plot for example 4.21–4.22 when n=m=64 59
4.8 The error plot for example 4.21–4.22 when n=m=128 59

List of Tables

2.1 Interpolation errors comparison between simple one-side scheme and one-side walk through scheme 32

3.1 Grid refinement analysis for example (3.1). 37
3.2 A grid refinement analysis for example (3.2). 40
3.3 A grid refinement analysis for example (3.3). 43
3.4 A grid refinement analysis for example (3.4). 46
3.5 The number of iterations and CPU time for Example (3.1) 49
3.6 The number of iterations and CPU time for Example (3.2) 49
3.7 The number of iterations and CPU time for Example (3.3) 49
3.8 The number of iterations and CPU time for Example (3.4) 49
3.9 The CPU time in dealing with irregular grid points and the CPU time for the linear solver for Example (3.1) and (3.2) 50
3.10 The CPU time in dealing with irregular grid points and the CPU time for the linear solver for Example (3.3) and (3.4) 50

4.1 A grid refinement analysis of $\mathbf{u} = (u, v)$ for example 4.10–4.12. 53
4.2 A grid refinement analysis of example 4.21–4.22 58

© Copyright 2003
by
Guo Chen

Chapter 1

Introduction

In this thesis, we develop an efficient numerical method to solve a biharmonic equation defined on an irregular domain. The Problem can be described as follows:

Let Ω be a bounded open set in R^2 with a smooth boundary. Consider the following Biharmonic boundary value problem problem:

$$\begin{aligned} \Delta^2 u(x,y) &= f(x,y) \qquad (x,y) \in \Omega \\ u(x,y) &= g_1(x,y) \qquad (x,y) \in \partial\Omega \\ u_n(x,y) &= g_2(x,y) \qquad (x,y) \in \partial\Omega \end{aligned} \tag{1.1}$$

where u_n denote the exterior normal derivative on $\partial\Omega$.

Our approach is based on finite difference method. It is of second order accuracy and the algorithm is optimal, requiring only $O(N \log N)$ arithmetic operations for a mesh of N grid points.

1.1 Some Application Examples

Equation (1.1) is called the (first) Dirichlet boundary value problem for the biharmonic operator

$$\Delta^2 \equiv \nabla^4 = \frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial x^2 \partial y^2} + \frac{\partial^4}{\partial y^4} \tag{1.2}$$

and this problem arises in several fields of physical applications. Classical examples can be found in elasticity and in fluid mechanics.

In fluid mechanics, the solution $u(x,y)$ of equation (1.1) can be used to describe the streamfunction of an incompressible two-dimensional creeping flow (Reynolds number zero). Efficient numerical methods for this problem can also be used when trying to solve the non-linear Navier-Stokes equation describing incompressible flow at nonzero Reynolds number. The biharmonic operator appears linearly in this equation when using the streamfunction formulation. For example, we consider incompressible viscous flow in a two dimensional

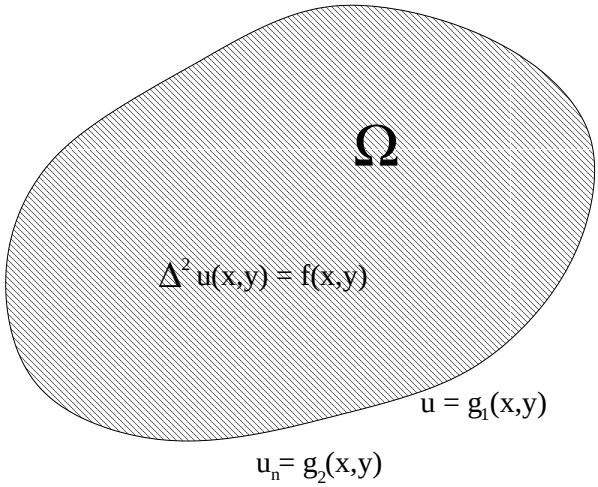


Figure 1.1: A diagram of the problem: a biharmonic equation on an irregular domain.

domain. The motion of the fluid is governed by the Navier-Stokes equation,

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} &= -\nabla p + \nu \Delta \mathbf{u} \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned} \tag{1.3}$$

where $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ is the Eulerian velocity field, $p = p(\mathbf{x}, t)$ is the pressure, and ν is the kinematic viscosity. In a bounded domain Ω enclosed by walls, the impermeability of the walls and the no-slip condition imply,

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{U}(\mathbf{x}, t) \tag{1.4}$$

where $\mathbf{U}$ is the velocity of the wall.

In terms of the vorticity field $\omega = (\nabla \times \mathbf{u})_z = \partial_x v - \partial_y u$, the flow equation (1.3) becomes

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla)\omega = \nu \Delta \omega \tag{1.5}$$

where $\mathbf{u}$ is obtained from ω through the div-curl relations

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0 \\ (\nabla \times \mathbf{u}) \cdot \hat{z} &= \omega. \end{aligned} \tag{1.6}$$

The divergence condition implies that the flow is derivable from a scalar stream function, $\psi(\mathbf{x}, t)$,

$$\mathbf{u} = \nabla^{\perp} \psi = (-\frac{\partial \psi}{\partial y}, \frac{\partial \psi}{\partial x}) \tag{1.7}$$

which, substituted into the curl condition, yields the Poisson equation

$$(\nabla \times \mathbf{u}) \cdot \hat{z} = \Delta\psi = \omega. \tag{1.8}$$

Finally, the boundary conditions (1.4) translate into boundary conditions for the stream function

$$\nabla^{\perp}\psi = \mathbf{U}, \qquad (x, y) \in \partial\Omega. \tag{1.9}$$

The set of equations (1.5), (1.7), and (1.8), together with the boundary conditions (1.9), is known as the vorticity-stream formulation of the Navier-Stokes equations.

The classical difficulty with the vorticity-stream function formulation is the improper partition of boundary conditions. The presence of a dissipative term in (1.5) requires the specification of boundary conditions for the vorticity, but these are not prescribed explicitly. Vorticity boundary conditions are extremely important from a physical point of view as they represent the mechanism of vorticity generation at the boundary. On the other hand, the Poisson equation (1.8) is overdetermined by both Neaumann and Dirichlet boundary conditions (1.9).

The difficulity is immediately removed if the vorticity equation (1.5) is interpreted instead as an equation for the stream function

$$\frac{\partial}{\partial t}\Delta\psi + [(\nabla^{\perp}\psi) \cdot \nabla]\Delta\psi = \nu\Delta^2\psi. \tag{1.10}$$

This equation contains a biharmonic operator so that the boundary conditions (1.9) are the natural ones with no over- or underdetermination. For more details on fluid mechanics applications see Kupferman [22] and Liu [14].

In linear elasticity[42] $u(x, y)$ can represent the airy stress function or as in theory of thin plates, the vertical displacement due to an external force. In the latter case, equation (1.1) represents a "clamped plate" where f is the external load. Another closely related case is that of a "support plate" where the boundary conditions in (1.1) are replaced by

$$\begin{cases} u(x, y) = g(x, y) & (x, y) \in \partial\Omega \\ \sigma\Delta u(x, y) + (1 - \sigma)u_{nn}(x, y) = h(x, y) & (x, y) \in \partial\Omega \end{cases} \tag{1.11}$$

where u_{nn} is the second normal derivative and σ is a material constant called Poisson's ration.

When Ω is the a polygon, this is equivalent to a problem (with data depending on σ) of the form:

$$\begin{cases} -\Delta v = f & \text{in } \Omega \\ v = -h & \text{on } \partial\Omega \end{cases}$$
$$\begin{cases} -\Delta u = v & \text{in } \Omega \\ u = g & \text{on } \partial\Omega \end{cases} \tag{1.12}$$

where $v = -\Delta u$ has been introduced. The original fourth order equation has been split into two Poisson problems. There exist many reliable algorithms that can be used to solve

(1.12). It is important to notice that the only difference between (1.1) and (1.12) is that different boundary condition have been specified.

The theory of thin plates allowing large vertical displacements, leads to a coupled pair of nonlinear equations know as von Kármáns equations,

$$\begin{cases} \Delta^2 u = [u,v] + f & \text{in } \Omega \\ u = g_1 & \text{on } \partial\Omega \\ u_n = h_1 & \text{on } \partial\Omega \end{cases} \qquad \begin{cases} \Delta^2 v = -[u,u] & \text{in } \Omega \\ v = g_2 & \text{on } \partial\Omega \\ v_n = h_2 & \text{on } \partial\Omega \end{cases} \tag{1.13}$$

where

$$[u,v] = \frac{\partial^2 u}{\partial x^2}\frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 u}{\partial y^2}\frac{\partial^2 v}{\partial x^2} - 2\frac{\partial^2 u}{\partial x \partial y}\frac{\partial^2 v}{\partial x \partial y}. \tag{1.14}$$

Here u represents the vertical displacement of the plate, v is the airy stress function and f is the external force on the plate. An efficient method for solving linear problems involving the biharmonic operator (with the appropriate boundary conditions) can be very valuable in iterative methods for solving more difficult problems of this type.

References describing equations involving the biharmonic operator in elasticity include [17].

1.2 Overview of numerical methods for the biharmonic equations

Various approaches for the numerical solution of (1.1) have been considered in the literature. A popular technique is the coupled equation approach(Smith [38] [39], Ehrlich [15]). In this approach, the problem of solving the biharmonic equation is reduced to perhaps an easier one of solving Poisson equations. The obvious formulation of (1.1) as a coupled system of Poisson equations is as follows:

$$\begin{cases} \Delta u(x,y) &= v(x,y) & (x,y) \in \Omega \\ u(x,y) &= g_1(x,y) & (x,y) \in \partial\Omega \end{cases} \tag{1.15}$$

$$\begin{cases} \Delta v(x,y) &= f(x,y) & (x,y) \in \Omega \\ v(x,y) &= g(x,y) & (x,y) \in \partial\Omega \end{cases} \tag{1.16}$$

each of which may be discretized using the standard 5-point approximations and solved using fast Poisson solvers. The difficulty with this approach is that the boundary conditions for (1.15) is overprescribed on $\partial\Omega$, while (1.16) had no boundary condition at all. Hence given a $g(x,y)$ defined in Ω, (1.15) cannot in general be solved. There are infinitely many solutions to (1.16). While this formulation is equivalent to (1.1), it definitely does not lead to a well-defined iterative scheme for the solution of (1.1).

Under the assumption that $u(x, y)$ is a classical solution of the biharmonic problems (i.e., $u \in C^4(\Omega) \bigcap C^1(\bar{\Omega})$ and u had piecewise continuous second derivatives on $\partial\Omega$), McLaurin [31] introduced another equivalent formulation of the coupled system

$$\begin{cases} \Delta u(x,y) &= v(x,y) \qquad (x,y) \in \Omega \\ u(x,y) &= g_1(x,y) \qquad (x,y) \in \partial\Omega \end{cases} \tag{1.17}$$

$$\begin{cases} \Delta v(x,y) = f(x,y) & (x,y) \in \Omega \\ v(x,y) = \Delta u(x,y) - c[\frac{\partial u}{\partial n} - g_2(x,y)] & (x,y) \in \partial\Omega \end{cases} \tag{1.18}$$

where c is an arbitrary nonzero constant and it is called the coupling constant. It satisfies $0 < c < 2\nu_1$, where ν_1 is the smallest eigenvalue of the so-called Dirichlet eigenvalue problem. When Ω is a rectangular domain, this formulation can lead to an iteration scheme which converges for all sufficiently small values of c.

Another approach for solving the biharmonic equations is to discretize equation (1.1) on a uniform grid directly. The classical 13-point stencil for the biharmonic operator is most easily derived by applying the standard 5-point Laplacian operator twice:

$$\begin{aligned} \Delta_{13}^2 u_{i,j} =& L_5(L_5 u_{i,j}) \\ =& \frac{2}{h^4}(20u_{i,j} - 8(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} \\ &+ u_{i,j-1}) + 2(u_{i+1,j+1} + u_{i-1,j+1} + u_{i-1,j-1} + u_{i+1,j-1}) \\ &+ (u_{i+2,j} + u_{i-2,j} + u_{i,j+2} + u_{i,j-2})). \end{aligned} \tag{1.19}$$

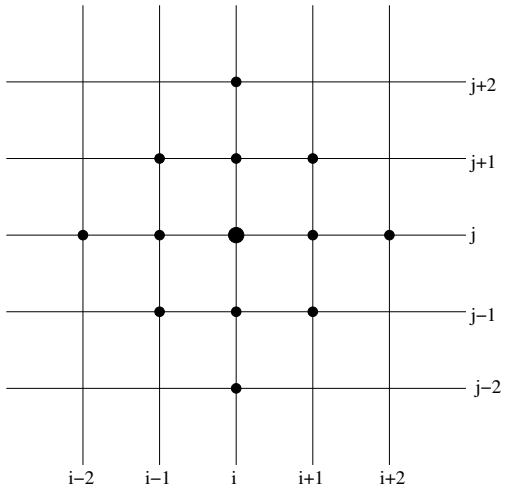


Figure 1.2: A classical 13 point finite difference stencil.

This approximation connects the values of $u_{i,j}$ in terms of 12 neighboring values of u and it is of truncation error of order h^2. Define irregular points as all the grid points that

are adjacent to the boundary. Unlike the discrete Laplace operator which can be applied to all interior grid points, this operator is not well defined on the irregular points. The above finite difference approximation needs to be modified at those irregular grid points. One popular choices is the quadratic extrapolation where the normal derivative boundary condition at the irregular points are used to extrapolated the "missing" (exterior) point in the 13 point stencil. This results in a stencil of the form:

$$\begin{aligned}\Delta^2 u_{i,j} =& \frac{2}{h^4}(21u_{i,j} - 8(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} \\ &+ u_{i,j-1}) + 2(u_{i+1,j+1} + u_{i-1,j+1} + u_{i-1,j-1} + u_{i+1,j-1}) \\ &+ (u_{i+2,j} + u_{i,j+2} + u_{i,j-2})) + 2hu_n(i-1,j)\end{aligned} \tag{1.20}$$

when the irregular point $u_{i,j}$ is adjacent to the left boundary.

Glowinski and Pironneau [16] made the observation that the 13-point finite difference scheme combined with quadratic extrapolation near the boundary is equivalent to solving the biarmonic equation using a mixed finite element method and piecewise linear elements in the triangulation shown.

Many similar modifications are discussed in [18]. Proper treatment of the points near the boundary remains a serious problem with these schemes since inaccuarate boundary approximations must be used in order not to destroy the matrix structure.

There are further difficulties with the solution of the linear systems obtained through the 13-point discretization of the biharmonic equation. The direct solvers for solving the resulting systems of linear equations can only be used for moderate values of grid width h, and the conventional iterative methods such as Jacobi or GaussSeidel either converge very slowly or diverge [19]. Bjørstad [6] introduced a new method for solving the linear system. For an $n \times n$ mesh the algorithm requires only $O(n^2)$ arithmetic operations.

There are many alternative finite difference approximations that can be derived for the biharmonic operator. Certain second- and fourth-order finite difference approximations for the biharmonic equation (1.1) on a 9-point compact cell are given in [40]. This approach involves discretizing the biharmonic equation (1.1) using not only the grid values of the unknown solution u but also the values of the gradients u_x and u_y at selected grid points.

For example, a finite difference approximation of order h^2 is obtained by choosing the values of u at eight neighboring points and the values of u_x, u_y at two neighboring points. With these choices, the 9-points stencil for the biharmonic operator is

$$\begin{aligned}\Delta_9^2 u_{i,j} =& \frac{2}{h^4}(28u_{i,j} - 8(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) \\ &+ (u_{i+1,j+1} + u_{i-1,j+1} + u_{i-1,j-1} + u_{i+1,j-1}) \\ &+ 3h(u_{x_{i+1,j}} - u_{x_{i-1,j}} + u_{y_{i,j+1}} - u_{y_{i,j-1}})).\end{aligned} \tag{1.21}$$

Corresponding finite difference approximations for u_x and u_y at (x_i, y_j) are given by

$$\begin{aligned}u_{x_{i,j}} &= \frac{3}{4h}(u_{i+1,j} - u_{i-1,j}) - \frac{1}{4}(u_{x_{i+1,j}} + u_{x_{i-1,j}}) \\ u_{y_{i,j}} &= \frac{3}{4h}(u_{i,j+1} - u_{i,j-1}) - \frac{1}{4}(u_{y_{i,j+1}} + u_{y_{i,j-1}})\end{aligned} \tag{1.22}$$

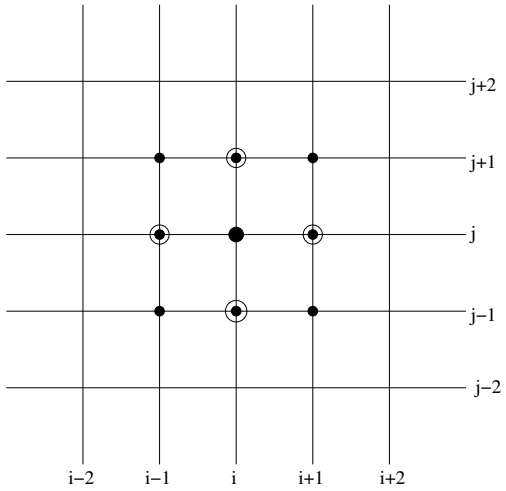


Figure 1.3: A 9-point stencil.

Here (1.22) are fourth-order stencils for the first derivative and (1.21) is a second order scheme. There also exist some fourth-order compact finite difference approximations [40] for the biharmonic equation.

Utilizing the grid values of the gradients is advantageous because the given Dirichlet boundary conditions are exactly satisfied and no approximations need to be carried out at the boundaries. At the same time, the proposed finite difference approximations are derived on a 9-point compact cell and no modifications are needed at grid points near the boundaries. The disadvantage is that the solution and its first derivatives are all carried as unknown at the grid points. This introduces extra computation.

The standard iteration methods suffer from slow convergence when used to solve the system of equations arising from the 9-point discretizations given in [40]. Altas etc. [2] applied the multigrid technique to this problem to alleviate the convergence problem.

Due to the difficulties in handling the finite difference approximation close to a curved boundary, all the methods discussed above are restricted to rectangular domains. More general domains are usually treated within the frame work of the finite element method. There has been little work about solving biharmonic equations on complex domains using the finite difference method based on Cartesian grid. Even for the finite element method, there are few computation results were reported in this category. In this thesis, we will develop a computation method based on finite difference approach for the biharmonic problem on irregular domains by using a decomposing and embedding technique.

1.3 Our strategy

Our iterative method takes various different forms for different problems. But there are some common characteristics which we list below.

1.3.1 The sketch of our method

Our method is based on an approach involves the following aspects:

- Decompose equation (1.1) into two coupled Poisson equation.
- Embed the irregular region into a rectangular domain.
- Use finite difference methods in the Cartesian grids.
- Apply a fast solver to solve the two embedded Poisson system.
- Set the Laplacian Δu on the boundary be an intermediate unknown.
- Use one-side interpolation to compute for Δu.
- Iteratively solve the resulting system of difference equations to get the approximate solution of the biharmonic system.

1.3.2 Decomposing the system

We use finite difference method to solve the first Dirichlet biharmonic problem. As discussed in the previous section, proper treatment of the points near the boundary is a serious problem with finite difference approaches to the biharmonic problem. To get a more accurate approximation to the points near the boundary without changing the structure of the resulting linear system, we apply local coordinate transformation on the boundary points. While these transformations are not preserved under the biharmonic operator, we first decompose equation (1.1) into two tightly connected Poisson equations

$$\begin{cases} \Delta u(x,y) = v(x,y) & (x,y) \in \Omega \\ u(x,y) = g_1(x,y) & (x,y) \in \partial\Omega \end{cases} \\ \begin{cases} \Delta v(x,y) = f(x,y) & (x,y) \in \Omega \\ v(x,y) = g(x,y) & (x,y) \in \partial\Omega \end{cases} \tag{1.23}$$

where $g(x,y)$ is the unknown boundary Laplacian which can only be solved iteratively. It is obvious that the solution of $u(x,y)$ and $v(x,y)$ depends heavily on $g(x,y)$. To get $g(x,y)$ efficiently and accurately turns out to be a crucial step in solving the coupled Poisson system. Our approach is based on a weighed least squares formulation. The same idea can also be, and has been, applied to the case, where we want to approximate certain boundary quantities from a grid function.

