

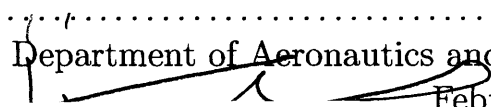
A Correction Function Method to Solve Incompressible Fluid Flows to High Accuracy with Immersed Geometries

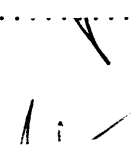
by
Alexandre Noll Marques

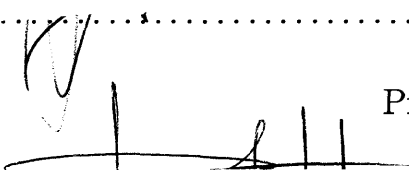
Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

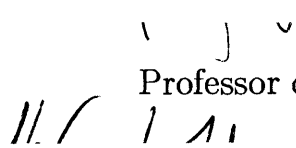
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2012

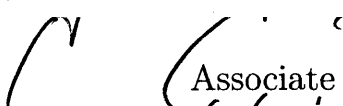
© Massachusetts Institute of Technology 2012. All rights reserved.

Author
Department of Aeronautics and Astronautics

February 27, 2012

Certified by

Rodolfo R. Rosales
Professor of Applied Mathematics
Thesis Supervisor

Certified by

Jean-Christophe Nave
Professor of Mathematics
Thesis Supervisor

Certified by

Jaime Peraire
Professor of Aeronautics and Astronautics
Thesis Committee

Certified by ...

Steven G. Johnson
Associate Professor of Applied Mathematics
Thesis Committee

Accepted by

Eytan H. Modiano
Chair, Department Committee on Graduate Theses

A Correction Function Method to Solve Incompressible Fluid Flows to High Accuracy with Immersed Geometries

by

Alexandre Noll Marques

Submitted to the Department of Aeronautics and Astronautics
on February 27, 2012, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Numerical simulations of incompressible viscous flows in realistic configurations are increasingly important in many scientific and engineering fields. In Aeronautics, for instance, relatively cheap numerical computations replace costly hours of wind tunnel investigations in the early design stages of new aircraft. However, standard methods to obtain numerical solutions over complex geometries require sophisticated meshing techniques and intensive human interaction. In contrast, “immersed methods” incorporate complex boundaries and/or interfaces into regular meshes (Cartesian meshes or simple triangulations). Hence, immersed methods simplify the task of mesh generation and are of great interest in the study of incompressible viscous flows.

The objective of this thesis is to advance current immersed methods by formulations that yield highly accurate discretizations without compromising computational efficiency. This is achieved by introducing a new type of immersed method, the *correction function method*. This new method is based on the concept of a correction function that provides smooth extensions of the solution across boundaries and/or interfaces, such that standard (accurate and efficient) discretizations of the governing equations remain valid everywhere in the computational domain. Furthermore, the key concept behind the correction function method is the introduction of the correction functions as solutions to partial differential equations, which are defined locally around the immersed boundaries and interfaces. Then, we can solve these equations to any desired order of accuracy, resulting in high accuracy methods.

Specifically, in this thesis the correction function method is implemented to 4th order of accuracy in the context of Poisson’s equation, the heat equation, and the nonlinear convection advection diffusion in 2D. Then, these techniques are combined

to solve the incompressible Navier-Stokes equations, which govern the dynamics of incompressible viscous flows.

Thesis Supervisor: Rodolfo R. Rosales
Title: Professor of Applied Mathematics

Thesis Supervisor: Jean-Christophe Nave
Title: Professor of Mathematics

Acknowledgments

There are many people that I need to thank for their support and guidance during this challenging journey. First I must thank my advisors, Profs. Ruben Rosales and Jean-Christophe Nave. It was my first time working closely with Mathematicians, and it was decisively one of the most interesting and rewarding experiences I ever had. It was Profs. Nave's vision and enthusiasm that persuaded me to get involved in this particular project and kept me on track to achieve so much. Prof. Rosales' uncanny capacity to identify and simplify the hardest problems was crucial in the development of this work. I am also thankful for their loyalty and commitment when I needed to overcome bureaucratic hurdles so I could focus on science for most of the time.