Using the coupled equation approach, we need to solve two Poisson equations in (1.23) on the irregular domain Ω at every iterative step. We refer the readers to the references [20], [29], [26] for the theoretical analysis and the details of the implementation.

Our Poisson solver on irregular domains is based on the Immersed Interface Method(IIM) [27]. The main idea is to extend the Poisson equation from Ω to a rectangular domain R.

This procedure allows us to use existing fast Poisson solvers such as FFT on a fixed Cartesian grid.

Fast Poisson solvers are methods for solving the linear systems of equations that arise from discretizing Poisson equation on regions that are rectangular with respect to some coordinates. These methods are efficient and require relatively low storage. For example, by using a Buneman solver one can solve a problem on a square with a n by n grid using only $5n^2 \log_2 n$ operations and $n^2 + 3n$ storage locations.

In order to take advantage of the fast Poisson solver which is designed for solving the discrete systems of Poisson equation on rectangular regions, we first embed the irregular region Ω into a rectangular region $R : [a, b] \times [c, d]$. We then extend the source term of each Poisson equation by zero outside Ω but inside R. We require the normal derivative of the solution to be continuous across the immersed boundary $\partial\Omega$. The solution itself is allowed to have a jump. In the language of potential theory this requirement is equivalent to introducing a double layer source on $\partial\Omega$. For the first Poisson problem in (1.23), this extension leads to the following interface problem:

$$\begin{aligned} \Delta u &= \begin{cases} v(x,y) & (x,y) \in \Omega \\ 0 & (x,y) \in R - \Omega \end{cases} \\ [u] = t, &\qquad [u_n] = 0, \qquad \text{on } \partial\Omega \\ u = 0 &\qquad \text{on } \partial R \end{aligned} \tag{1.24}$$

where [.] denotes the jump across $\partial\Omega$ and t is a function defined on along the boundary. We need to determine the particular t so that the solution u of (1.24) satisfies the Dirichlet boundary condition

$$u^- = g_1 \qquad \text{on } \partial\Omega \tag{1.25}$$

where u^- is the limiting value of the solution on the boundary from within the domain Ω. Note that the solution of the interface problem above is a functional of t. There is a unique solution $u(t)$ which is in piecewise $H^2(R)$ space if v and t are in L^2 space, and the interface $\partial\Omega$ is Lipschitz continuous. Discussion on this topic can be found in [8]. A similar analysis also applies to the solution v of the second Poisson problem in (1.23).

As we will see, the discontinuities in the normal and tangential directions can be expressed in terms of the solution pair u and t of (1.24) through interface analysis. Using these discontinuities and the shape of the boundary curve, we can find all the discontinuities in the coordinate directions and use them to compute an approximation to the discrete Laplacian at mesh points near the irregular boundary. Fast solvers can then be used to compute the Poisson equation on the extended rectangular domain.

The Poisson solver that we outlined above is second-order accurate. Our numeric simulation show that the number of calls to the fast Poisson solver on the rectangular domain is almost independent of the mesh size but depends only on the geometry of the irregular domain.

There are other elliptic solvers for irregular domains using embedding or fictitious domain techniques. The earlier ones include the capacitance matrix method [36], the integral

equation approach [30], and the recent finite volume method [21]. The similarity of all embedding methods is that the standard finite difference/volume schemes are used away from the interface. The difference is the way to deal with the boundary condition. The capacitance matrix method works through manipulating the structure of the coefficient matrix. In the integral equation approach, the integral equation is discretized and the dense system is solved before a fast Poisson solver is called, see [30]. Note that the integral equation is a second kind, and therefor it is well conditioned. In our approach, we use the GMRES iteration to solve a Schur complement system which is equivalent to the discrete integral equation [28]. Theoretically, the GMRES method converges to the solution, or the least squares solution if the system is singular, at most N_1 steps, where N_1 is the number of unknown, which in our case is the number of irregular grid points.

1.3.3 Uniform Cartesian grid

Basically, our scheme is to solve the biharmonic problem (1.1) using finite difference methods on uniform Cartesian grid. After domain embedding, the definitions of two Poisson equations are extended to a rectangular domain with some jumps across the immersed boundary, or the interface. One of the most obvious advantages of using uniform Cartesian grids is that there is almost no cost for grid generation, and the conventional finite difference schemes can be used at most grid points (regular) which are away from the boundary since there are no irregularities there. Only those points near the boundary, which are usually much fewer than the regular grid points, need special attention.

In particular, for these two Poisson problems, we will see that our modified finite difference equation derived from the Immersed Interface Method uses the standard 5-point finite difference operator and only the right hand side of the linear system is modified. This means we can take advantage of many software packages or methods developed for Cartesian grids, for example, fast Poisson solvers [41], Clawpack [24], Amrclawpack [5], the level set method [32], algebraic multigrid solvers [43] and many others.

Certainly there are many other ways to discretize these problems. Using a grid that conforms to the interface is an obvious alternative, for example a structured grid that is deformed in the neighborhood of the interface [7] or an unstructured triangulation. The finite element method on such a grid would be a natural choice for elliptic equations, and can be used very successfully [3]. However, our modified finite difference scheme use the standard five point finite difference operator for the coupled Poisson equations and only the right hand side of the linear system needs to be modified. A uniform grid is preferable in that we can easily apply fast Poisson solvers to the resulting linear system.

1.3.4 Represent the boundary

To solve for u, getting the boundary quantities Δu is an essential step in our method. The process to update Δu in our method shares a similar iteration process to that of immersed interface method. Hence we choose to embed the irregular region into a rectangular domain and then solving the equation on a Cartesian grid in the rectangle. The original boundary

then becomes an interface. The original application of Peskin's immersed boundary method [34] [35] used this approach: the fluid dynamics problem within the heart was extended to a flow problem over a rectangle.

To solve the embedded problem numerically, we need the information about the immersed boundary such as the position, tangential and normal directions, and sometimes curvature as well. One common approach to express the boundary is analytic expression. If the interface is of certain regular shape, we may have an analytic expression for the interface. However, it can still be difficult to calculate other information needed such as the first derivative to determine the curvature, etc., if the analytic expression is too complicated. Then a discrete method to calculate those quantities to a certain accuracy is needed.

One discrete approach to express the boundary is to use discrete difference formulas such as the central difference to get the required derivatives. This approach had been widely used in implementing the immersed boundary method for many problems. Another different approach is used in [25] [27] [26], etc. The idea is to use piecewise interpolation, mostly cubic splines, to get an analytic expression of the boundary, then calculate all the information about the boundary from the analytic expression of the interpolated boundary.

Depending on the knowledge of the physical problem, we can choose a suitable method to express the interface. We use the level set approach in this thesis. This approach was introduced by Osher and Sethian in [32]. In this approach, the boundary is modelled as the zero level set of a smooth function ϕ defined on the whole physical domain.

This approach does not rely on a discrete parameterizations of the express and can be used for complicated moving boundary in two and three dimensions. It can handle cusps and spikes and situations in which the boundaries break or merge.

1.4 Outline of the thesis

In chapter 2, we will give the detailed description on how to apply the immersed interface method to solve the biharmonic equation on a irregular domain. First the biharmonic equation is decomposed into two coupled Poisson equations. The irregular domain is then embedded into a rectangular region. The definition of the solution u and v of two Poisson equations are extended to the rest of the rectangular region. This extension introduces discontinuity across the immersed boundary, and the immersed boundary can be deemed as a interface in the rectangle.

The solution of the resulting coupled system depends on the Laplacian Δu on the boundary. The Laplacian on the boundary is unknown and needs to be solved iteratively. We use a weighted least square formulation to interpolate the Laplacian on the boundary by the computed solution u . The accuracy of the interpolation scheme turns out to be a crucial step in solving the biharmonic problem in our algorithm. Three different kinds of interpolation methods are tested and the one-sided walk-through interpolation are selected in our final algorithm. It is a second order accurate method and the cost for computing Δu is less than other interpolation schemes.

The final linear system need to be solved involves both the solution u and the Laplacian

on the boundary. In order to take advantage of fast Poisson solver, we use GMRES method solve a Schur complement system for Δu iteratively, and use the immersed interface method to solve the coupled Poisson problem at every iterating step. Putting all these works together, we get a second order fast algorithm for solving biharmonic equations on irregular domain.

As part of the thesis, we develop some general application programs for two dimensional biharmonic problem based on the fast algorithms proposed in this thesis. In this fast algorithm, the coefficient matrices for the Schur complement and the Poisson equations are never formed in practice. This is a big advantage for large scale problems since most of the coefficient matrices are sparse matrix. Numerical analysis confirmed our method is of second order accuracy. Another property of the algorithm needs to be pointed out is that the number of iterations of our method seems to be independent of the mesh size.

Some applications of the proposed fast algorithm are investigated in Chapter 4.

Chapter 2

The Immersed Interface method for biharmonic equations

The Dirichlet problem for the biharmonic operator we intend to solve is the following:

$$\begin{cases} \Delta^2 u(x,y) = f(x,y) & (x,y) \in \Omega \\ u(x,y) = g_1(x,y) & (x,y) \in \partial\Omega \\ u_n(x,y) = g_2(x,y) & (x,y) \in \partial\Omega \end{cases} \tag{2.1}$$

here Ω is an irregular region in R^2.

Under the assumption that u is a classical solution of the above problem (i.e. $u \in C^4(\Omega) \cap C^1(\overline{\Omega})$ and u has piecewise continuous second derivatives on $\partial\Omega$), a new and efficient solution technique will be described for the case when the above system is decomposed in to two Poisson problems. It will be shown that by using this method, the solution of the discrete problem on an $N = n \times n$ uniform grid can be computed in $O(N \log N)$ arithmetic operations.

Some of the important characteristics of the method are:

- The Biharmonic equation is decomposed into a system of two coupled Poisson equations.
- The irregular region is embedded into a rectangular domain.
- Finite difference method is used to derive the discrete form of the coupled Poisson problem, and the unknown of the resulting finite difference equation involves both the solution u and the Laplacian Δu on the boundary.
- IIM is used to solve the Poisson problems in irregular domain.
- A one-side interpolation scheme is used to get accurate approximation to the Laplacian on the boundary.
- We use a GMRES process to iteratively update Δu to get the approximate solution of the differential equation.

We will discuss the details of each step in the following sections.

2.1 Coupled System

Consider the solution $u_g(x, y)$ of the following problem as a functional of $g(x, y) \in \partial\Omega$:

$$\begin{cases} \Delta u(x,y) = v(x,y) & (x,y) \in \Omega \\ u(x,y) = g_1(x,y) & (x,y) \in \partial\Omega \end{cases} \qquad \begin{cases} \Delta v(x,y) = f(x,y) & (x,y) \in \Omega \\ v(x,y) = g(x,y) & (x,y) \in \partial\Omega. \end{cases} \tag{2.2}$$

Let the solution of problem (2.1) be $u^*(x, y)$, and define

$$g^*(x,y) = \Delta u^*(x,y) \qquad (x,y) \in \partial\Omega \tag{2.3}$$

along the boundary. Then $u^*(x, y)$ satisfies the biharmonic equation (2.1) and (2.2) with $g(x, y) \equiv g^*(x, y)$. In other words, $u_{g^*}(x, y) \equiv u^*(x, y)$ and:

$$\frac{u^*(x,y)}{\partial n} = g_2(x,y) \qquad (x,y) \in \partial\Omega \tag{2.4}$$

is satisfied. Therefore, solving problem equation (2.1)is equivalent to finding the corresponding g^* and then $u_{g^*}(x, y)$ in problem (2.2). Notice that g^* is only defined along the boundary, so it is one dimensional lower than $u(x, y)$.

In our approach, we use a weighted least square formulation. This method is inspired by Peskin's idea [33] in interpolating a velocity field to get the velocity $u(x, y)$ of the immersed boundary using a discrete delta function. His approach is very robust and only a few neighboring grid points near the boundary are involved. However it is only first order accurate and may smear out the solution near the boundary. We use a refined interpolation formula to get g^* in our approach. Also we have a different way to choose the grid points that are involved in interpolating. Besides the advantages of Peskin's method, our interpolation formula has the flexibility to choose the coefficients in discrete form and it can achieve second order accuracy.

To get the numerical solution of $u_{g^*}(x, y)$ in problem (2.2), we use the Immersed Interface Method developed by Li[26]. The IIM is originally designed to solve interface problems. The idea is to use a five-point stencil to discretize the left hand side of a Poisson equation, and modify the right hand side by adding an additional correction term to get a second order scheme. If the Poisson equation has a discontinuous coefficient, a six-point stencil is used instead of the standard five-point stencil. Besides interface problems, the immersed interface method can also handle problems on irregular regions by imbedding the region into a rectangular domain and then solving the equation on a Cartesian grid in the rectangle.

Since the coefficients in problem (2.2) are all constants, we simply use the five-point stencil. Then we can take advantage of fast Poisson solver for the two discrete Poisson system. Under the framework of IIM, the cost in solving the coupled problem (2.2) is just a little more than that in solving two regular Poisson equations on the rectangle with a smooth solution. At the same time, our iterative process in solving for g^* share a computation

structure similar to that of IIM. As we will see in the following chapter, this makes the implementation of our method much easier.

There also exist many other numerical schemes that can be used to solve this kind of problem in a efficient way both for special geometries and in more general domains. More details are discussed in [41] and [36].

To apply the Immersed Interface Method to the problem pair (2.2), we need embed the region into a rectangular domain and then solve the equation in the whole rectangle. The original boundary then becomes an interface. The original application of the immersed boundary method [33] used this approach: the fluid dynamics problem within the heart was extended to a flow problem over a rectangle.

Based on its idea, we use the embedding techniques introduced in [27] to solve Poisson equations on complicated regions with Dirichlet boundary conditions.

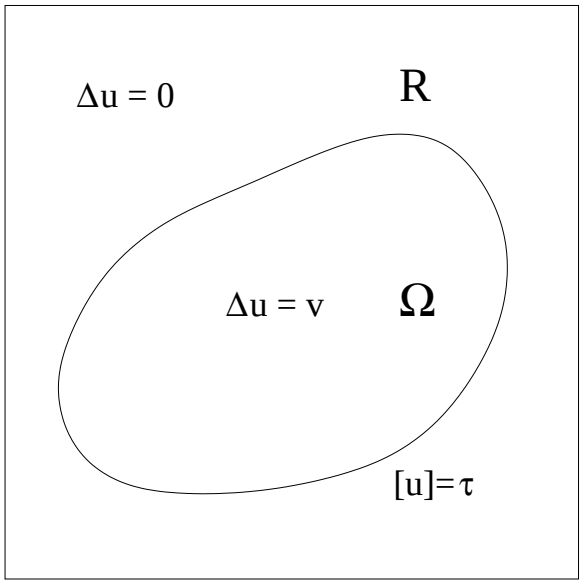


Figure 2.1: A diagram of extending $u(x, y)$ to $[a, b] \times [c, d]$

First, we extend the definition of the coupled problem (2.2) to a rectangular region $R : [a, b] \times [c, d]$. For the first Poisson problem, it can be extended to R by:

$$\begin{aligned} \Delta u(x,y) &= \begin{cases} v(x,y) & (x,y) \in \Omega \\ 0 & (x,y) \in R - \Omega. \end{cases} \\ [u](s) &= \tau(s) \qquad (x(s), y(s)) \in \partial\Omega \\ u(x,y) &= 0 \qquad (x,y) \in \partial R \end{aligned} \tag{2.5}$$

where $(x(s), y(s))$ is the arc-length parameterization of $\partial\Omega$ and $\tau(s)$ is a unknown jump

function defined on $\partial\Omega$ satisfies:

$$u_\tau(x,y) = u(x,y) \qquad (x,y) \in \partial\Omega. \tag{2.6}$$

For the second Poisson problem, the extension to R is:

$$\begin{aligned} \Delta v(x,y) &= \begin{cases} f(x,y) & (x,y) \in \Omega \\ 0 & (x,y) \in R - \Omega \end{cases} \\ [v](s) &= \tilde{\tau}(s) \qquad (x(s),y(s)) \in \partial\Omega \\ v(x,y) &= 0 \qquad (x,y) \in \partial R \end{aligned} \tag{2.7}$$

where $\tilde{\tau}(s)$is another unknown jump function defined on $\partial\Omega$ satisfies:

$$v_{\tilde{\tau}}(x,y) = \Delta u(x,y) \qquad (x,y) \in \partial\Omega. \tag{2.8}$$

Details on how to implement the IIM methods to solve problem (2.5) and (2.7) will be discussed in the following sections.

2.2 The discrete approximation

2.2.1 Cartesian grid and level set function

To apply the IIM method to the Dirichlet boundary value problem, the irregular region is first embedded into a rectangular domain $R : [a,b] \times [c,d]$. The original boundary then becomes an interface. We solve the coupled problem (2.5) and (2.7) on a uniform Cartesian grid on the whole rectangular region with:

$$\begin{aligned} x_i &= a + ih \qquad 0 \le i \le m \\ y_j &= c + jh \qquad 0 \le j \le n. \end{aligned} \tag{2.9}$$

For simplicity, we assume that $h = (b-a)/m = (d-c)/n$. To solve the problem numerically, we need the information about the boundary such as the position, tangential and normal directions, and sometimes curvatures as well. We use a level set formulation to represent the immersed boundary to facilitate the computation of the boundary quantities. In this approach, the boundary is defined as the zero set of a smooth function ϕ defined on the entire rectangular region

$$\phi(x,y) \begin{cases} > 0, & \text{if } (x,y) \notin \Omega \\ = 0, & \text{if } (x,y) \in \partial\Omega \\ < 0, & \text{if } (x,y) \in \Omega. \end{cases} \tag{2.10}$$

Figure (2.2) gives an example of the uniform grid and the immersed boundary.

The level set method has many advantages. An obvious one to use the level set approach is: it is very easy to check whether a grid point is regular or irregular. The irregular points

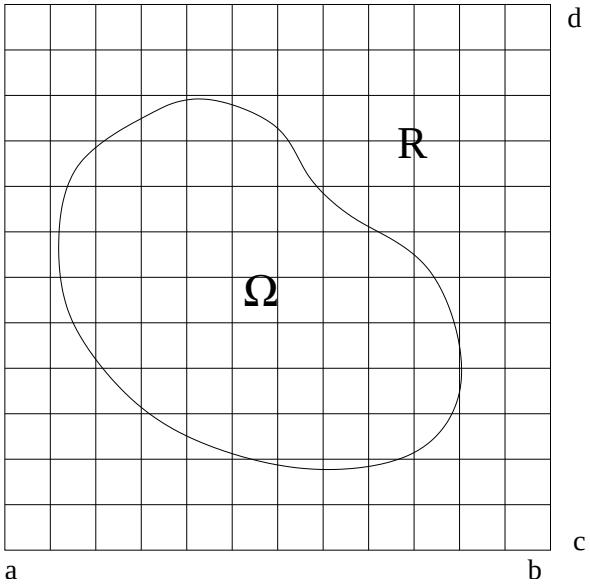


Figure 2.2: An uniform cartesian grid and the immersed boundary

are those grid points where the boundary cuts through their standard five-point stencil. They are identified by

$$\phi_{ij}^{\max}\phi_{ij}^{\min} \leq 0 \tag{2.11}$$

where

$$\begin{aligned} \phi_{ij}^{\max} &= \max\{\phi_{i-1,j}, \phi_{i+1,j}, \phi_{i,j-1}, \phi_{i,j+1}, \phi_{i,j}\} \\ \phi_{ij}^{\min} &= \min\{\phi_{i-1,j}, \phi_{i+1,j}, \phi_{i,j-1}, \phi_{i,j+1}, \phi_{i,j}\}. \end{aligned} \tag{2.12}$$

All other grid points in the rectangle are marked as regular points.

Another advantage to use the level set approach is: much of the geometric information of the immersed boundary is completely determined by the level set function. As we will see later in the chapters, at each irregular grid point, we need find out the its projection on the boundary as well as the normal direction and the curvature at the projection point. If the expression of the level set function is explicitly given, the accurate value at any point on the boundary can be obtained analytically. However, this is not the case in real. Using level set approach, all these boundary quantities can be easily computed by interpolating the grid points function.