Furthermore, I thank all MIT professors with whom I had the pleasure to work. In particular, I thank the members of my thesis committee for their constructive criticism and recommendations: Profs. Jaime Peraire, Steven Johnson, Youssef Marzouk, and David Darmofal. I also thank the help I received from all the colleagues that I met in this time, specially the Applied Mathematics Fluids Lunch group, David Shirokoff, and Anas Alfaris. In addition, I thank the always prompt assistance from the Department of Aeronautics and Astronautics' staff, specially Jean Sofronas, Beth Marois and Marie Stuppard, and the people at the Department of Mathematics headquarters.

This work counted with the financial support by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES - Brazil) and the Fulbright Commission through grant BEX 2784/06-8. It was also partially funded by the National Science Foundation through grant DMS-0813648, and the NSERC Discovery program.

Finally, I must thank the unwavering and unconditional love and support I received from my family. I am specially grateful to my loving and caring wife, Nielle Marques, who gave me strength and purpose to move forward, and always made everything possible.

Contents

1	Introduction	15
1.1	Motivation	15
1.2	Literature Review: Immersed Methods	18
1.3	Contributions	23
1.4	Organization of the thesis	25
2	The Correction Function Method	27
2.1	Definition of the problem	28
2.2	The basic idea	29
2.3	The correction function and the equation defining it	31
2.4	A 4 th order accurate scheme in 2D	35
2.4.1	Overview	35
2.4.2	Standard Stencil	36
2.4.3	Definition of $\Omega_{\Gamma}^{(i,j)}$	38
2.4.4	Solution of the local PDE	42
2.4.5	Computational Cost	44
2.4.6	Interface Representation	45
2.4.7	Error analysis	46
2.4.8	Computation of gradients	46
2.5	Results	47
2.5.1	Example 1	48
2.5.2	Example 2	50
2.5.3	Example 3	51

2.5.4	Example 4	55
3	Extensions of the Correction Function Method	59
3.1	Boundary conditions on complex geometries	60
3.1.1	Overview	60
3.1.2	The correction function and the equation defining it	61
3.1.3	Definition of $\Omega_{\Gamma}^{(i,j)}$	63
3.1.4	Solution of the Local PDE	64
3.1.5	Neumann boundary condition	66
3.1.6	Results	67
3.2	Poisson's equation with discontinuous coefficient	69
3.2.1	Overview	69
3.2.2	Smooth extensions of the solution	71
3.2.3	Solution of the Local PDE	73
3.2.4	Results	76
4	Alternative Method to Solve the Poisson equation – using boundary integral equations	81
4.1	Solution procedure	82
4.1.1	Laplace equation	83
4.1.2	Including a non-homogeneous source	86
4.1.3	Poisson equation with Piece-wise constant coefficients	87
4.1.4	Summary	90
4.2	Results	93
4.2.1	Example 1. Imposing boundary conditions over arbitrarily shaped surfaces	94
4.2.2	Example 2. Poisson equation with piece-wise constant coefficients	96
5	Dynamic Problems	99
5.1	Heat equation	100
5.1.1	Overview	100

5.1.2	A compact discretization	101
5.1.3	The Correction Function Method	103
5.1.4	Results	105
5.2	Convection-Diffusion equation	106
5.2.1	Overview	106
5.2.2	A compact discretization	109
5.2.3	Results	112
6	Incompressible Navier-Stokes Equations	115
6.1	Formulation	116
6.2	Numerical Scheme	119
6.3	Results	123
6.3.1	Purely periodic boundaries	124
6.3.2	Immersed Boundary	125
6.3.3	Flow over a cylinder	129
7	Conclusion	137
7.1	Final Remarks	137
7.2	Future work	139
A	The 9-point stencil for the Poisson equation	141
B	Bicubic interpolation	143
C	Issues affecting the construction of $\Omega_{\Gamma}^{i,j}$	145
C.1	Naive Grid-Aligned Stencil-Centered Approach.	147
C.2	Compact Grid-Aligned Stencil-Centered Approach.	148
C.3	Free Stencil-Centered Approach	149
C.4	Node-Centered Approach	150
D	Ill-posed problem	153