2.2.2 The finite difference scheme and the treatment of boundary

For problem (2.5), we want to develop a finite difference equation of the form

$$\sum_k \gamma_k u_{i+i_k,j+j_k} = v_{ij} + C_{ij} \tag{2.13}$$

for use at the point (x_i, y_j). The sum over k involves 5 points neighboring (x_i, y_j). So each i_k, j_k will take values in the set $\{-1, 0, 1\}$. The coefficients γ_k and indices i_k, j_k will depend on (i, j), so these should really be labelled γ_{ijk}, etc., but for simplicity of notation we will concentrate on a single point (i, j) and drop these indices.

We say (i, j) is a *regular point* if the interface does not come between any points in the standard 5-point stencil centered at (i, j). At these points we obtained an $O(h^2)$ truncation error using the standard 5-point formula

$$\frac{1}{h^2}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}) = v_{ij} \tag{2.14}$$

with

$$C_{ij} = 0. \tag{2.15}$$

We wish to determine formula of the form (2.13) for the irregular points also. Since these points are adjacent to the curve Γ, and form a lower dimensional set, it turns out to be sufficient to require an $O(h)$ truncation error at these points. We expand all $u_{i+i_k,j+j_k}$ about some point (x_i^*, y_j^*) on the interface Γ. We have flexibility in choosing (x_i^*, y_j^*). In our case, We take the point closest to (x_i^*, y_j^*) as illustrated in Figure (2.2). To compute the local truncation error at the point (x_i, y_j), we expand each $u_{i+i_k,j+j_k}$ in Taylor series about (x_i^*, y_j^*), being careful to use the limiting values of derivatives of u from the correct side of the interface. Since we expect the γ_k coefficients to be $O(h^2)$ we must expand out through $O(h^3)$ in order to ensure an $O(h)$ truncation error. We use the superscripts $-$ or $+$ to denote the limiting values of a function from one side or the other.

As an example, in the configuration shown in Figure (2.3), we would expand

$$\begin{aligned} u(x_i, y_j) =& u^- + u_x^-(x_i - x_i^*) + u_y^-(y_j - y_j^*) + \frac{1}{2}u_{xx}^-(x_i - x_i^*)^2 \\ &+ \frac{1}{2}u_{yy}^-(y_j - y_j^*)^2 + u_{xy}^-(x_i - x_i^*)(y_j - y_j^*) + O(h^3) \end{aligned} \tag{2.16}$$

and

$$\begin{aligned} u(x_{i+1}, y_j) =& u^+ + u_x^+(x_{i+1} - x_i^*) + u_y^+(y_j - y_j^*) + \frac{1}{2}u_{xx}^+(x_{i+1} - x_i^*)^2 \\ &+ \frac{1}{2}u_{yy}^+(y_j - y_j^*)^2 + u_{xy}^+(x_{i+1} - x_i^*)(y_j - y_j^*) + O(h^3). \end{aligned} \tag{2.17}$$

If we do this expansion at each point used in the difference equation (2.13) then the local truncation error T_{ij} can be expressed as a linear combination of the values $u^\pm$, $u_x^\pm$, $u_y^\pm$, $u_{xx}^\pm$, $u_{xy}^\pm$, $u_{yy}^\pm$. We wish to eliminate all values on one side of the interface, say the values u^+, u_x^+, u_y^+, u_{xx}^+, u_{xy}^+, u_{yy}^+, in terms of the values on the other side, u^-, u_x^-, u_y^-, u_{xx}^-, u_{xy}^-, u_{yy}^-.

As detailed below, differentiating the jump conditions across Γ and manipulating the results allow us to perform the desired elimination. In order to do the differentiation, it turns out to be very convenient to first perform a local coordinate transformation into direction ξ, normal to Γ, and ζ, tangential to Γ. Under the local coordinate system, the

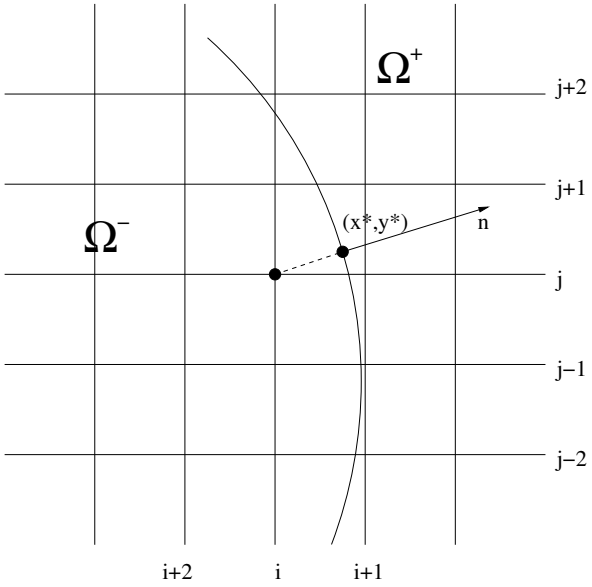


Figure 2.3: An irregular grid point and its projection on the boundary.

local truncation error T_{ij} can be expressed as a linear combination of the values u^-, u^-_x, u^-_y, u^-_{xx}, u^-_{xy}, u^-_{yy}. We then must require that the coefficient of each of these terms vanished in order to achieve an $O(h)$ truncation error. This gives a linear system of six equations to determine the coefficients γ_k in (2.13).

In order to perform the the Taylor expansion, we first need to find a point (x_i^*, y_j^*) on the interface for each irregular grid point (x_i, y_j). We usually take this point as the projection of (x_i, y_j) on the interface if the boundary is smooth at this point. Otherwise we can take any smooth point on the interface in the neighborhood of (x_i, y_j). In some context it may be more convenient to choose a nearby point that lies on a coordinate line between (x_i, y_j) and one of its neighbors.

After choosing (x_i^*, y_j^*) we are ready to apply a local coordinate transformation near this grid point. Let θ be the angle between the x-axis and the normal direction, pointing in the outside direction. The transformation is as follows:

$$\begin{aligned} \xi =& (x - x_i^*)\cos\theta + (y - y_j^*)\sin\theta \\ \eta =& -(x - x_i^*)\sin\theta + (y - y_j^*)\cos\theta. \end{aligned} \tag{2.18}$$

Notice that under this local coordinate transformation both problem (2.5) and (2.7) remain unchanged. In fact, taking an arbitrary function $\omega(x, y)$, under the transformation (2.18), we have

$$\begin{aligned} \omega_x =& \bar{\omega}_\xi \cos\theta + \bar{\omega}_\eta \sin\theta \\ \omega_y =& -\bar{\omega}_\xi \sin\theta + \bar{\omega}_\eta \cos\theta \end{aligned} \tag{2.19}$$

where $\bar{\omega}(\xi,\eta) = \omega(x,y)$ and so forth, so we have

$$\begin{aligned}\omega_{xx} + \omega_{yy} =& \bar{\omega}_{\xi\xi}\cos^2\theta + \bar{\omega}_{\xi\eta}\cos\theta\sin\theta + \bar{\omega}_{\eta\xi}\sin\theta\cos\theta \\ &+ \bar{\omega}_{\eta\eta}\sin^2\theta + \bar{\omega}_{\xi\xi}\sin^2\theta - \bar{\omega}_{\xi\eta}\sin\theta\cos\theta \\ &- \bar{\omega}_{\eta\xi}\cos\theta\sin\theta + \bar{\omega}_{\eta\eta}\cos^2\theta \\ =& \bar{\omega}_{\xi\xi} + \bar{\omega}_{\eta\eta}.\end{aligned} \tag{2.20}$$

We should have a new notation for $u(x,y)$, $v(x,y)$, $f(x,y)$ in the local coordinates, say, $\bar{u}(\xi,\eta) = u(x,y)$, $\bar{v}(\xi,\eta) = v(x,y)$, and $\bar{f}(\xi,\eta) = f(x,y)$. But for simplicity we drop the bars and use the same notation in the local coordinates as in the old ones. With these local coordinates we are able to derive the interface conditions. If some grid point $u(x_i, y_j)$ happens to fall on the interface, then $u(x_i, y_j)$ is defined as the limiting value of $u(x,y)$ from one side of the interface or the other. The same argument applies to all other functions such as v, f and the derivatives of $u(x,y)$.

For a fixed point (x_i^*, y_j^*) and the $\xi-\eta$ coordinate system based on the directions normal and tangential to Γ at this point using the formulas (2.18), the interface lies roughly in the η-direction in a neighborhood of this point, so we can parameterize Γ locally by $\xi = \chi(\eta)$, $\eta = \eta$. Note that $\chi(0) = 0$ and, provided the boundary is smooth at (x_i^*, y_j^*), $\chi'(0) = 0$ as well.

Now two jump conditions are needed in advance to make the problem well-posed. Since jump in the solution to the two coupled Poisson problems were introduced by extending the definition of the solution into the whole rectangular domain, we have the flexibility to choose the jump conditions. In this thesis, we assume under the local coordinate system they are defined by

$$[u] = \omega(\eta) \tag{2.21}$$

and

$$[u_n] = 0 \tag{2.22}$$

where $\omega(\eta)$ is arbitrary (smooth) function that is used to impose quite general jump condition across Γ. Differentiating (2.21) with respect to η along the interface we get

$$[u_\xi]\chi' + [u_\eta] = \omega'(\eta). \tag{2.23}$$

Differentiating this again with respect to η we obtain

$$[u_\xi]\chi'' + \chi'\frac{d}{d\eta}[u_\xi] + [u_{\xi\eta}]\chi' + [u_{\eta\eta}] = \omega''(\eta). \tag{2.24}$$

Notice that in the local coordinates, (2.22) can be written as

$$u_\xi^+ - u_\eta^+\chi' = u_\xi^- - u_\eta^-\chi'. \tag{2.25}$$

Differentiating this with respect to η along the interface we have

$$\begin{aligned}&u_{\xi\xi}^+\chi' + u_{\xi\eta}^+ - \frac{d}{d\eta}(u_\eta^+)\chi' - u_\eta^+\chi'' \\ =& u_{\xi\xi}^-\chi' + u_{\xi\eta}^- - \frac{d}{d\eta}(u_\eta^-)\chi' - u_\eta^-\chi''.\end{aligned} \tag{2.26}$$

Also from the PDE we know that

$$u^+_{\xi\xi} = u^-_{\xi\xi} + u^-_{\eta\eta} - u^+_{\eta\eta} + u^-_{\xi} - u^+_{\xi}u^-_{\eta} - u^+_{\eta} + [v]. \tag{2.27}$$

Using these relations, we can express quantities with + superscripts in terms of those with − superscripts. The detailed analysis is similar to the process introduced by Li [27]. To save space here we omit the detailed analysis and simply present the results. Recall that the parameterization $\xi = \chi(\eta)$ is assumed to be smooth with $\chi'(0) = 0$ and that we are considering the jumps across Γ at a fixed point (x_i^*, y_j^*) corresponding to $\xi = \eta = 0$. In the expression below, all functions are evaluated at this point. The jump relations are given by:

$$\begin{aligned}
u^+ =& u^- + \omega \\
u^+_{\xi} =& u^-_{\xi} \\
u^+_{\eta} =& u^-_{\eta} + \omega' \\
u^+_{\xi\xi} =& \chi'' u^+_{\xi} - \chi'' u^-_{\xi} + u^-_{\xi\xi} - \omega'' + [v] \\
u^+_{\eta\eta} =& u^-_{\eta\eta} + \omega'' \\
u^+_{\xi\eta} =& [u_\eta]\chi'' + u^-_{\xi\eta}.
\end{aligned} \tag{2.28}$$

Now we have expressed all the quantities with + superscripts in terms of those with − superscripts for the case $\chi'(0) = 0$. We are now ready to derive the difference schemes at irregular points. Following the analysis by Li, at an arbitrary irregular point (x_i, y_j), the local truncation error T_{ij} of the difference scheme (2.13) can be written as

$$\begin{aligned}
T_{ij} =& \gamma_1 u(\xi_1, \eta_1) + \gamma_u(\xi_2, \eta_2) + \gamma_3 u(\xi_3, \eta_3) + \gamma_4 u(\xi_4, \eta_4) \\
&+ \gamma_5 u(\xi_5, \eta_5) - v_{ij} - C_{ij}
\end{aligned} \tag{2.29}$$

where (ξ_1, η_1) denotes the $\xi - \eta$ coordinates of (x_{i-1}, y_j), (ξ_2, η_2) denotes the coordinates of (x_i, y_j), etc. We now expand all the terms about $(0, 0)$ in the local coordinates from each side of the interface, obtaining

$$u(\xi_k, \eta_k) = u^{\pm} + \xi_k u^{\pm}_{\xi} + \eta_k u^{\pm}_{\eta} + \frac{1}{2}\xi_k^2 u^{\pm}_{\xi\xi} + \xi_k\eta_k u^{\pm}_{\xi\eta} + \frac{1}{2}\eta_k^2 u^{\pm}_{\eta\eta} + O(h^3) \tag{2.30}$$

where the + and − sign is chosen depending on whether (ξ_k, η_k) lies on the + or − side of Γ.

We also use

$$v_{ij} = v^- + O(h) \tag{2.31}$$

where $v^- = v(0, 0)$. At the same time, let K^+ denote the index set

$$K^+ = \{k : (\xi_k, \eta_k) \text{ is on the } + \text{ side of } \Gamma\} \tag{2.32}$$

and define coefficients a_j as

$$\begin{aligned} a_1 &= \sum \gamma_k & a_2 &= \sum \xi_k \gamma_k \\ a_3 &= \sum \eta_k \gamma_k & a_4 &= \frac{1}{2} \sum \xi_k^2 \gamma_k \\ a_5 &= \frac{1}{2} \sum \eta_k^2 \gamma_k & a_6 &= \sum \xi_k \eta_k \gamma_k. \end{aligned} \tag{2.33}$$

Using these expansions in (2.30) and collecting terms gives an expression of the form

$$\begin{aligned} T_{ij} =& a_1 u^- + a_2 u^-_\xi + a_3 u^-_\eta + (a_4 - 1) u^-_{\xi\xi} + (a_5 - 1) u^-_{\eta\eta} \\ &+ a_6 u^-_{\xi\eta} + (\hat{T}_{ij} - C_{ij}) + O(h) \end{aligned} \tag{2.34}$$

where

$$\begin{aligned} \hat{T}_{ij} = & \sum_{k \in K^+} \gamma_k \omega + \Big(\sum_{k \in K^+} \eta_k \gamma_k + \sum_{k \in K^+} \xi_k \eta_k \gamma_k \chi^{''} \Big) \omega' \\ &+ \sum_{k \in K^+} \xi_k^2 \gamma_k ([v] - \omega^{''}). \end{aligned} \tag{2.35}$$

We can ensure that $T_{ij} = O(h)$ by requiring that each coefficient of u^-, u^-_ξ, u^-_η, $u^-_{\xi\xi}$, $u^-_{\eta\eta}$ and $u^-_{\xi\eta}$ vanish, as well as the term $(\hat{T}_{ij} - C_{ij})$. This gives seven equations for the unknowns $\gamma_1, \ldots, \gamma_5$ and C_{ij}. The first six equations give a linear system for the γ_k's. Li [27] has proved that the coefficients for the standard formula (2.14) are the solution to this linear system. Once the the γ_k's are known, we can use level set function to interpolate all the other boundary quantities ξ_k, η_k, $\chi^{''}$, ω' and $\omega^{''}$. Having done all these, we can easily obtain C_{ij} as

$$C_{ij} = \hat{T}_{ij}. \tag{2.36}$$

The same analysis also applies to the problem (2.7). We use a similar difference formulation for (2.7) except a little change in the right hande side.

2.3 The algorithm description

In this section, unless otherwise specified, we use upper case letters to indicate the solution of the discrete problem and lower case letters for the continuous problem.

2.3.1 Matrix representation

Using the modified finite difference formulation derived in previous section, we have got a discrete form for the problem pair (2.5) and (2.7). The discrete form for (2.5) can be written as

$$L_h U_{ij} = V_{ij} + C_{ij} \tag{2.37}$$

where L_h denote the discrete Laplacian operator. Note that if (x_i, y_j) happens to be on the interface, then V_{ij} is defined as the limiting value from the inside of the interface. C_{ij}

is zero except at those irregular grid points where the interface cuts through the 5-point stencil. From (2.36), we can write (2.37) in matrix vector form as

$$AU + \mathcal{B}(T) = V \tag{2.38}$$

where $AU = V$ is the discrete linear system for the Poisson equation when the jump T in solution are all zero. $\mathcal{B}(T)$ is a mapping from $T = [\tau_1, \tau_2, \ldots, \tau_{n_b}]$ to C_{ij}. As we will see in the following sections, in the discrete case, all the derivatives are obtained by differentiating the corresponding zero level set curve which is linear combinations of the values on those control points. Therefore $\mathcal{B}(T)$ is indeed linear function of T and can be written as

$$\mathcal{B}(T) = BT \tag{2.39}$$

where B is a matrix with real entries. Thus (2.38) becomes

$$AU + BT = F \tag{2.40}$$

The solution U of the above equation certainly depends on T and we are interested in finding T^* which satisfies the discrete form of (2.21)

$$U^+(T^*) - U^-(T^*) = T^* \tag{2.41}$$

As shown in [26], U^+ and U^- depend on U, T and G_1 linearly

$$EU + DT = G_1 \tag{2.42}$$

where E and D are some matrices and G_1 is the discrete form of g_1 in (2.5). Combine (2.40) and (2.42) to obtain the linear system of equations for U and G_1:

$$\begin{bmatrix} A & B \\ E & D \end{bmatrix} \begin{bmatrix} U \\ T \end{bmatrix} = \begin{bmatrix} V \\ G_1 \end{bmatrix}. \tag{2.43}$$

Similarly, the matrix vector form for (2.7) can be written as:

$$\begin{bmatrix} A & B \\ E & D \end{bmatrix} \begin{bmatrix} V \\ \hat{T} \end{bmatrix} = \begin{bmatrix} F \\ G \end{bmatrix}. \tag{2.44}$$

Put the linear systems (2.43) and (2.44) together and rearrange the unknowns, we get

$$\begin{bmatrix} A & -I & B & 0 \\ 0 & A & 0 & B \\ E & 0 & D & 0 \\ 0 & E & 0 & D \end{bmatrix} \begin{bmatrix} U \\ V \\ T \\ \hat{T} \end{bmatrix} = \begin{bmatrix} 0 \\ F \\ G_1 \\ G \end{bmatrix} \tag{2.45}$$

where I is an identity matrix. The solution U and V of the linear systems above certainly depends on G and we are interested in finding G^* which satisfies the discrete form of (2.4)

$$U_n(G^*) - G_2 \equiv 0 \tag{2.46}$$

where the components of the vector U_n is a discrete approximation of the normal derivative at boundary control points from inside of the interface.

To get the solution of the biharmonic equation (2.1), we need to find numerical methods to compute g^* with which we can then compute $u_{g^*}(x, y)$ to second order accuracy. We also hope that the total cost in computing g^* and u_{g^*} is less than that in computing u_{g^*} through the original problem directly. The key to success is computing g^* efficiently. Below we start to describe our method to determine g^*. Once g^* is found, we just need two more fast Poisson solvers to get the solution $u^*(x, y)$.

2.3.2 The weighted least squares interpolation

To solve for g^*, our approach is based on a weighted least squares formulation. We start from the continuous situation, the discrete version can be obtained accordingly. Let $u(x, y)$ be a piecewise smooth function. We want to interpolate $u(x_i, y_j)$ to get approximations to the normal derivative $u_n(x, y)$, where (x, y) are only defined on the boundary, to second order. Our idea is inspired by Peskin's method in interpolating a velocity field $u(x, y)$ to get the velocity of the interface using a discrete delta function. The continuous form of his method are the following

$$u(x, y) = \iint_{\Omega} u(s, t)\delta(x - s)\delta_h(y - t)\, dsdt \tag{2.47}$$

The discrete form is

$$u(x, y) = h^2 \sum_{i,j} u_{ij}\delta_h(x - s)\delta_h(y - t) \tag{2.48}$$

where δ_h is a discrete delta function. Some commonly used examples are the hat function

$$\delta_h(x) = \begin{cases} (h - |x|)/h^2 & \text{if } |x| < h \\ 0 & \text{if } |x| \geq h \end{cases} \tag{2.49}$$

and Peskin's delta function

$$\delta_h(x) = \begin{cases} \frac{1}{4h}(1 + \cos(\pi x/2h)) & \text{if } |x| < 2h \\ 0 & \text{if } |x| \geq 2h. \end{cases} \tag{2.50}$$

Notice both functions are continuous. The hat function is not smooth but gives second order accuracy for many one dimensional problems. Peskin's approach is smooth and very robust. However this approach is only first order accurate and may smear out the solution near the boundary.