List of Figures

1-1	Examples of body-fitted and immersed meshes for the domain between circles of radius 0.1 and 1.	16
2-1	Example of solution domain Ω . The solution is discontinuous across the interface Γ	29
2-2	Example in 1D of a solution with a jump discontinuity.	30
2-3	(a) The 9-point compact stencil next to the interface Γ . (b) The set $\Omega_{\Gamma}^{(i,j)}$ for this stencil.	37
2-4	Configuration where multiple $\Omega_{\Gamma}^{(i,j)}$ are needed in the same stencil. (a) Same interface crossing the stencil multiple times. (b) Distinct interfaces crossing the same stencil.	41
2-5	Example 1. (a) Solution domain embedded in a 33×33 Cartesian grid. (b) Solution obtained with a 193×193 grid.	49
2-6	Example 1. Convergence of the error in the solution and its gradient in the L_2 and L_{∞} norms.	49
2-7	Example 1. Convergence of the error in the solution and its gradient evaluated along the interface.	50
2-8	Example 2. Convergence of the error in the solution and its gradient in the L_2 and L_{∞} norms.	52
2-9	Example 2. Convergence of the error in the solution and its gradient in the L_2 and L_{∞} norms.	52
2-10	Example 2. Convergence of the error in the solution and its gradient evaluated along the interface.	53

2-11	Example 3. Convergence of the error in the solution and its gradient in the L_2 and L_∞ norms.	54
2-12	Example 3. Convergence of the error in the solution and its gradient in the L_2 and L_∞ norms.	55
2-13	Example 3. Convergence of the error in the solution and its gradient evaluated along the interface.	55
2-14	Example 4. Convergence of the error in the solution and its gradient in the L_2 and L_∞ norms.	57
2-15	Example 4. Convergence of the error in the solution and its gradient in the L_2 and L_∞ norms.	57
2-16	Example 4. Convergence of the error in the solution and its gradient evaluated along the interface.	58
3-1	Example of a solution domain with arbitrary shape.	60
3-2	Steps involved in forming $\Omega_\Gamma^{(i,j)}$	65
3-3	(a) Solution domain embedded in a 65×65 Cartesian grid. (b) Solution obtained with a 193×193 grid.	68
3-4	Convergence of the error in the L_2 and L_∞ norms.	68
3-5	Convergence of the error evaluated along the interface.	69
2-1	Example of solution domain with arbitrary shape.	70
3-6	Regions where the extended solutions are defined. This example assumes that node (i, j) lies in Ω^+	73
3-7	(a) Solution domain embedded in a 65×65 Cartesian grid. (b) Solution obtained with a 193×193 grid.	78
3-8	Convergence of the error in the L_2 and L_∞ norms.	79
3-9	Convergence of the error along the interface.	80
4-1	Rectangular domain \mathcal{B} that involves Ω	85
4-2	(a) Solution domain embedded in a 33×33 Cartesian Grid. (b) Solution obtained with a 193×193 grid.	95
4-3	Error convergence in L_2 and L_∞ norms.	96