Our interpolation formula for u_n is written in the form:

$$u_n(\vec{X}) = \sum_{i,j} \gamma_{ij} u_{ij} d_\alpha(|\vec{X} - \vec{x}_{ij}|) \tag{2.51}$$

where $\vec{X} = (X, Y)$ and $d_\alpha(r)$ is a weighted distance function,

$$d_\alpha(r) = \alpha\delta_{\alpha/2}(r) = \begin{cases} \frac{1}{2}(1 + \cos(\pi r/\alpha)) & \text{if } |r| < \alpha \\ 0 & \text{if } |r| \geq \alpha. \end{cases} \tag{2.52}$$

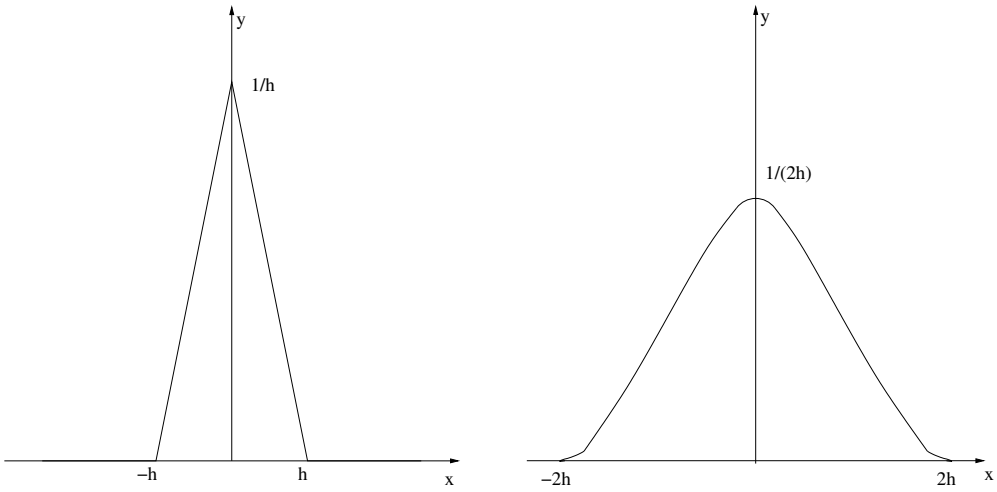


Figure 2.4: Comparison of two typical delta functions. Left: the hat function. Right: Peskin's delta function

Although we are trying to approximate the normal derivative $u_n(x, y)$ here, the same principle also applies to interpolate the function values $u(x, y)$, the Laplacian $\Delta u(x, y)$ on the boundary as well with different choices of γ_{ij}.

In addition to the advantages of Peskin's approach, we also have flexibility in choosing the coefficients γ_{ij} to achieve second order accuracy. The parameter α in (2.51) can be fixed or chosen according to problems.

Note here since we only have information of the inside grid function u_{ij}, selection of the grid points that will be involved in the interpolation must be very careful. The basic idea is to choose those grid points that are not far away from the original boundary point and their projections to the normal direction do not exceed certain value, which in our experiment is $\sqrt{2}h$.

In Peskin's approach, a simple scheme is used to choose grid points to interpolate $u_n(x, y)$. He used a auxiliary circle with (x, y) as the center and radius $4h$. All the grid points in this circle are used to interpolate $v_n(x, y)$. We tried two different ways to choose the grid points. One is to choose only the grid points belonging to the intersection of the auxiliary circle and the irregular region. The other one requires that not only the grid points should be inside the circle and the irregular region, but also they must satisfy certain "walk routine". Our numerical simulation show that the way we choose the grid points for interpolation has great impact on the accuracy of our algorithm. We will discuss this in detail at the end of this section.

Below we will show how to determine the coefficients γ_{ij}. Actually, these coefficients are different from point to point on the boundary. So they should really be labelled as $\gamma_{ij,\vec{X}}$. But for simplicity of notation we will concentrate on a single point $\vec{X} = (X, Y)$ and drop

the subscript $\vec{X}$.

Since it is the normal derivative that we are trying to interpolate, we introduce local coordinates at boundary point (X,Y),

$$\begin{aligned} \xi &= (x - X)\cos\theta + (y - Y)\sin\theta, \\ \eta &= -(x - X)\sin\theta + (y - Y)\cos\theta \end{aligned} \tag{2.53}$$

where θ is the angle between the x-axis and the normal direction at point (X, Y). Under such new coordinates, the interface can be parameterized by $\xi = \chi(\eta), \eta = \eta$. Note that $\chi(0) = 0$ and, provided the boundary is smooth enough at (X, Y), $\chi'(0) = 0$ as well.

Let (ξ_i, η_j) be the $\xi - \eta$ coordinates of (x_i, y_j), we have from Taylor expansion:

$$u(x_i, y_j) = u + u_\xi \xi_i + u_\eta \eta_j + \frac{1}{2} u_{\xi\xi} \xi_i^2 + \frac{1}{2} u_{\eta\eta} \eta_j^2 + u_{\xi\eta} \xi_i \eta_j + O(h^3). \tag{2.54}$$

Put (2.54) into (2.51), after collecting terms we have

$$\begin{aligned} u_n(\vec{X}) &= \sum_{i,j} \gamma_{ij} u_{ij} d_\alpha(|\vec{X} - \vec{x_{ij}}|) \\ &= a_1 u + a_2 u_\xi + a_3 u_\eta + a_4 u_{\xi\xi} + a_5 u_{\eta\eta} + a_6 u_{\xi\eta} + O(h^3 \max|\gamma_{ij}|) \end{aligned} \tag{2.55}$$

where the a_i's are given by

$$\begin{aligned} a_1 &= \sum_{i,j} \gamma_{ij} d_\alpha(|\vec{X} - \vec{x_{ij}}|) \\ a_2 &= \sum_{i,j} \xi_k \gamma_{ij} d_\alpha(|\vec{X} - \vec{x_{ij}}|) \\ a_3 &= \sum_{i,j} \eta_k \gamma_{ij} d_\alpha(|\vec{X} - \vec{x_{ij}}|) \\ a_4 &= \sum_{i,j} \frac{1}{2} \xi_k^2 \gamma_{ij} d_\alpha(|\vec{X} - \vec{x_{ij}}|) \\ a_5 &= \sum_{i,j} \frac{1}{2} \eta_k^2 \gamma_{ij} d_\alpha(|\vec{X} - \vec{x_{ij}}|) \\ a_6 &= \sum_{i,j} \xi_k \eta_k \gamma_{ij} d_\alpha(|\vec{X} - \vec{x_{ij}}|). \end{aligned}$$

From the local coordinate transformation, we have $u_n = u_\xi$, hence the linear system of equations for the coefficients γ_{ij} can be written as:

$$\begin{aligned} a_1 &= 0, \\ a_2 &= 1, \\ a_3 &= 0, \\ a_4 &= 0, \\ a_5 &= 0, \\ a_6 &= 0. \end{aligned} \tag{2.56}$$

Note that we would use the same equation to interpolate a smooth function $u(x_i, y_j)$ to get an approximation u_n at $\vec{X}$ to second order accuracy. If the linear system (2.56) has a solution, then we can obtain a second order approximation to the normal derivative u_n. Therefore we want to choose a selection criteria such that at least six grid points are involved. Usually we have an under-determined linear system which has infinitely many solutions. We should then choose the one γ_{ij}^* with the minimal L_2-norm

$$\sum_{i,j}(\gamma_{ij}^*)^2 = \min \sum_{i,j} \gamma_{ij}^2. \tag{2.57}$$

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

The only trade off of our weighted least square approach is that we have to solve an extra under-determined 6 by p linear system of equation. The larger α is, the more computational cost in solving (2.56). Fortunately, the linear system has full row rank and can be solved by the LR-RU method[7] or other efficient least square solvers. And for fixed boundary problems, these extra computation are only 'over-head', which means it only need to be made only once.

We already know that the linear system of equation for the normal derivative G^* is implicitly defined in the discrete form of

$$U_n(G^*) - G_2 = 0.$$

With the least square interpolation described in this section, the component of the equation above for each point $\vec{X} = (X, Y)$ on the boundary is approximated by

$$\sum_{i,j} \gamma_{ij} u_{ij} d_\alpha(|\vec{X} - \vec{x}_{ij}|) - g_2(\vec{X}) = 0. \tag{2.58}$$

In matrix and vector form, it is

$$\bar{E}U = G_2. \tag{2.59}$$

2.3.3 Selecting grid points for interpolation

We have mentioned that two different schemes for choosing grid points involved in interpolating the normal derivative u_n on the boundary have been tested. Given a boundary point $(\tilde{x}, \tilde{y})$, the original scheme of Peskin's approach is straight forward: All the grid points included in the circle

$$Cir(\tilde{x}, \tilde{y}) = \{(x, y) | (x - \tilde{x})^2 + (y - \tilde{y})^2 \leq \tilde{r}\} \tag{2.60}$$

are used for interpolation. For each grid point, the only information that is needed to tell if this point should be included is its distance to the boundary point $(\tilde{x}, \tilde{y})$. This method is easy to implement. But when applied to the biharmonic equation we are trying to solve, it never reaches a satisfactory accuracy. The reason is probably that after solving two Poisson equations, using computed u_{ij} out side of Ω to interpolate $u_n(\tilde{x}, \tilde{y})$ does not make much sense.

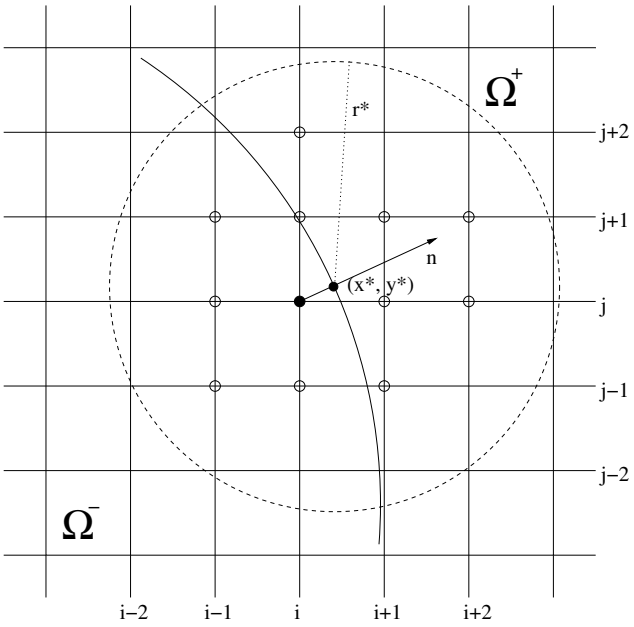


Figure 2.5: Selected grid points for interpolation: Peskin's approach

Our first method intends to eliminate the disadvantage of the above method. For a boundary point $(\tilde{x}, \tilde{y})$, define the index set $K(\tilde{x}, \tilde{y})$

$$K(\tilde{x}, \tilde{y}) = \{(i, j) | (x_i, y_j) \in Cir(\tilde{x}, \tilde{y}) \cap \Omega^-\} \tag{2.61}$$

where Ω^- stands for the in side of the irregular region Ω. Any grid points falling into $K(\tilde{x}, \tilde{y})$ will be chosen for interpolation. In practice, this method works fine when the curvature of the boundary is not too small. But for boundary with big curvature, it is hard to reach second order accuracy for points on the boundary with big curvature.

The idea of the second method is to choose grid points that are "close" to the normal direction and keep the number of grid points on each side of the normal direction equal. We call the process for choosing the interpolation points a"walk through " routine. Suppose $(\tilde{x}, \tilde{y})$ is the projection of a inside grid point (x_i, y_j) on the boundary. Start from (x_i, y_j), our algorithm will walk through grid points in $K(\tilde{x}, \tilde{y})$, find out all points that lies in a sector of $Cir(\tilde{x}, \tilde{y})$. The angle of the sector depends on the curvature at $(\tilde{x}, \tilde{y})$ and the direction

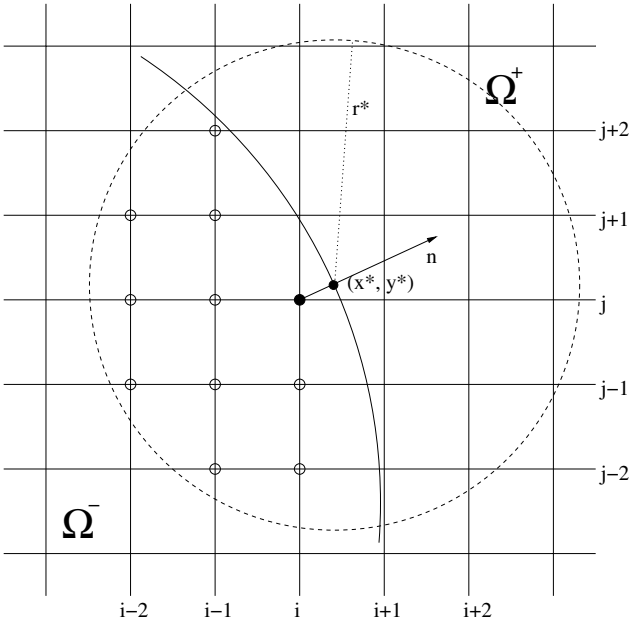


Figure 2.6: Selected grid points for interpolation: the one-side approach

of the normal direction. This idea is intuitive since we are trying to interpolate the normal derivative u_n. Define θ as the angle between the normal direction and the y axis. Listed below are some examples on how to select interpolation grid points for different boundary points.

We have done numerical tests for all three methods. The first method never reaches a satisfactory accuracy in our tests. Table (2.1) shows the numerical results of the last two methods for example 2.1.

Example 2.1 *Consider a biharmonic problem defined on a ellipse* $\frac{x^2}{0.5^2} + \frac{y^2}{0.15^2} = 1$. *The differential equation is:*

$$\Delta^2 u = e^x. \tag{2.62}$$

We use the Dirichlet boundary condition which is determined from the exact solution

$$u(x, y) = x^2 + y^2 + e^x \qquad (x, y) \in \partial\Omega \tag{2.63}$$

where Γ *is the boundary of the circle. The analytic form of the boundary normal derivative is*

$$u_n(x, y) = \frac{0.5^2 0.15^4 (2x^2 + xe^x) + 0.5^4 0.15^2 (2y^2)}{0.15^4 x^2 + 0.5^4 y^2} \qquad (x, y) \in \partial\Omega \tag{2.64}$$

The maximum error over all boundary points

$$\|E_n\|_\infty = \max_k |u_n(\tilde{x}, \tilde{y}) - U_n(\tilde{x}, \tilde{y})|, \tag{2.65}$$

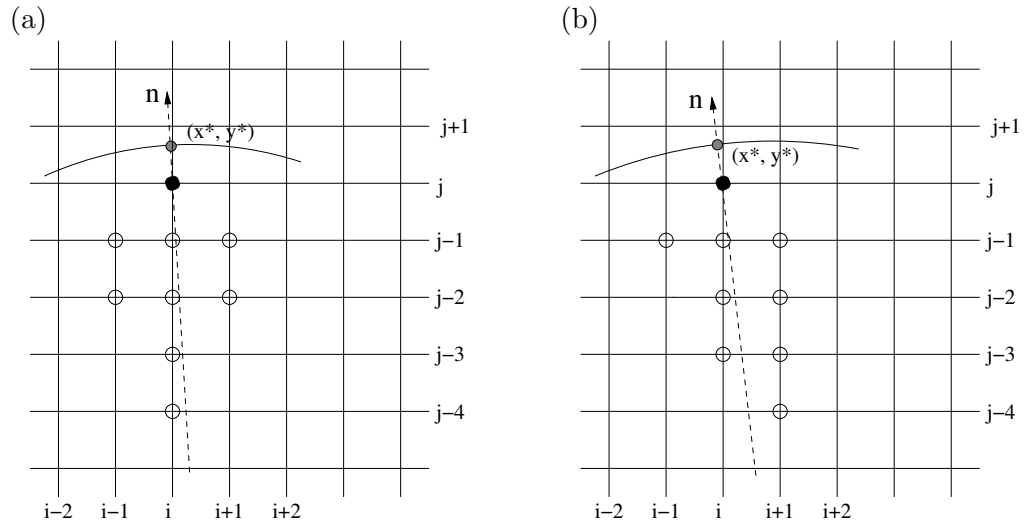


Figure 2.7: Selected grid points for interpolation when (a) $ctg(\theta) \in [0, \frac{1}{10})$ and (b) $ctg(\theta) \in [\frac{1}{10}, \frac{1}{5})$

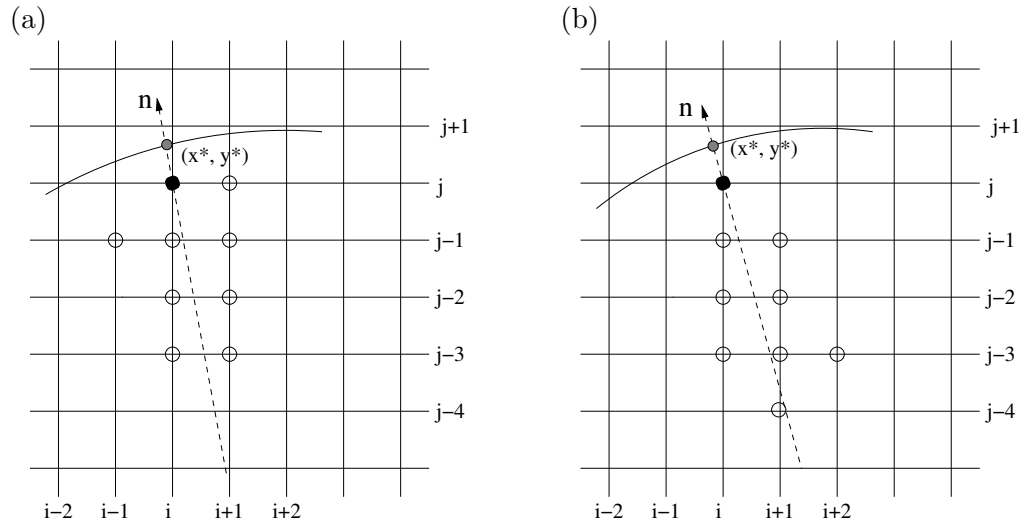


Figure 2.8: Selected grid points for interpolation when (a) $ctg(\theta) \in [\frac{1}{5}, \frac{1}{4}]$ and (b) $ctg(\theta) \in [\frac{1}{4}, \frac{1}{3}]$

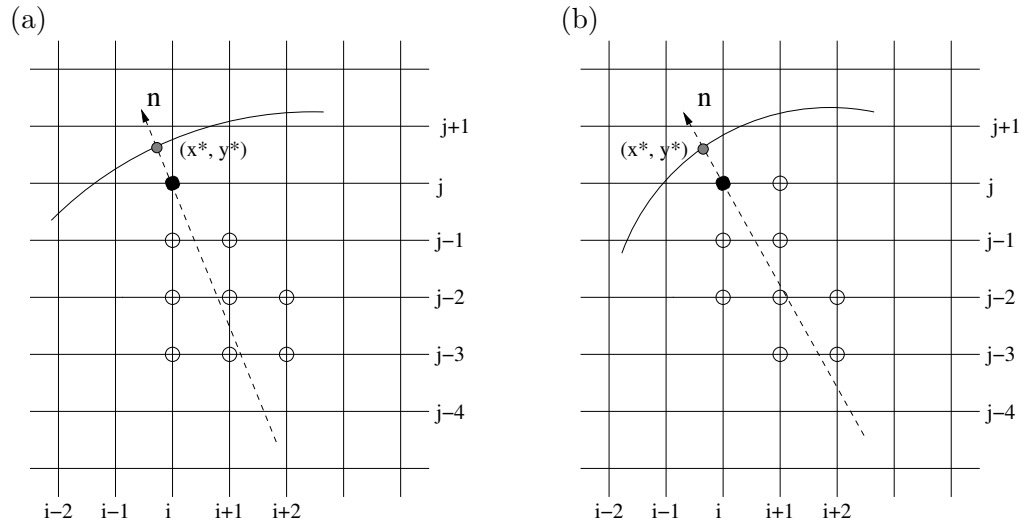


Figure 2.9: Selected grid points for interpolation when (a) $ctg(\theta) \in [\frac{1}{3}, \frac{1}{2}]$ and (b) $ctg(\theta) \in [\frac{1}{2}, \frac{4}{7}]$