4-4	(a) Solution domain embedded in a 33×33 Cartesian Grid. (b) Solution obtained with a 193×193 grid.	98
4-5	Error convergence in L_2 and L_∞ norms.	98
3-1	Example of solution domain with arbitrary shape.	100
5-1	(a) Solution domain embedded in a 65×65 Cartesian grid. (b) Convergence of the error in the L_2 and L_∞ norms.	106
5-2	Solution at different times.	106
5-3	Convergence of the error along the boundary.	107
5-4	Solution domain discretized with a 65×65 Cartesian grid. The internal boundary is immersed in the grid.	113
5-5	Plot of the u component of the solution at different times.	113
5-6	Plot of the v component of the solution at different different in time.	113
5-7	Convergence of the error in the L_∞ and L_2 norms.	114
5-8	Convergence of the error along the boundary.	114
6-1	Solution at $t = 10$ for $Re = 1$	125
6-2	L_∞ norm of the divergence of velocity for $Re = 1 \times 10^6$	126
6-3	Convergence of the error in the L_∞ norm.	126
6-4	Solution domain with the boundary immersed in a 96×96 Cartesian grid.	127
6-5	Solution at $t = 1$ for $Re = 1$	127
6-6	(a) Variation of the L_∞ norm of the divergence of the velocity over time. (b) Convergence of the error in the L_∞ norm.	128
6-7	Convergence of the error along the boundary.	128
6-8	One period of the infinite array of cylinders. The boundary is immersed in a 268×128 Cartesian grid.	130
6-9	Solution at $t = 6$ for $Re = 1$. Speed denotes $s = \sqrt{u^2 + v^2}$	131
6-10	Streamlines of the flow around the cylinder at $t = 6$, with $Re = 1$	131
6-11	(a) Variation of the L_∞ norm of the divergence of velocity over time. (b) Nondimensional forces acting over the cylinder: drag and lift.	132

6-12	Solution at $t = 6$ for $Re = 30$. Speed denotes $s = \sqrt{u^2 + v^2}$	132
6-13	Streamlines of the flow around the cylinder with $Re = 10$	133
6-14	(a) Variation of the L_∞ norm of the divergence of velocity over time.	
	(b) Nondimensional forces acting over the cylinder: drag and lift. . .	133
6-15	Solution at $t = 30$ for $Re = 20$. Speed denotes $s = \sqrt{u^2 + v^2}$	134
6-16	Solution at $t = 60$ for $Re = 20$. Speed denotes $s = \sqrt{u^2 + v^2}$	134
6-17	Streamlines of the flow around the cylinder with $Re = 20$	135
6-18	Streamlines showing vortex shedding behind cylinder for $t > 50$	136
6-19	(a) Variation of the L_∞ norm of the divergence of velocity over time.	
	(b) Nondimensional forces acting over the cylinder: drag and lift. . .	136
3-1	$\Omega_\Gamma^{i,j}$ as defined by the naive grid-aligned stencil-centered approach. . .	147
3-2	$\Omega_\Gamma^{i,j}$ as defined by the compact grid-aligned stencil-centered approach.	148
3-3	$\Omega_\Gamma^{i,j}$ as defined by the free stencil-centered approach.	150
3-4	$\Omega_\Gamma^{i,j}$ as defined by the node-centered approach.	151

Chapter 1

Introduction

1.1 Motivation

Many applications in science and engineering involve the dynamics of incompressible viscous flows. From mixtures of immiscible fluids [1, 2], to motion of microorganisms [3], to flight of insects [4, 5], to the flow of blood in the heart [6]. In Aeronautics, the airflow over low-speed aircrafts can often be idealized as incompressible. Such is the case with many of the unmanned air vehicles (UAVs) in production today (*e.g.* USAF's MQ-1B Predator cruises at Mach 0.11 [7]). Moreover, even though many of the characteristics of the airflow over these vehicles can be estimated with simplified inviscid models, there are important features that can only be accurately determined by including viscous effects, such as drag force and flow separation.