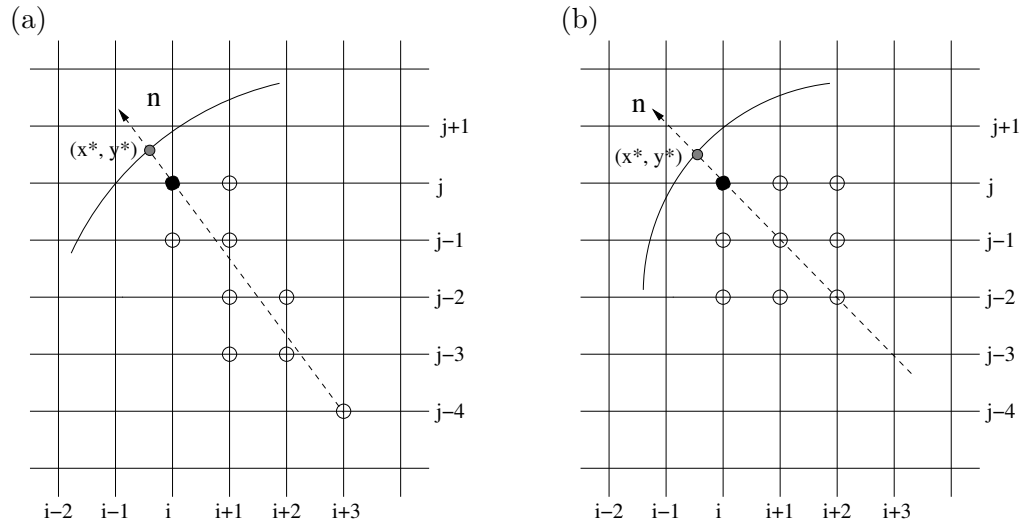


Figure 2.10: Selected grid points for interpolation when (a) $ctg(\theta) \in [\frac{4}{7}, \frac{4}{5}]$ and (b) $ctg(\theta) \in [\frac{4}{5}, 1]$

mesh size	Simple one-side		One-side walk through	
	$\|E_n\|_\infty$	r	$\|E_n\|_\infty$	r
64×64	5.01361×10^{-2}		4.08871×10^{-2}	
128×128	3.68065×10^{-2}	1.36215	1.63559×10^{-2}	2.49984
256×256	1.94529×10^{-2}	1.89208	3.66647×10^{-3}	4.46093
512×512	8.61007×10^{-3}	2.25932	9.50161×10^{-4}	3.85879

Table 2.1: Interpolation errors comparison between simple one-side scheme and one-side walk through scheme

is presented, where $(\tilde{x}, \tilde{y})$ is the kth projection points of the inside irregular grid point and $U_n(\tilde{x}, \tilde{y})$ is the computed approximation. We also display the ratios of successive errors $\|E_n\|_\infty$,

$$r = \frac{\|E_n\|_\infty}{\|E_{2n}\|_\infty} \tag{2.66}$$

A ratio of 2 corresponds to first order accuracy, while a ratio of 4 indicates second order accuracy. The results show that the one-side walk through approach reaches a desired order of accuracy.

Chapter 3

The algorithm description

We have derived two linear systems (2.45) and (2.59) for U, G and an intermediate unknown V. The next question is how to solve these two linear systems efficiently. In order to take advantage of fast Poisson solver, we use GMRES [37] method solve a Schur complement system for G iteratively, and use two IIM solvers computing the solution U and V of (2.45) at every iterate step. Putting all these works together, we get a second order fast algorithm for solving biharmonic equations on irregular domain.

3.1 Schur complement system and the GMRES method

To apply the immersed interface method in solving the linear system (2.45), we need use its two components (2.43) and (2.44). That is, we choose to solve the coupled linear system for U instead of solving (2.45) directly.

First, we need rewrite the coupled system (2.43) and (2.44) to eliminate the unknowns T and $\hat{T}$. Consider the second equation in (2.43), we can write it as:

$$T = D^{-1}(G_1 - EU). \tag{3.1}$$

Then eliminating T from the first equation in (2.43) gives a linear system for U

$$(A - BD^{-1}E)U = V - BD^{-1}G_1. \tag{3.2}$$

Substituting this into the first equation in (2.44), we have

$$(A^2 - ABD^{-1}E)U + B\hat{T} = F - ABD^{-1}G_1. \tag{3.3}$$

Putting (3.2) into the second equation in (2.44), we have

$$(EA - EBD^{-1}E)U + D\hat{T} = G. \tag{3.4}$$

Put (3.3) and (3.4) together and eliminate $\hat{T}$, we have

$$[A^2 - ABD^{-1}E - BD^{-1}EA + (BD^{-1}E)^2]U + D^{-1}G = F - ABD^{-1}G_1. \tag{3.5}$$

The above system can be solved using two IIM solver [26]. To simplify the notations, now let

$$\bar{A} = [A^2 - ABD^{-1}E - BD^{-1}EA + (BD^{-1}E)^2] \tag{3.6}$$

and

$$\bar{F} = F - ABD^{-1}G_1. \tag{3.7}$$

Now (3.5) becomes

$$\bar{A}U + D^{-1}G = \bar{F}. \tag{3.8}$$

Putting (3.8) and (2.59) together, the linear system of equations for U and G can be written as:

$$\begin{bmatrix} \bar{A} & D^{-1} \\ \bar{E} & 0 \end{bmatrix} \begin{bmatrix} U \\ G \end{bmatrix} = \begin{bmatrix} \bar{F} \\ G_2 \end{bmatrix}. \tag{3.9}$$

The solution U and G are the discrete forms of $u_{g^*}(x, y)$ and g^*, the solution of (1.23).

There exist many numerical methods that can solve (3.9). The GMRES method applied to (3.9) directly or the multi-grid approach [1] are two attractive choices. However, in order to take advantage of fast Poisson solvers, we choose to solve the Schur complement system for G in (3.9) first,then find the solution U by applying the IIM solver to (3.8) two times.

Consider the linear system written in the form (3.9), in which $\bar{A}$ is assumed to be nonsingular. From the first equation the unknown U can be expressed as

$$U = \bar{A}^{-1}(\bar{F} - D^{-1}G). \tag{3.10}$$

Upon substituting this into the second equation, the following reduced system is obtained

$$\begin{aligned} (-E\bar{A}^{-1}D^{-1})G &= G_2 - E\bar{A}^{-1}\bar{F} \\ &= R. \end{aligned} \tag{3.11}$$

The matrix

$$S = -E\bar{A}^{-1}D^{-1} \tag{3.12}$$

is the Schur complement matrix associated with the variable G. It is an $n_b \times n_b$ system, a much smaller linear system compared to the one for U. Here n_b is the number of boundary control points. In practice, the coefficient matrices S, E, $\bar{A}$, D and the vectors $\bar{F}$, R are never formed. The matrix and vector form are merely for theoretical purposes. Thus an iterative method, such as the GMRES iteration, is preferred. The GMRES(Generalized Minimum Residual method is designed to solve nonsymmetric linear systems[37]. It is a kind of Krylov-subspace method which ensures convergency in no more than n_b steps. Here in this thesis, we use a restart form of GMRES method based on the modified Gram-Schmidt procedure to control the storage requirement. When the linear system (3.11) is solved, we can then use put G back into (3.10) to get U.

In short, our solution method based on a approach involves three steps:

- Obtain the right hand side R of the reduced system (3.11).
- Solve the reduced system (3.11) for G.
- Back-substitute using (3.10) to obtain U.

Below we will discuss in detail on how to implement the iteration process without forming the coefficient matrices.

3.2 Some implementation details

The main process of our algorithm is to solve the Schur complement system (3.11) using the GMRES method with an initial guess

$$G^{(0)} = \{g_1^{(0)},\ g_2^{(0)}, \cdots,\ g_{n_b}^{(0)}\}. \tag{3.13}$$

We first derive the right hand side, then compute the matrix-vector multiplication of the system without explicitly forming the coefficients matrix.

In order to compute the right hand side R of the Schur complement system, we take $G = 0$ and apply two steps of the immersed interface method to solve (3.10) to get $U(0)$, then

$$U(0) = \bar{A}^{-1}\bar{F}. \tag{3.14}$$

With the knowledge of $U(0)$, we can compute the normal derivative $U_n(0)$ on the boundary using the one-side interpolation approach described in the previous section. Thus the right hand side of the Schur complement system is

$$\begin{aligned} R &= G_2 - E\bar{A}^{-1}\bar{F} \\ &= G_2 - EU(0). \end{aligned} \tag{3.15}$$

In discrete form, it is

$$r^k = g_2^k - \sum_{i,j} \gamma_{ij}^k u_{ij}^k(0) d_\alpha(|\vec{X}^k - \vec{x}_{ij}|). \tag{3.16}$$

Note here

$$EU(0) = \sum_{i,j} \gamma_{ij}^k u_{ij}^k(0) d_\alpha(|\vec{X}^k - \vec{x}_{ij}|) \tag{3.17}$$

is derived from (2.58).

Now we are able to compute the right hand side of the Schur complement system. The next step is to compute the matrix-vector multiplication of the Schur complement system. Consider the left hand side

$$(-E\bar{A}^{-1}D^{-1})G \tag{3.18}$$

of the Schur complement. Given G, computing this matrix-vector multiplication involves essentially two steps.

1. First we use the immersed interface method for computing

$$U(G) = \bar{A}^{-1}(\bar{F} - D^{-1}G). \tag{3.19}$$

   This involves two steps of IIM solver. First an IIM solver is used to solve for $V(G)$ from (2.44). Then the computed $V(G)$ is substituted into the linear system (2.43) and the second IIM step is used to solve for $U(G)$. After get $U(G)$, we use the similar process for computing the right hand side of the Schur complement to get $EU(G)$.

2. The weighted least squares interpolation is used to compute $U_n(G)$. In practice, (3.18) is written as

$$\begin{aligned}(-E\bar{A}^{-1}D^{-1})G &= E(-\bar{A}^{-1}D^{-1}G)\\ &= E(U(G) - \bar{A}^{-1}\bar{F})\\ &= EU(G) - E\bar{A}^{-1}\bar{F}\\ &= EU(G) + R - G_2.\end{aligned} \tag{3.20}$$

   The last equality is obtained from (3.15). In discrete form, it can be written as

$$\sum_{i,j} \gamma_{ij}^k u_{ij}^k(G) d_\alpha(|\vec{X}^k - \vec{x}_{ij}|) + r_k - g_2^k. \tag{3.21}$$

It is worth to point out that once our algorithm is successfully terminated, which means that the residual vector is close to the zero vector, we not only have an approximation $U(G)$ to the solution U, but also an approximation G to the Laplacian ΔU on the boundary. The Laplacian information on the boundary is very useful for the stream function formulation for the Navier-Stokes problems where it stands for the vorticity term.

3.3 Numerical results

We have done a number of numerical experiments which confirm the expected order of accuracy and the order of convergency rate. The computation are done using Sun Ultra 10 workstations. The fast Poisson solver used here is from Fishpack developed by the National Center For Atmospheric Research, and the tolerance is 10^{-6}.

Example 3.1 *In this example we consider a biharmonic problem defined on a circle* $x^2 + y^2 = 1/4$ *with the exact solution*

$$u(x, y) = x^2 + y^2 + e^x \qquad (x, y) \in \partial\Omega. \tag{3.22}$$

The forcing term $f(x, y)$ *is obtained from* u*.*

$$f(x, y) = e^x. \tag{3.23}$$

The normal derivative on the boundary is

$$u_n(x, y) = 8x^2 + 4xe^x + 8y^2 \qquad (x, y) \in \partial\Omega \tag{3.24}$$

where $\partial\Omega$ *is the boundary of the circle. The computation domain is* $[-1, 1] \times [-1, 1]$*.*

mesh size	$\|E_n\|_\infty$	r_1	$\|T_n\|_\infty$	r_2
64×64	3.0003×10^{-4}		1.6169×10^{-2}	
128×128	8.0076×10^{-5}	3.74678	4.2197×10^{-3}	3.8317
256×256	1.9710×10^{-5}	4.06270	1.6806×10^{-3}	2.5108
512×512	4.7654×10^{-6}	4.13604	5.8031×10^{-4}	2.8961

Table 3.1: Grid refinement analysis for example (3.1).

Table (3.1) shows the results of a grid refinement study, where n is the number of uniform grid points in x direction and y direction. The maximum error over all grid points,

$$\|E_n\|_\infty = \max_{i,j} |u(x_i, y_j) - u_{ij}|, \tag{3.25}$$

is presented, where u_{ij} is the computed approximation at the uniform grid points (x_i, y_j). We also display $\|T_n\|_\infty$, the infinity norm of the local truncation error over all grid points. The local truncation errors are $O(h^2)$ expected at those points which are close to the interface where they are $O(h)$. We also display the ratios of successive errors $\|E_n\|_\infty$,

$$r_1 = \frac{\|E_n\|_\infty}{\|E_{2n}\|_\infty} \tag{3.26}$$

as well as the ratios of successive errors $\|T_n\|_\infty$

$$r_2 = \frac{\|T_n\|_\infty}{\|T_{2n}\|_\infty}. \tag{3.27}$$

A ratio of 2 corresponds to first order accuracy, while a ratio of 4 indicates second order accuracy. The same notations are used for the rest of examples in this section.

We also did the simulation on 14 other different grids from 40×40 to 560×560. Together with the previous 4 grids, we have 18 points. By doing the linear regression on the errors, we can see from Fig. 3.4 that the order of accuracy is 1.9395 which is very close to the expected second order accuracy.

Example 3.2 *In this example we consider a biharmonic problem defined on a ellipse* $\frac{x^2}{0.5^2} + \frac{y^2}{0.15^2} = 1$. *The differential equation is:*

$$\Delta^2 u = \frac{24y}{(1+x)^5} - \frac{12x}{(1+y)^2} - \frac{6x^3}{(1+y)^4}. \tag{3.28}$$

We use the Dirichlet boundary condition which is determined from the exact solution

$$u(x, y) = x^3 \ln(1+y) + \frac{y}{1+x} \qquad (x, y) \in \partial\Omega \tag{3.29}$$

and the boundary normal derivative $u_n(x, y)$ *is computed through the exact solution and the level set function. The computation domain is* $[-0.6, 0.6] \times [-0.3, 0.3]$.

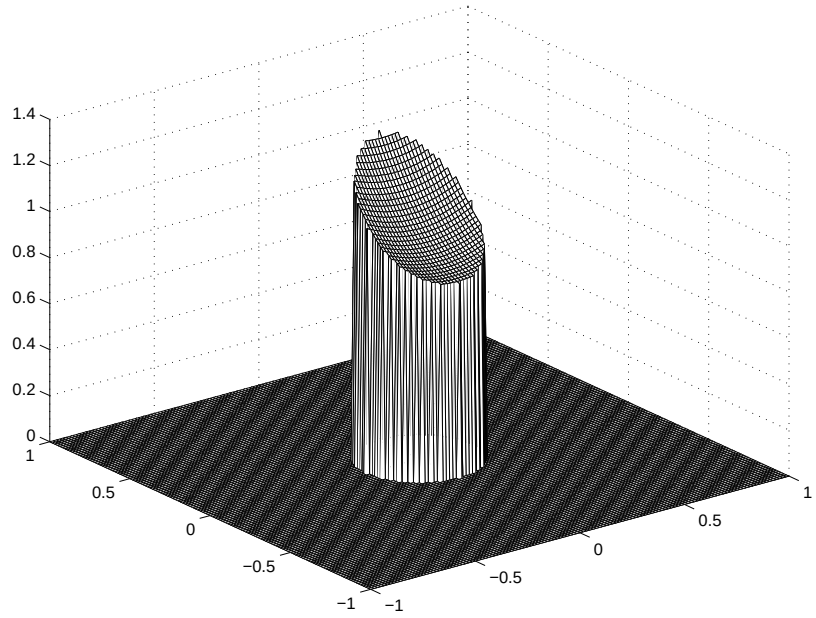


Figure 3.1: The exact solution for example (3.1).

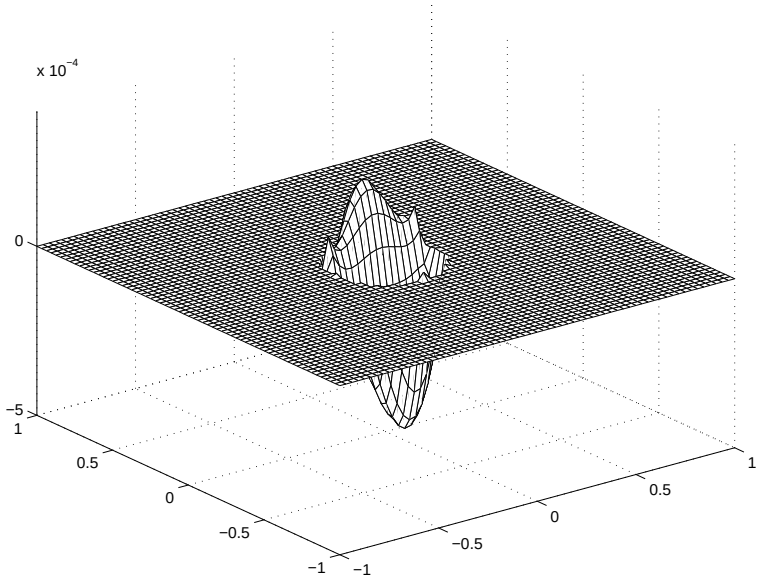


Figure 3.2: Error of the computed solution for example (3.1) when n=m=64.

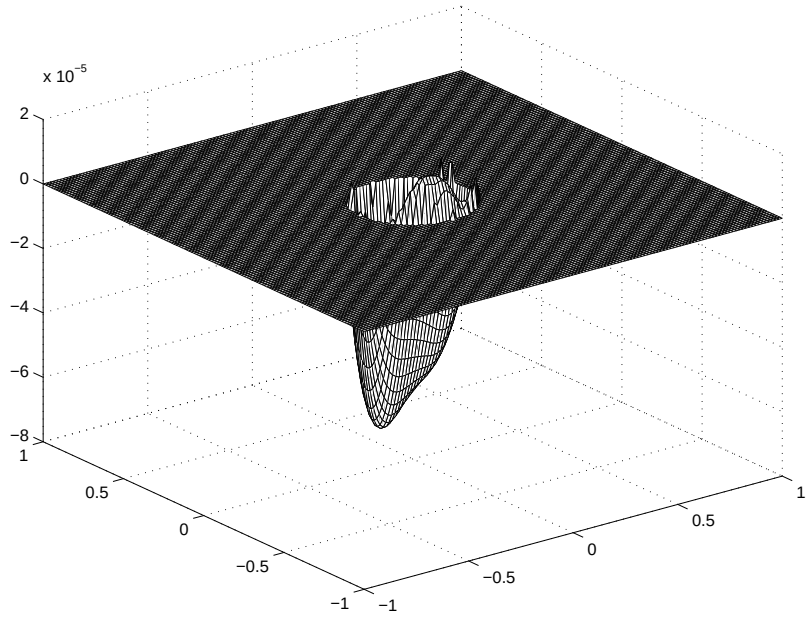


Figure 3.3: Error of the computed solution for example (3.1) when n=m=128.

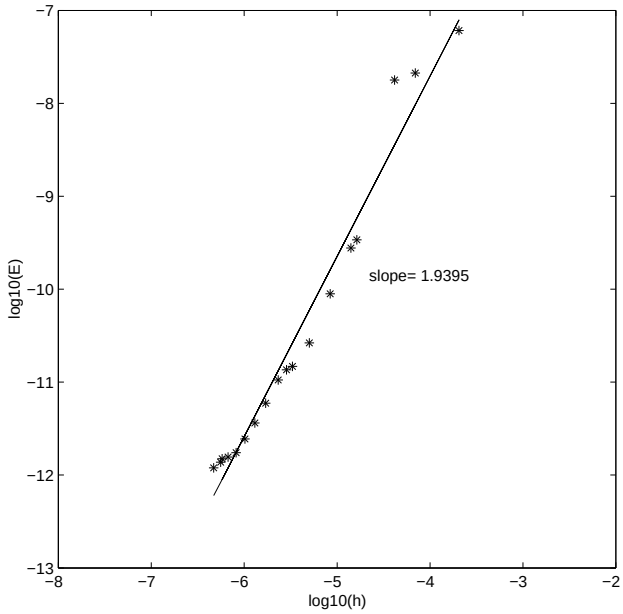


Figure 3.4: The linear regression analysis on the errors for example (3.1).

mesh size	$\|E_n\|_\infty$	r_1	$\|T_n\|_\infty$	r_2
64×32	3.6576×10^{-4}		6.6260×10^{-2}	
128×64	9.5430×10^{-5}	3.8327	2.9425×10^{-2}	2.2518
256×128	2.0889×10^{-5}	4.5684	1.6662×10^{-2}	1.7660
512×256	4.9899×10^{-6}	4.1862	5.213×10^{-3}	3.1962

Table 3.2: A grid refinement analysis for example (3.2).

Table (3.2) shows the results of a grid refinement study. We can see that the method still has second order accuracy when we use a smaller rectangle to embed the irregular ellipse region.

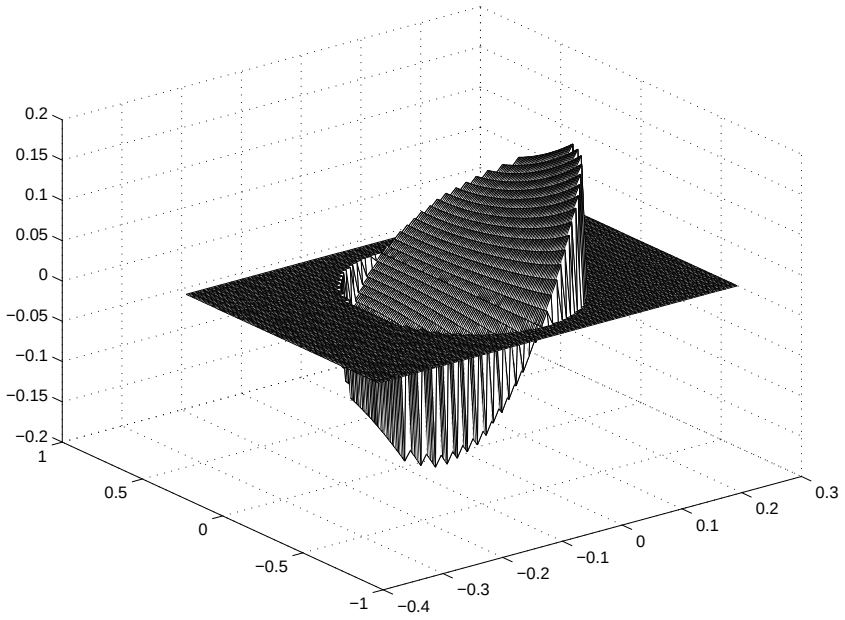


Figure 3.5: The exact solution for example (3.2).

Again we did the simulation on 14 other different grids from 40×40 to 560×560. By doing the linear regression on the errors, we can see from Fig. 3.8 that the order of accuracy is 2.0240.

Example 3.3 *In this example we consider a biharmonic problem defined on a five-star region*

$$\begin{cases} x = & r(\theta)\cos(\theta) \\ y = & r(\theta)\sin(\theta) \end{cases} \tag{3.30}$$

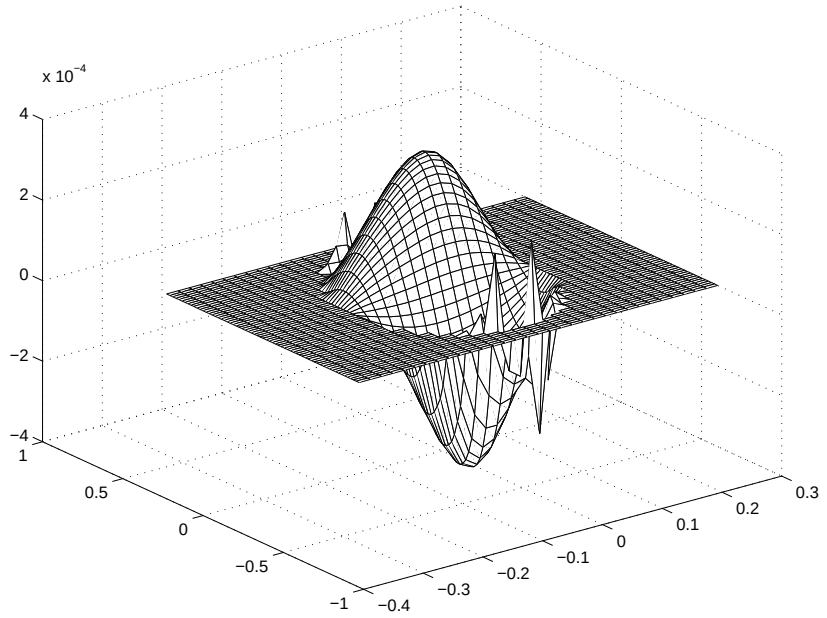


Figure 3.6: Error of the computed solution for example (3.2) with 64×32 mesh.

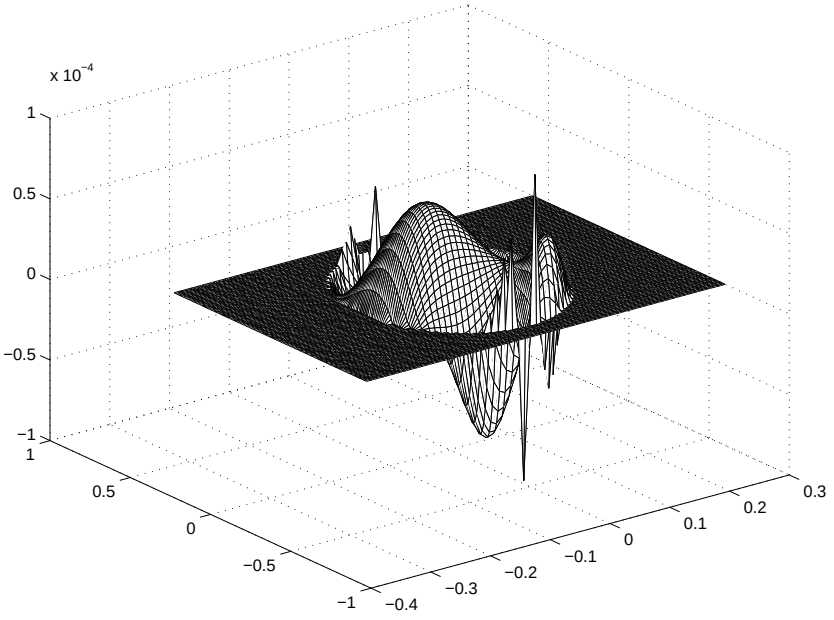


Figure 3.7: Error of the computed solution for example (3.2) with 128×64 mesh.

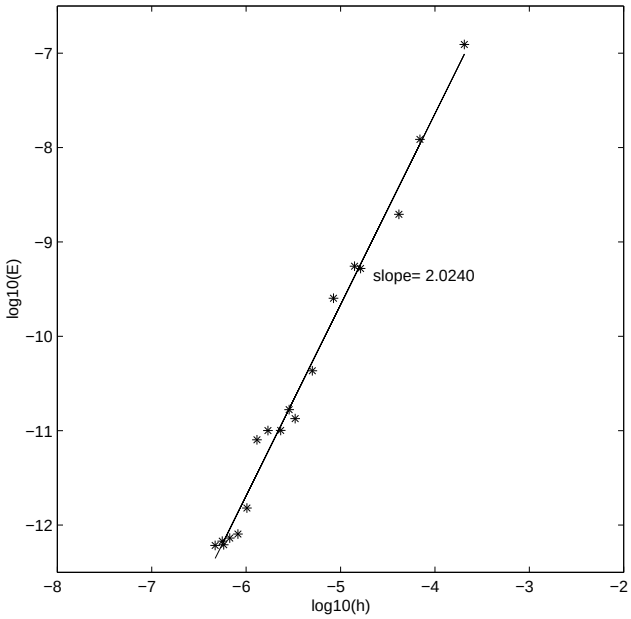


Figure 3.8: The linear regression analysis on the errors for example (3.2).

where

$$r(\theta) = 0.6 + 0.25\sin(5\theta) \tag{3.31}$$

and $\theta \in [0, 2\pi)$. *The differential equation is:*

$$\Delta^2 u = 0. \tag{3.32}$$

We use the Dirichlet boundary condition which is determined from the exact solution

$$u(x, y) = x^2 + y^2 + e^x \cos(y) \tag{3.33}$$

and the boundary normal derivative $u_n(x, y)$ *is computed by interpolating the exact solution.*

Table (3.3) shows the results of a grid refinement study.

By doing the linear regression on the errors, we can see from Fig. 3.12 that the order of accuracy is 1.9300 for this example.

Example 3.4 *In this example we consider a biharmonic problem defined on a disconnected region. The level set function is*

$$\phi(x, y) = \min(S_1(x, y), S_2(x, y)) \tag{3.34}$$

mesh size	$\|E_n\|_\infty$	r_1	$\|T_n\|_\infty$	r_2
64×64	8.14721×10^{-4}		5.95697×10^{-2}	
128×128	2.08149×10^{-4}	3.91412	2.205692×10^{-2}	2.70072
256×256	8.06897×10^{-5}	2.57962	1.15031×10^{-2}	1.91747
512×512	1.83151×10^{-5}	4.40563	7.89028×10^{-3}	1.45788

Table 3.3: A grid refinement analysis for example (3.3).

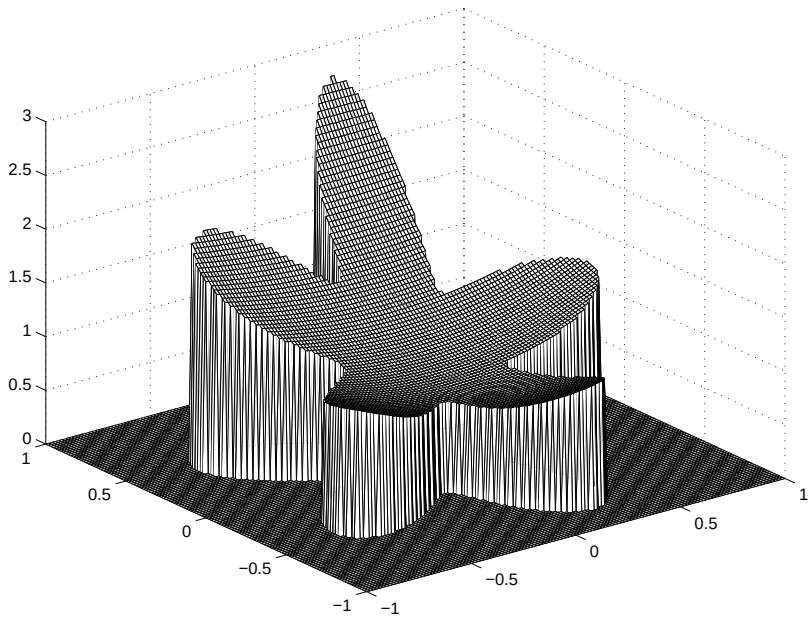


Figure 3.9: The exact solution for example (3.3).

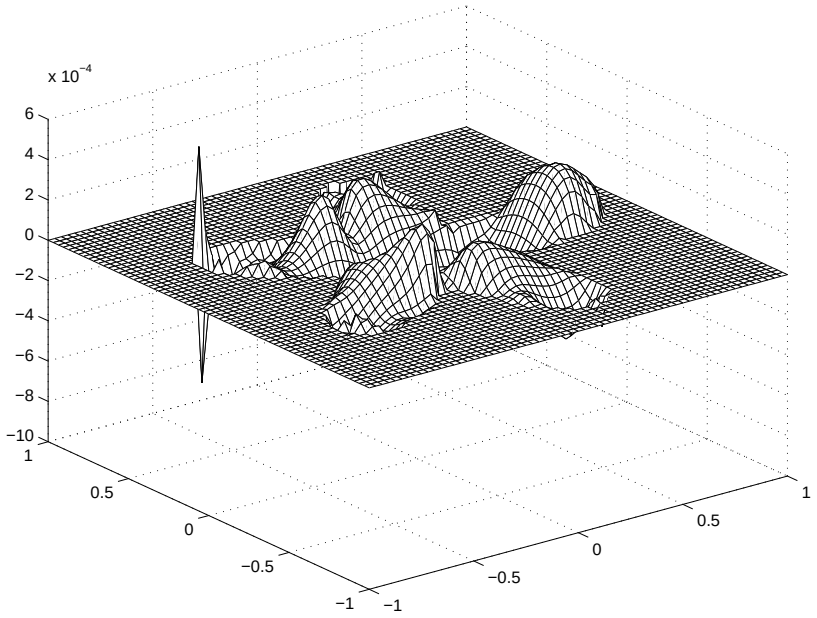


Figure 3.10: Error of the computed solution for example (3.3) when n=m=64.

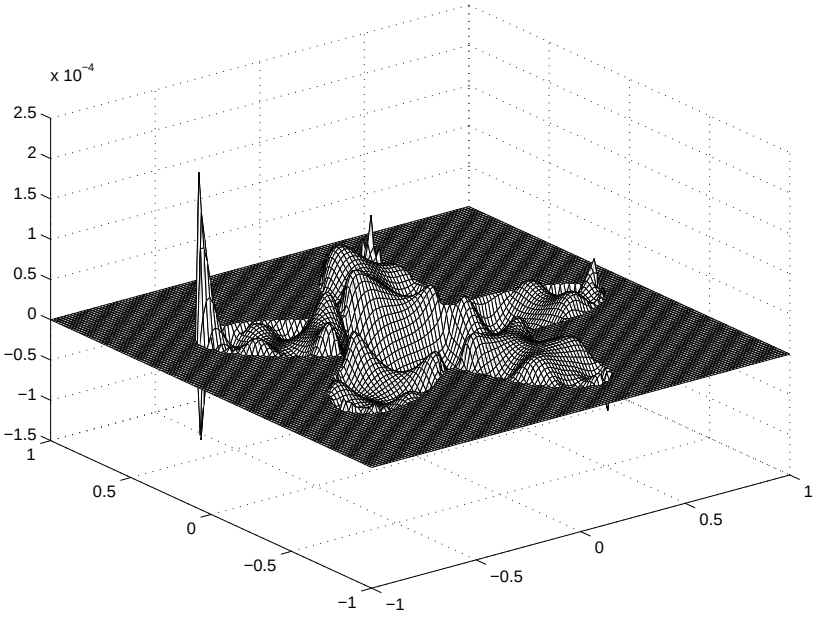


Figure 3.11: Error of the computed solution for example (3.3) when n=m=128.

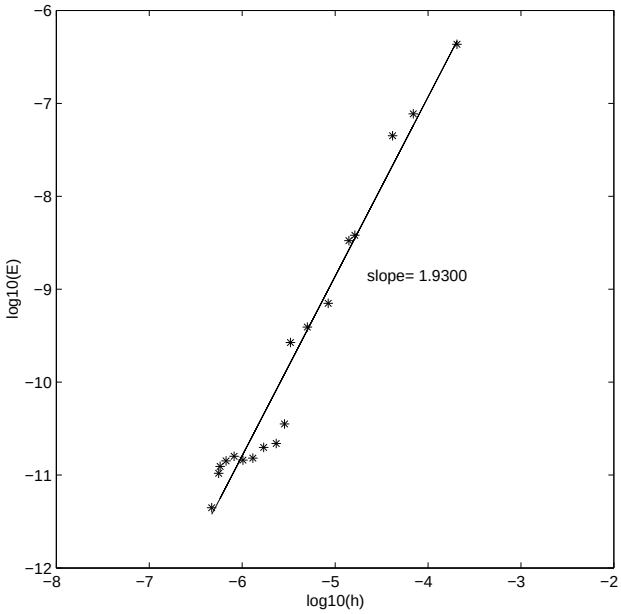


Figure 3.12: The linear regression analysis on the errors for example (3.3).

where

$$\begin{aligned} S_1(x,y) &= \sqrt{x^2+(y+0.3)^2} - 0.2 \\ S_2(x,y) &= \sqrt{x^2+(y-0.3)^2} - 0.12. \end{aligned} \tag{3.35}$$

It is obvious that $S_1(x,y)$ *is a circle centered at* $(0, \ -0.3)$ *and* $S_2(x,y)$ *is another circle centered at* $(0, \ 0.3)$. *The differential equation and the boundary are the same as example* (3.1).

Table (3.4) shows the results of a grid refinement study. We can see again the average ration r_1 is close to 4 indicating second order accuracy.

3.3.1 Algorithm efficiency analysis

CPU time and iterations steps versus geometry. Table (3.5)-Table (3.8) show the number of iterations and the computation time of different grid refinement analysis for examples in this section. Here n_{irreg} denotes the total number of irregular grid points, n_{irreg}^{in} is the number of inside irregular grid points, and t_{solve} is the CPU time for solving the linear system. The time unit is second(s). We can see the number of iterations needed in the entire solution process depends mostly on the geometry of the irregular region. Our algorithm iterates only about 4 steps for regular circular regions. Even for the complicated five star region in our simulation, it only takes 9

mesh size	$\|E_n\|_\infty$	r_1	$\|T_n\|_\infty$	r_2
64×64	7.24776×10^{-4}		4.71738×10^{-2}	
128×128	8.63084×10^{-5}	8.39751	1.57728×10^{-2}	2.99083
256×256	3.26879×10^{-5}	2.64723	1.17202×10^{-2}	1.34578
512×512	8.13915×10^{-6}	4.01474	4.04705×10^{-3}	2.89604

Table 3.4: A grid refinement analysis for example (3.4).

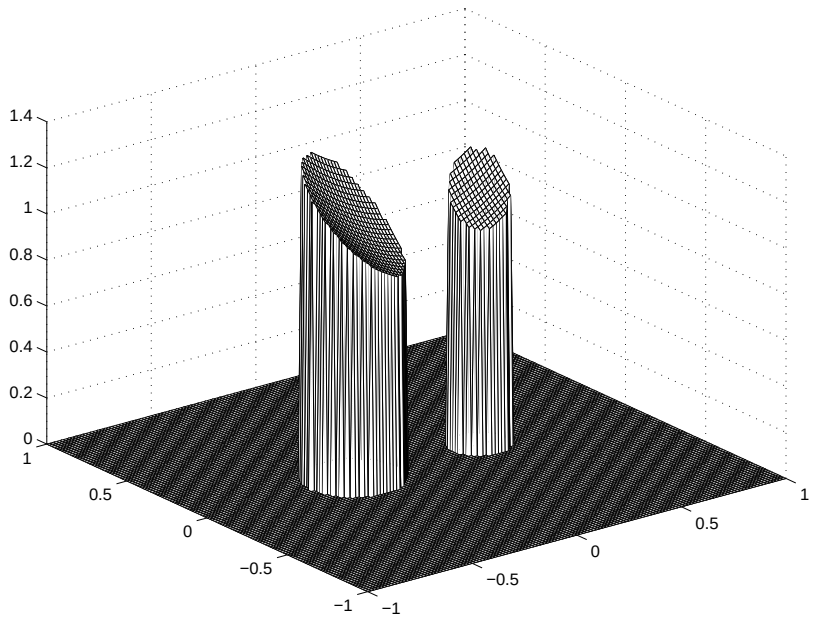


Figure 3.13: The exact solution for example (3.4).

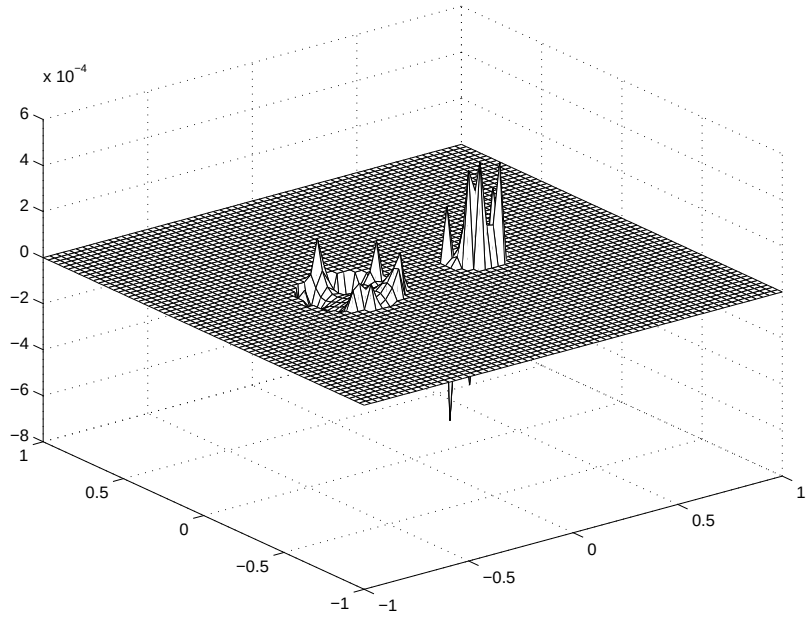


Figure 3.14: Error of the computed solution for example (3.4) when n=m=64.