The advances in computational power and memory over the last decades have made it possible to study increasingly realistic and complex flow configurations with numerical methods. However, an accurate representation of the complex geometries involved in many applications demands sophisticated meshing techniques, such as multi-block meshes [8, 9], octree decomposition [10], automatic triangulations [11], and hybrid methods [12, 13] (for a thorough review, see [14]). Although these techniques constitute very powerful tools for mesh generation, the task of creating adequate meshes for complex geometries still requires a considerable amount of human interaction. This issue is even more complicated if the boundaries or the interfaces

are moving in unsteady simulations. Since regular solvers require a body-fitted mesh (see figure 1-1(a)), one needs an algorithm to adapt the mesh to the motion of the boundary at every time step. A common practice is to deform the original mesh to account for the motion of the boundary [15–17]. In some situations, it is possible to construct a smooth mapping between the deformed mesh and a reference mesh. In this case, one can use arbitrary Lagrangian-Eulerian (ALE) methods to account for the mesh deformation in an accurate fashion [16]. However, in general situations the process of mesh deformation can be costly and adds errors to the solution. Moreover, in extreme cases (*e.g.* large deformations, interfaces merging/splitting) complete remeshing may be necessary.

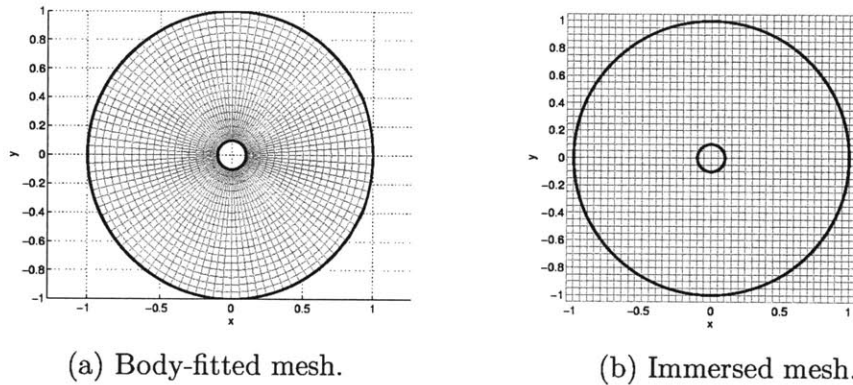


Figure 1-1: Examples of body-fitted and immersed meshes for the domain between circles of radius 0.1 and 1.

To avoid the complications related to creating quality body-fitted meshes and adapting the mesh to moving boundaries, a family of “immersed methods” for incompressible viscous flows has arisen over the last four decades. In these methods, one does not need to fit the mesh to the boundary or interface (see figure 1-1(b)): boundaries and/or interfaces are immersed into regular meshes (Cartesian meshes or simple triangulations), and the methods automatically adapt the discretization to the boundary or interface conditions. A thorough literature review of this class of methods is presented in the next section.

Despite recent advances, many of these methods are at most second order accurate

– with a few exceptions. Moreover, in general, either immersed methods do not offer clear extensions to higher orders of accuracy, or the extensions are excessively convoluted and inefficient. The objective of this thesis is to contribute to the theory of immersed methods by introducing a general formulation that allows the systematic creation of numerical schemes with high order accuracy. Hence, this thesis is focused on a completely new concept, rather than simply on extending the already existing immersed methods.

The method presented in this thesis is based on the construction of smooth extensions of the solution across boundaries and interfaces. These extensions are then used to define *correction functions*, which can be used to “complete” standard discretizations of the equations. Hence the name *correction function method* (CFM). The key concept behind the CFM is characterizing the correction functions as solutions to partial differential equations defined locally in the vicinity of the boundaries and interfaces. The idea of extending the solution is not new, but defining these extensions as the solution to PDEs is an entirely original concept. Furthermore, in principle one can devise schemes to solve these local PDEs to any desired order of accuracy. Therefore, this concept is the main feature of the CFM that allows us to obtain high order of accuracy.