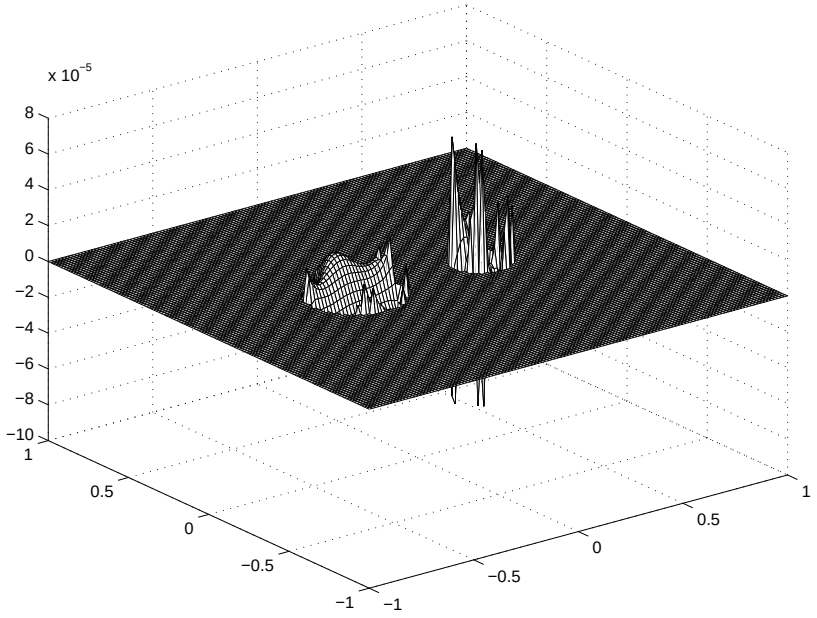


Figure 3.15: Error of the computed solution for example (3.4) when n=m=128.

iteration steps. At the same time, the number of iterations seems independent of the size of the gird. Once the problem is given and the boundary is set, the iteration steps almost do not change at all.

Cost of dealing with irregular grid points. Another concern about our algorithm is how much overhead cost is needed for dealing with the irregular points. The cost includes indexing the irregular grid points, finding projections, and solving a linear system for interpolation coefficients at each irregular point. It depends on the number of irregular points. As shown in Table (3.9) and (3.10), the cost is linearly proportional to the number of irregular points, which is proportional to the mesh size. But the cost of solving the final linear system is not just linearly proportional to the mesh size. Actually, even for a small mesh size n, the cost for dealing with irregular points does not make a big part of the total computation cost. As n increases, its contribution to computation cost becomes smaller and smaller. In both tables, t_{ov} denotes the total CPU time excluding solving the final linear system.

3.3.2 Summary of the computation scheme

In summary, we have developed a second order accurate difference method for biharmonic equations in irregular regions. In our method, the biharmonic equation is decomposed into two coupled Poisson equations. A refined least square formulation is adopted to interpolate the Laplacian Δu on the boundary from one side. We use a GMRES process to iteratively solve for the boundary Δu. At each iterate step, fast IIM solvers are used to solve the coupled Poisson equations on irregular region.

Based on this difference method, we have designed a numerical program. This program has been used successfully in many testing problems. It is proven to be able to deal with many complicated boundaries and disconnected regions.

mesh size	n_{irreg}	n_{irreg}^{in}	iter	t_{solve}
64×64	92	44	4	0.2510
128×128	180	88	5	0.7709
256×256	364	180	5	2.3530
512×512	724	360	4	10.3750

Table 3.5: The number of iterations and CPU time for Example (3.1)

mesh size	n_{irreg}	n_{irreg}^{in}	iter	t_{solve}
64×64	132	64	8	0.7310
128×128	268	132	4	1.08099
256×256	532	264	5	4.33599
512×512	1068	532	6	21.0000

Table 3.6: The number of iterations and CPU time for Example (3.2)

mesh size	n_{irreg}	n_{irreg}^{in}	iter	t_{solve}
64×64	372	184	9	2.2130
128×128	752	374	7	3.7360
256×256	1504	750	9	13.5200
512×512	3008	1502	9	52.0349

Table 3.7: The number of iterations and CPU time for Example (3.3)

mesh size	n_{irreg}	n_{irreg}^{in}	iter	t_{solve}
64×64	112	52	4	0.4310
128×128	232	112	4	0.9420
256×256	464	228	4	2.9040
512×512	928	460	5	16.5339

Table 3.8: The number of iterations and CPU time for Example (3.4)

n	Example (3.1)			Example (3.2)		
	n_{irreg}	t_{ov}	t_{solve}	n_{irreg}	t_{ov}	t_{solve}
64	92	0.0499	0.2510	132	0.0700	0.7310
128	180	0.1100	0.7749	268	0.1700	1.08099
256	364	0.1210	2.3530	532	0.1801	4.33599
512	724	0.3999	10.3750	1068	0.2210	21.0000

Table 3.9: The CPU time in dealing with irregular grid points and the CPU time for the linear solver for Example (3.1) and (3.2)

n	Example (3.3)			Example (3.4)		
	n_{irreg}	t_{ov}	t_{solve}	n_{irreg}	t_{ov}	t_{solve}
64	372	0.0699	2.2130	112	0.0600	0.4310
128	752	0.2710	3.7360	232	0.1300	0.9420
256	1504	0.5410	13.5200	464	0.2799	2.9040
512	3008	0.6610	52.0349	928	0.5509	16.5339

Table 3.10: The CPU time in dealing with irregular grid points and the CPU time for the linear solver for Example (3.3) and (3.4)

Chapter 4

Applications of the biharmonic solver

4.1 Solving incompressible Stokes equation on an irregular domain

Depending on the magnitude of the Reynolds number $1/\mu$, numerical methods for solving incompressible Stokes equation in two dimensional space can be divided into two categories. The first one is based on the primitive variables formulation for problems with small to medium-sized Reynolds numbers, such as the projection method, see [4] [9]. Another one is the vorticity-stream function formulation for problems with large Reynolds numbers, see, for example, [14] [28] and the references therein. The numerical method presented in this section uses the vorticity-stream function formulation.

We consider a incompressible Stokes equation in two dimensional space

$$\begin{aligned} \mu\Delta u &= p_x - F_1, \qquad \mathbf{x} \in \Omega \\ \mu\Delta v &= p_y - F_2, \qquad \mathbf{x} \in \Omega \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned} \tag{4.1}$$

where μ is the fluid viscosity, p is the pressure, $\mathbf{u} = (u, v)$ is the velocity, and $\mathbf{F} = (F_1, F_2)$ is the force. Equations (4.1) are supplemented by the "no-slip" boundary condition.

$$\mathbf{u}(\mathbf{x}) = 0, \qquad \mathbf{x} \in \partial\Omega. \tag{4.2}$$

The vorticity function, which is a scalar in two-dimensional case, is defined by

$$\omega(\mathbf{x}) = (\nabla \times \mathbf{u})_z = -u_y + v_x, \qquad \mathbf{x} \in \Omega. \tag{4.3}$$

The first two equation of (4.4) can be written in vector form

$$\mu\Delta\mathbf{u} = \nabla p - \mathbf{F}, \qquad \mathbf{x} \in \Omega. \tag{4.4}$$

Taking curl of the above equation we get

$$-\mu\Delta\omega = (\nabla \times \mathbf{F})_z, \qquad \mathbf{x} \in \Omega. \tag{4.5}$$

The velocity field $\mathbf{u}$ is obtained by using (4.3) in conjunction with (4.2). This can be done via the stream-function formulation as follows. Due to $\nabla \cdot \mathbf{u} = 0$, there is a scalar function $\psi(\mathbf{x})$ such that

$$\mathbf{u}(\mathbf{x}) = \nabla^{\perp}\psi = (-\frac{\partial \psi}{\partial y}, \frac{\partial \psi}{\partial x}), \qquad \mathbf{x} \in \Omega. \tag{4.6}$$

Therefore from (4.3), we get

$$\Delta \psi = \omega. \tag{4.7}$$

Thus, equations (4.5), (4.7), along with the relation (4.6), serve as the vorticity stream-function formulation of the problem.

We note that the "no-slip" condition (4.2) and the relation (4.6) implies

$$\psi(\mathbf{x}) = \frac{\partial}{\partial \mathbf{n}}\psi(\mathbf{x}) = 0, \qquad \mathbf{x} \in \partial\Omega \tag{4.8}$$

where $\mathbf{n}$ is the unit normal vector of the boundary $\partial\Omega$ pointing outward. Note that (4.8) follows since $\psi = const.$ on the boundary and is only determined up to an additive constant.

Substituting (4.5) into (4.7), using the relation (4.8), we get

$$\begin{cases} -\mu\Delta^2\psi &= (\nabla \times \mathbf{F})_z, & \mathbf{x} \in \Omega \\ \psi(\mathbf{x}) &= 0, & \mathbf{x} \in \partial\Omega \\ \psi_{\mathbf{n}}(\mathbf{x}) &= 0, & \mathbf{x} \in \partial\Omega. \end{cases} \tag{4.9}$$

This is a well defined biharmonic problem. The advantage of using the formulation to solve for the velocity field $\mathbf{u}$ is that it eliminates the difficulty of dealing with the improper partition of boundary conditions of the vorticity-stream function formulation. We assume that the boundary of the domain, denoted as $\partial\Omega$ is piecewise smooth.

We use the algorithm developed in previous chapter to solve (4.9) numerically using a Cartesian grid that enclosed in the domain Ω. Once ψ is obtained, the standard central difference formula can be used to interpolate u and v on regular grid points. For irregular grid points, we use the same one-side interpolation technique developed in Chapter 2 and second order accuracy is preserved.

We tested our algorithm for a stokes flow with Reynolds number 400. The stream function is

$$\psi(x, y) = \frac{1}{\pi}\sin^2(\frac{x^2 + y^2}{4}\pi) - \frac{1}{\pi}. \tag{4.10}$$

The fluid is confined in a circular region

$$\Omega = \{(x, y)|\ x^2 + y^2 = 2\} \tag{4.11}$$

and the computation region is choosing to be $[-1.6, 1.6] \times [-1.6, 1.6]$. We have the no-slip boundary condition

$$\begin{aligned} \psi(x, y) &= 0 \qquad (x, y) \in \partial\Omega \\ \psi_n(x, y) &= 0 \qquad (x, y) \in \partial\Omega. \end{aligned} \tag{4.12}$$

Table 4.1 show the results of a grid refinement study for u and v.

mesh size	$\|E(u)_n\|_\infty$	r	$\|E(v)_n\|_\infty$	r
64×64	1.07739×10^{-2}		1.08006×10^{-2}	
128×128	2.54015×10^{-3}	4.2415	2.56114×10^{-3}	4.2171
256×256	4.49293×10^{-4}	5.6570	4.49161×10^{-4}	5.7037
512×512	1.09871×10^{-4}	4.0892	1.09842×10^{-4}	4.0907

Table 4.1: A grid refinement analysis of $\mathbf{u} = (u, v)$ for example 4.10–4.12.

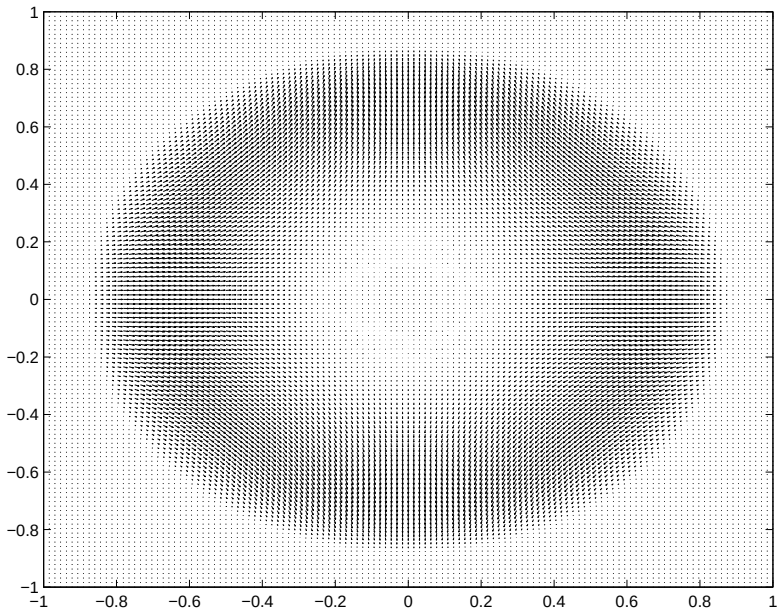


Figure 4.1: The exact velocity field for example 4.10–4.12.

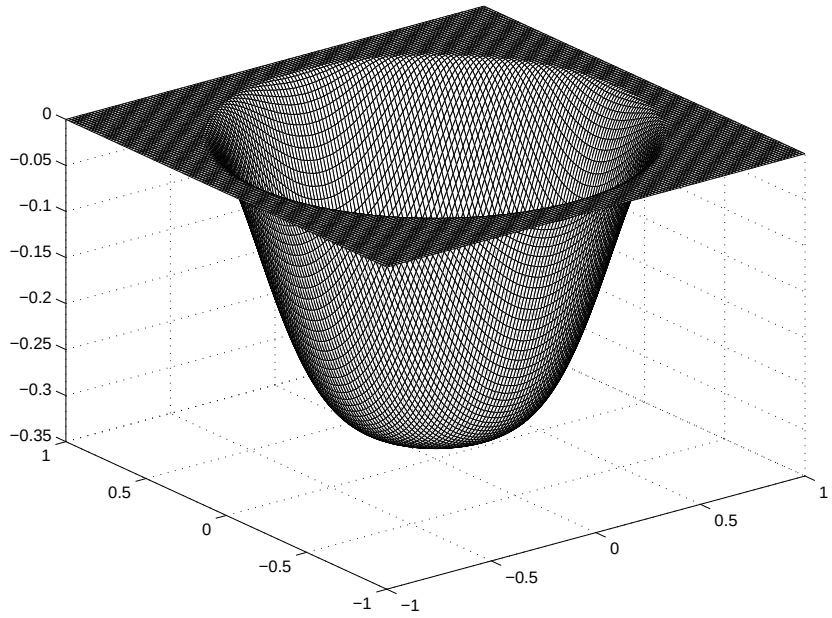


Figure 4.2: Plot of the exact stream function ψ for example 4.10–4.12.

4.2 Solving the biharmonic equation with a linear parameter

In this section, we discuss the application of the fast algorithm for the biharmonic equation with a linear parameter:

$$\begin{cases} \Delta^2 u(x,y) - \lambda \Delta u(x,y) = f(x,y) & (x,y) \in \Omega \\ u(x,y) = g_1(x,y) & (x,y) \in \partial\Omega \\ u_n(x,y) = g_2(x,y) & (x,y) \in \partial\Omega. \end{cases} \tag{4.13}$$

This problem arise from using the stream-function formulation for solving the Navier-Stokes equation. When considering incompressible viscous flow in a two-dimensional domain, we know the motion of the fluid is governed by the Navier-Stokes equations

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} &= -\nabla p + \nu \Delta \mathbf{u} \\ \nabla \cdot \mathbf{u} &= 0. \end{aligned} \tag{4.14}$$

Interpreted as an equation for the stream function, the first equation in (4.14) can be written as

$$\frac{\partial}{\partial t}\Delta\psi + [(\nabla^{\perp}\psi) \cdot \nabla]\Delta\psi = \nu \Delta^2 \psi. \tag{4.15}$$

To solve the above equation numerically, the temporal and the spatial discretizations can be considered separately. This is legitimate when the scheme has a well-behaved semidiscrete

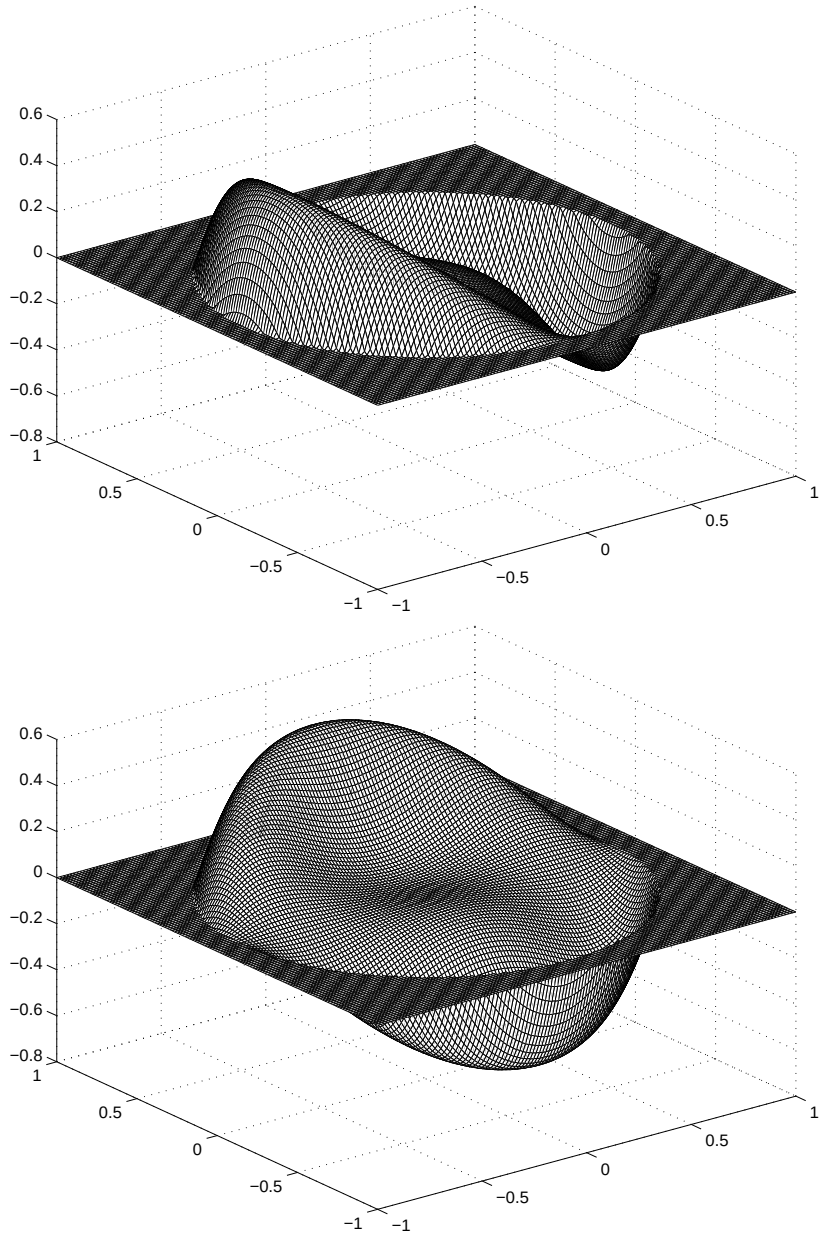


Figure 4.3: Plot of the exact solution **u** for example 4.10–4.12.

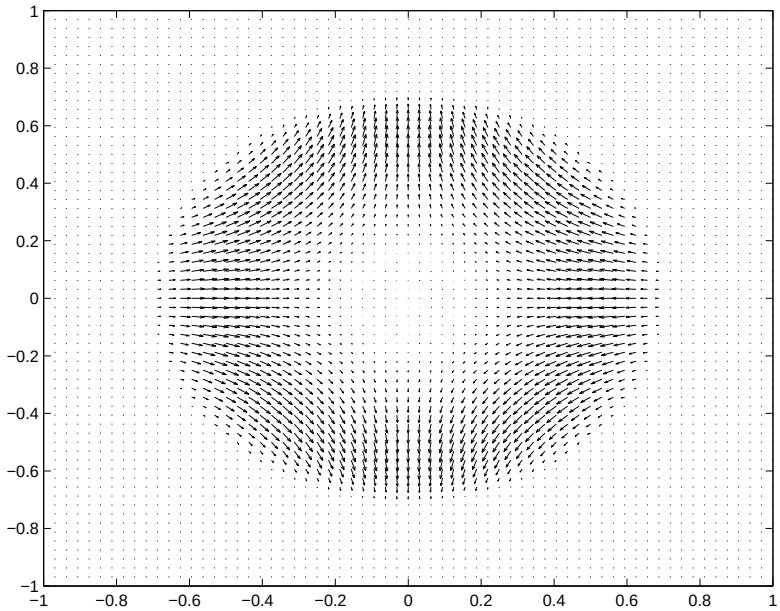


Figure 4.4: The computed velocity field for example 4.10–4.12 when n=m=64.

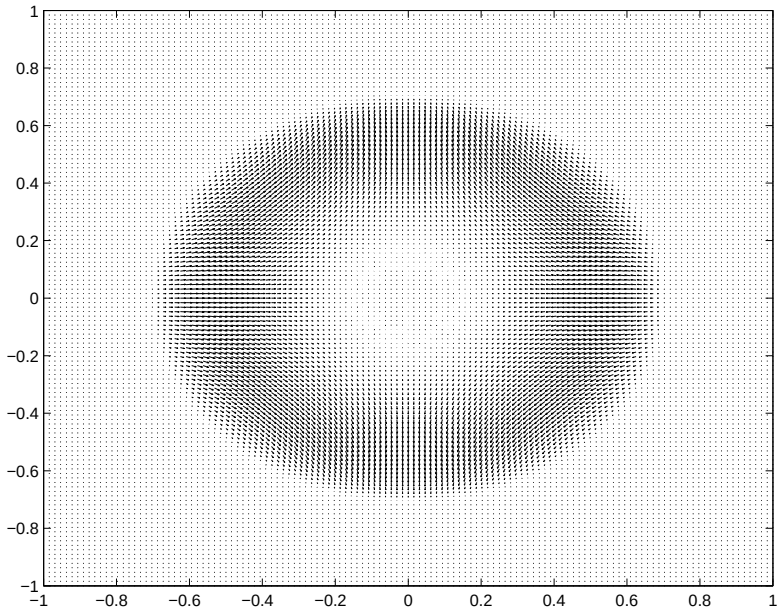


Figure 4.5: The computed velocity field for example 4.10–4.12 when n=m=128.

limit [23]. Kupferman [22] used this approach. He use Crank-Nicholson formula for the viscous term and a midpoint rule for the advection term. Hence (4.15) is approximated by a discretization that is second order in time:

$$\begin{aligned}(\Delta - \frac{1}{4}\nu\kappa\Delta^2)\psi^{n+\frac{1}{2}} &= (\Delta + \frac{1}{4}\nu\kappa\Delta^2)\psi^n - \frac{1}{2}\kappa[(\mathbf{u}\cdot\nabla)\omega]^n \\ (\Delta - \frac{1}{2}\nu\kappa\Delta^2)\psi^{n+1} &= (\Delta + \frac{1}{2}\nu\kappa\Delta^2)\psi^n - \kappa[(\mathbf{u}\cdot\nabla)\omega]^{n+\frac{1}{2}}.\end{aligned} \tag{4.16}$$

Here ψ^n denotes the stream function at time t^n, and $\kappa = t^{n+1} - t^n$ is the time step interval.

There are a number of implementation issues concerning the above approach. From the computational point of view, the most difficult part is the solution of a linear system of the form

$$(\lambda\Delta_h - \Delta_h^2)\psi_{i,j} = rhs_{i,j} \tag{4.17}$$

where $rhs_{i,j}$ stands for the right hand side of the linear system which can be calculated directly by some finite difference schemes. This difference equation is resulted from the spatial discretization of (4.16); two such linear systems need to be solved at every time step. (4.17) is a biharmonic equation with a linear parameter. Standard iterative methods are known to converge very slowly, if at all, for biharmonic operators. After some modification to the algorithm, our computation method is a very good choice for such problems, especially when they are defined on irregular regions.

Following the analysis done in section (2.1), we can write the solution $u_g(x, y)$ of the following problem as a functional of $g(x, y) \in \partial\Omega$:

$$\begin{cases} \Delta u(x,y) = v(x,y) & (x,y) \in \Omega \\ u(x,y) = g_1(x,y) & (x,y) \in \partial\Omega, \end{cases}$$
$$\begin{cases} \Delta v(x,y) - \lambda v(x,y) = f(x,y) & (x,y) \in \Omega \\ v(x,y) = g(x,y) & (x,y) \in \partial\Omega. \end{cases} \tag{4.18}$$

Let the solution of problem (4.13) be $u^*(x, y)$, and define

$$g^*(x,y) = \Delta u^*(x,y) \qquad (x,y) \in \partial\Omega \tag{4.19}$$

along the boundary. Then $u^*(x, y)$ satisfies the biharmonic equation (4.13) and (4.18) with $g(x, y) \equiv g^*(x, y)$. In other words, $u_{g^*}(x, y) \equiv u^*(x, y)$ and:

$$\frac{u^*(x,y)}{\partial n} = g_2(x,y) \qquad (x,y) \in \partial\Omega \tag{4.20}$$

is satisfied. The second equation in (4.18) is a Helmholtz equation. This means now we need to solve a Poisson equation and a Helmholtz equation at every iterate step. This change can be easily done by simply supporting an additional parameter λ to the IIM solver in our algorithm.

We tested our algorithm for a biharmonic equation with a linear parameter. The stream function is given by:

$$\psi(x,y) = \frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y) \tag{4.21}$$

mesh size	iter	$\|E(\psi)_n\|_\infty$	r
64×64	5	4.56959×10^{-4}	
128×128	6	1.47798×10^{-4}	3.0917
256×256	7	5.51815×10^{-5}	2.6783
512×512	8	2.241133×10^{-5}	2.46221

Table 4.2: A grid refinement analysis of example 4.21–4.22

This function comes from a testing problem in [22]. We select λ to be 1. The fluid is confined in a ellipse region

$$\Omega = \{(x, y)|\ 4x^2 + 49y^2 = 1\} \tag{4.22}$$

The boundary condition and the right hand side can be computed directly from the stream function. The numerical simulation show the refined algorithm works very well for the biharmonic equation with a linear parameter.

Table 4.2 show the results of the grid refinement study.

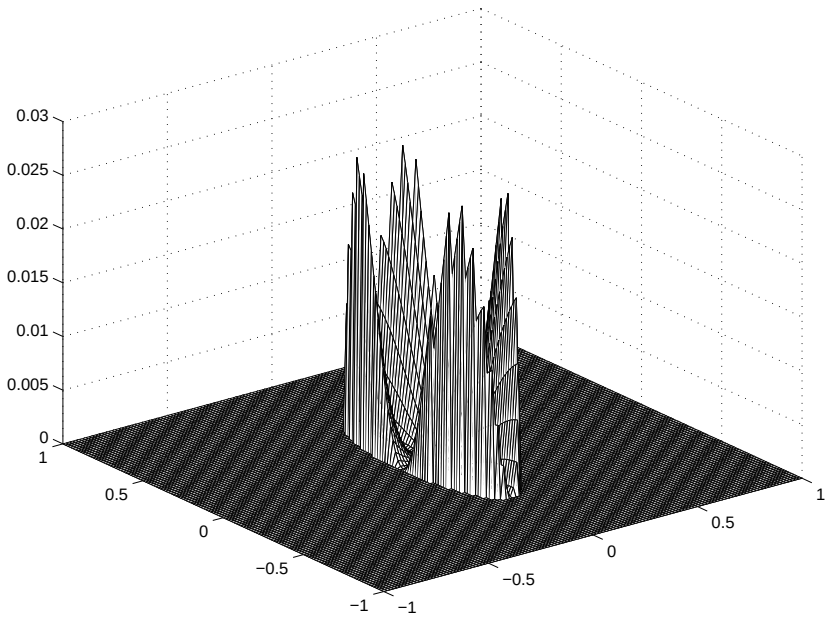


Figure 4.6: The exact solution for example 4.21–4.22

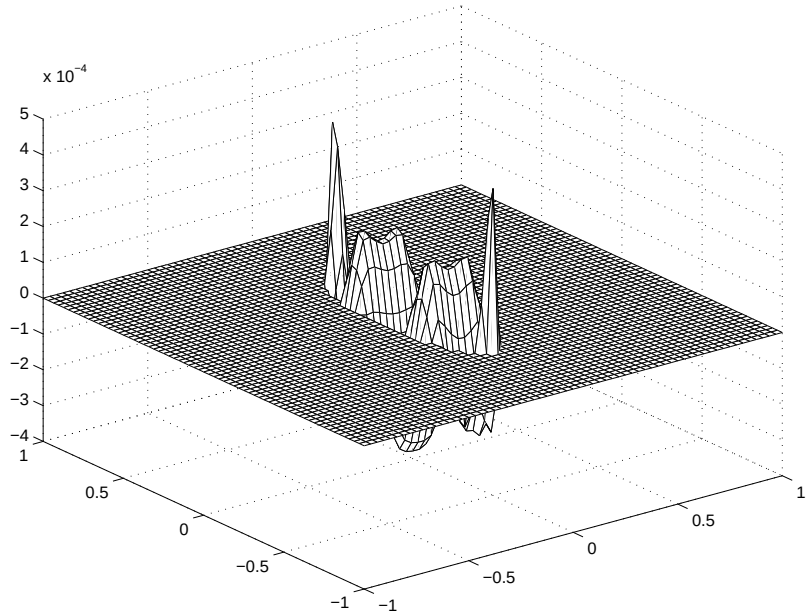


Figure 4.7: The error plot for example 4.21–4.22 when n=m=64

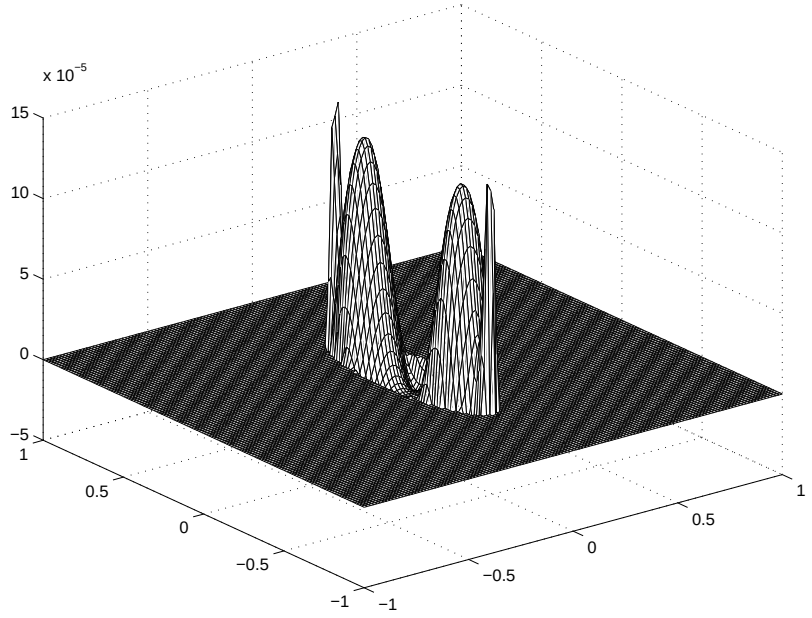


Figure 4.8: The error plot for example 4.21–4.22 when n=m=128

Chapter 5

Conclusions and future work

5.1 Thesis conclusions

In this thesis, We have presented a fast algorithm for solving two-dimensional biharmonic problems on irregular regions. To avoid mesh generation difficulties associated with unconstructed, body fitted grid, the irregular domain is embedded into a uniform Cartesian grid. The biharmonic equation is decomposed into two coupled Poisson equations. The solution of the coupled Poisson system is dependent on the Laplacian ΔU on the boundary. We use a weighted least square formulation to interpolate the Laplacian on the boundary from inside of the region. The accuracy of the interpolation scheme turns out to be a crucial step in solving the biharmonic problem for our algorithm.

The resulting linear system which involves both the solution and the Laplacian on the boundary. In order to take advantage of fast Poisson solvers, we use Generalized Minimum Residue method to solve for ΔU, and use the Immersed Interface Method to solve the coupled Poisson problem. Putting all these works together, we get a second order fast algorithm for solving biharmonic equations on irregular domain. Numerical analysis show the algorithm is very stable and the number of iterations of our method seems to be independent of the mesh size.

The highlights of the thesis include the following:

- A coupled equation approach is developed for solving the biharmonic equation in irregular regions. The algorithms has an order of accuracy between 1.8 and 2, depending on the geometry of the boundary.
- The Immersed Interface Method is used to solve the coupled Poisson equations on irregular domain at each iteration step. Determining the boundary quantities requires very little overhead. The major computation cost comes from solving the two linear systems of Poisson equations. Since IIM is based on fast Poisson solvers, the total arithmetic steps is of order $O(N \log N)$.
- The one-side "walk through" interpolation scheme based on weighted least squares method is much more accurate than other interpolation schemes tested in our method. And it is more efficient since only 9 inside grid points are used for interpolation to achieve second order accuracy.

- The program code was designed to take advantage of the algorithms in Fishpack and IIM solver. Using these algorithms, the results show very little evidence of numerical oscillations or diffusion.

5.2 Future work

The algorithm proves to be stable and converges very fast, especially for the testing problem from Kupferman's [22] implicit Runge-Kutta method for Navier-Stokes equation. Only $5 \sim 8$ iteration steps are needed for solving a bihrmonic equation with a linear parameter. The error obtained from a coarse 64×64 grid is $10^{-4} \sim 10^{-5}$ already. So we want to try applying our method to solve the biharmonic problem derived from the vorticity-stream function formulation for the Navier-Stokes problem.

Another idea of improvement comes from the one-sided interpolation scheme. As mentioned before, we found in our numerical simulation the accuracy of the one-side interpolation is crucial to the whole algorithm. A third order accurate interpolation schemes should be able to improve its performance. But this will result in requiring more grid points involved in the interpolation for every boundary control point.

In this thesis, we use a level set approach to express the boundary. As seen in the algorithm efficiency analysis in Chapter 3, the overhead for dealing with boundary quantities in very small. This is advantageous for free boundary problem or regular problems in three space dimension. There are also three dimensional IIM solver available [13]. We can also choose to extend our work to three dimensional biharmonic equations by adopting the new 3-D IIM solver into our algorithm.

List of References

[1] L. M. Adams. *A multigrid algorithm for immersed interface problem.* Copper Mountain Conference on Multigrid Methods, 1995.

[2] I. Altas, J. Dym, M. M. Gupta, R. P. Manohar. *Multigrid solution of automatically generated high-order discretizations for the biharmonic equation.* SIAM J. Sci. Comput., vol 19, No5:1578–1585, 1998.

[3] R. E. Bank. *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations, User's Guide 6.0.* SIAM, 1990.

[4] J. B. Bell, P. Collela and H. M. Glaz. *A second-order projection method for the incompressible Navier-Stokes equations.* J. Comput. Phys., 85:257-283 1989

[5] M. J. Berger, R. J. LeVeque. *Adaptive mesh refinement using wave-propagation method.* SIAM J. Num. Anal., 35:2298–2316, 1998.

[6] P. Bjørstad. *Fast numerical solution of the biharmonic dirichlet problem on rectangles.* J. SIAM. J. Nemer. Anal., vol 20, N 1, Feb. 1983.

[7] C. Börgers and O. Widlund. *On finite element domain imbedding methods.* SIAM J. Num. Anal., 27:963–978, 1990.

[8] J. Bramble, J. King. *A finite element method for interface problems in domains with smooth boundaries and interfaces.* Advances in Comput. Math., 6:109-138, 1996

[9] D. L. Brown, R. Cortez and M. L. Minion. *Accurate projection methods for the incompressible Navier-Stokes equations.* J. Comput. Phys., 168:464, 2001

[10] B. L. Buzbee, F. W. Dorr, J. A. George, G. H. Golub. *The direct solution of the discrete Poisson equation on irregular girds.* SIAM J. Num. Anal., 8:722-736, 1971.

[11] R. Cortez. *The method of regularized stokeslets.* Siam J. Sci. Comput., Vol.23, pp.1204-1225, 2001

[12] R. Courant, D. Hilbert. *Methods of mathematical physics.* Interscience, NewYork, 1953.

[13] S, Deng, K. Ito, and Z. Li. *Three dimensional elliptic solvers for interface problems and applications.* J. Comput. Phys., 18:215-243, 2003

[14] W. E and J. G. Liu. *Vorticity boundary condition and related issues for finite difference schemes.* J. Comput. Phys., vol 124 (1996), pp. 368382.

[15] L. W. Ehrlich. *Solving the biharmonic equation as coupled finite difference equations.* SIAM J. Num. Anal., vol. 8, No. 2, June 1971.

[16] R. Glowinski and O. Pironneau. *Numerical methods for the first biharmonic equation and for the two-dimensional Stokes problem.* SIAM Review., vol. 21:167-212, 1979.

[17] P. L. Gould. *Introdiction to linear elasicity.* Springer-Verlag, 1994.

[18] M. M. Gupta and R. Manohar. *Direct solution of biharmonic equation using noncoupled approach.* J. Comput. Phys., vol. 33:236-248, 1979.

[19] M. M. Gupta. *Spectrum transformation methods for divergent iteration.* NASA Techinal Memorandum, 103745 (ICOMP-91-02), 1991.

[20] T. Hou, Z. Li, S. Osher, and H. Zhao. *A hybrid method for moving interface problems with application to the Hele-Shaw flow.* J. Comput. Phys., 134:236-252, 1997

[21] H. Johansen P. Collella. *A Cartesian grid embedded boundary method for Poisson equations on irregular domains.* J. Comput. Phys., 147:60-85, 1998

[22] R. Kupferman. *A central-difference scheme for a pure stream function formulation of incompressible viscous flow.* SIAM J. Sci. Comput., vol. 23, No. 1, 2001

[23] A. Kurganov E. Tadmor. *New high-resolution central difference schemes for nonlinear conservation laws and convection-diffusion equations.* J. Comput. Phys., 160:214-282, 2000

[24] R. J. LeVeque. *Clawpack and Amrclaw – Software for high-resolution Godunov methods.* 4-th Intl. Conf. on Wave Propagation, Golden, Colorado, 1994.

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

[26] Z. Li. *A fast iterative algorithm for elliptic interface problems.* SIAM J. Num. Anal., 35:230–254, 1998.

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

[28] Z. Li, C. Wang. *A fast finite difference method for solving Navier-Stokes equations on irregular domains.* Comm. Math. Sci., Vol.1, pp.180-196, 2003

[29] Z. Li, H. Zhao, and H. Gao. *A numerical study of electro-migration voiding by evolving level set functions on a fixed cartesian grid.* J. Comput. Phys., 152:281-304, 1999

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

[31] J. W. McLaurin. *A general coupled equation approach for solving the biharmonic boundary value problem.* SIAM J. Num. Anal., vol 11:14-33, 1974

[32] S. Osher, and J. A. Sethian. *Front propogating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulation.* J. Comput. Phys., 79:12-49, 1988

[33] C. S. Peskin. *Numerical analysis of blood flow in the heart.* J. Comput. Phys., 25:220-252, 1988.

[34] C. S. Peskin. *Lectures on mathematical aspects of physiology.* Lectures in Appl. Math., 19:69–107, 1981.

[35] C. S. Peskin and D. M. McQueen. *A general method for the computer simulation of biological systems interacting with fluids.* Symposia of the Sociey for Experimental Biology, 49:265, 1995.

[36] W. Proskurowski, O. Widlund. *On the numerical solution of Helmholtz equation by the capacitance matrix method.* Math. Comp., vol. 30,433–468. 1976.

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

[38] J. Smith. *The coupled equation approach to the numerical solution of the biharmonic equation by finite differences. I* SIAM J. Num. Anal., vol 5:323-339, 1968

[39] J. Smith. *The coupled equation approach to the numerical solution of the biharmonic equation by finite differences. II* SIAM J. Num. Anal., vol 7:104-112, 1970

[40] J. W. Stephenson. *Single cell discretizations of order two and four for biharmonic problems.* J. Comput. Phys., 55:65–80, 1984.

[41] P. N. Swarztrauber. *Fast Poisson solver.* In Gene H. Golub, editor, *Studies in Numerical Analysis*, volume 24, pages 319–370. The Mathematical Association of America, 1984.

[42] S. Timoshenko, J. Goodier. *Theory of Elasticity.* McGraw-Hill Co., NY. 1970.

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