The incompressible viscous flows of interest in this thesis are described by the incompressible Navier-Stokes equations (INSE). A common practice to solve these equations is reformulate the problem in terms of a nonlinear convection-diffusion equation for the velocity distribution, and a Poisson equation for the pressure distribution [18–26]. The core contributions of this thesis are

- (i) A new and highly accurate CFM to solve the Poisson equation with immersed boundaries.
- (ii) A new and highly accurate CFM to solve the nonlinear convection-diffusion equation with immersed boundaries.
- (iii) The integration of these methods to solve the incompressible Navier-Stokes equations to high order of accuracy with immersed boundaries.

1.2 Literature Review: Immersed Methods

Immersed methods were originally conceived to solve problems with interfaces (or infinitely thin membranes) dividing multiphase flows, and later extended to deal with complex boundaries in an immersed fashion. Hence, much of the development in this area happened around interface problems. Two important aspects must be considered when using immersed methods: (a) the representation (and tracking) of the interface, and (b) the discretization of the governing equations in the vicinity of the interface. The latter ultimately characterizes the different methods, but the development of both aspects is intertwined.

There are two classes of methods used to represent the interface: explicit and implicit. Explicit methods are based on introducing fictitious particles that represent the location of the interface. The advantage of explicit representations is that one can follow the interface by simply tracking the individual particles that move according to simple kinematics. Probably the first explicit method was the marker and cell (MAC) method introduced by Harlow and Welch [27–29]. This method is aimed at flows with a free surface. Basically, fictitious markers are introduced within the fluid and the position of the outermost markers characterize the location of the free surface. Another explicit approach is to place particles along the interface itself [30–32]. The location of the neighboring particles is used to produce local interpolations (*e.g.* splines), which are then applied to compute geometric information — such as curvature and normal directions. Although this approach can be quite accurate, it requires special treatment when the interface undergoes either large deformations or topological changes — such as mergers or splits. This issue can be particularly challenging in 3D [32].

On the other hand, in an implicit representation the location of the interface is extracted from some function that is defined everywhere in the regular computational grid. An early implicit representation was obtained by extending the MAC method into the volume-of-fluid (VOF) method [2, 33, 34]. In the VOF, instead of using markers, one registers the partial volume occupied by the fluid in each cell of the grid.

This approach is more efficient than MAC since it considerably decreases the number of degrees of freedom necessary to track the free surface, but it does not result in better accuracy. A more popular approach to represent the interface implicitly is the level set (LS) method introduced by Osher and Sethian [35–39]. In the LS method, the interface is given by the zero level of a function defined everywhere in the domain. The basis of the LS method, however, is the effective and efficient algorithm that is used to advance the interface by advecting the LS function with the local fluid velocity. The success of the LS method gave rise to a vast literature that covers a wide range of applications other than flow problems [40, 41]. In particular, in this thesis the gradient-augmented level set (GA-LS) method [42] was adopted. With this extension of the LS method, one can obtain highly accurate representations of the interface, and other geometric information, with the additional advantage that this method uses only local grid information.

As mentioned before, a common practice to solve the incompressible Navier-Stokes equations (INSE) is to reformulate the problem in terms of a nonlinear convection-diffusion equation and a Poisson equation. When the solution is known to be smooth, it is easy to obtain highly accurate discretizations to these equations on a regular grid. Furthermore, these discretizations usually yield symmetric and banded linear systems, which can be inverted efficiently [43]. On the other hand, when singularities occur (*e.g.* discontinuities) across internal interfaces, some of the regular discretization stencils will straddle the interface, which renders the whole procedure invalid.

Several strategies have been proposed to tackle this issue. Peskin [30] introduced the immersed boundary method (IBM) [6, 30, 44–47], in which the discontinuities are re-interpreted as additional (singular) force terms concentrated on the interface. These singular terms are then “regularized” and appropriately spread out over the regular grid — in a “thin” band enclosing the interface. This approach is very appealing since the discretization of the flow equations is not affected, only the right-hand-side (RHS). The result, however, is a first order scheme that smears discontinuities. Goldstein [48] presents an extension of the IBM where linear control theory is used to compute the singular forces needed to impose the no-slip condition on a boundary

of the domain. A more direct method to determine the singular forces for this problem was later introduced by Fadlun *et al.* [49]. A recent review of the IBM and its applications was presented in [47].

In order to avoid the smearing of the interface information, LeVeque and Li [50] developed the immersed interface method (IIM) [50–54], which is a methodology to modify the discretization stencils, taking into consideration the discontinuities at their actual locations. The IIM guarantees second order accuracy and sharp discontinuities, but at the cost of added discretization complexity and loss of symmetry. The IIM was also extended to treat no-slip boundary conditions by adding singular forces along the boundary in the work of Le *et al.* [55]. This extension preserves the second order accuracy of the IIM. Recently, Zhong [56] introduced a new version of the IIM where the modified discretization stencils are obtained by matching polynomials on both sides of the interface. The result is a high order method based on wide dimension-by-dimension stencils.

The new method advanced in this thesis builds on some ideas introduced by the ghost fluid method (GFM) [57–62]. The GFM is based on defining both actual and “ghost” fluid variables at every grid node that lies in a narrow band enclosing the interface. The ghost variables work as extensions of the actual variables across the interface — the solution on each side of the interface is assumed to have a smooth extension into the other side. After the ghost values are computed, one can apply standard discretizations everywhere in the domain. Moreover, in most GFM versions the ghost values are written as the actual values, plus corrections that are independent on the underlying solution to the problem. Hence, the corrections can be pre-computed, and moved into the source term for the equation. In this fashion, the GFM yields the same linear system as the one produced by the problem without an interface, except for changes in the RHS only. Thus, this linear system can be inverted just as efficiently as in problems with smooth solutions. The key difficulty in the GFM is the calculation of the correction terms, since the overall accuracy of the scheme depends heavily on the quality of the assigned ghost values. In [57–61] the authors develop first order accurate approaches to deal with discontinuities.

It is relevant to note other methods developed to solve boundary problems in a immersed fashion. Mayo [31] introduced a method to solve the Laplace equation in which one first solves a boundary integral equation to obtain “jump conditions” over the boundary. After these jump conditions are known, one can solve the Laplace equation using finite differences. The jump conditions are used to create corrections to the RHS of the discretized equation, similarly to the GFM (even though Mayo’s method preceded the GFM). Moreover, since the solution to the integral equation can also yield jump conditions in derivatives of the solution, this method can be used to create corrections to high order accuracy. Mayo [31,63] shows second and fourth order results. Extensions of Mayo’s method to the Poisson equation are presented in [63,64]. In §4 similar ideas are used to solve Poisson equations in general configurations.

Johansen and Colella [65] introduced a second order accurate finite volume discretization of the incompressible Navier-Stokes equations (INSE). This method is based on building second order approximations to the fluxes on the edges of partial cells (cells cut by the boundary) based on quadratic interpolations in the direction normal to the boundary. The result is a second order accurate method that yields non-symmetric linear systems. Jomaa and Macaskill [66] use a similar concept to obtain a finite-difference discretization of the Poisson equation with immersed boundary. In their work, however, Jomaa and Macaskill [66] show that stencil modifications based on dimension-by-dimension linear interpolations suffice to yield second order accuracy, resulting in a symmetric discretization. In addition, Linnick and Fasel [67] were able to build a fourth order accurate method to solve the Navier-Stokes equations in stream function-vorticity formulation with complex boundaries in a immersed fashion. This method introduces modifications to a fourth order accurate compact finite-difference discretization of the equations using concepts similar to the IIM.

Similar methods were also developed for situations where Dirichlet-type conditions are applied to an internal interface. Such is the case with Stefan problems that describe the solidification/liquefaction of pure substances, including unstable solidification that leads to dendritic crystal growth [37,68]. Udaykumar *et al.* [32] introduced a method to solve the full INSE to second order of accuracy for problems with phase

transformation. In this method, the INSE are discretized using regular second-order finite-difference stencils, except for the vicinity of the interface, where a first order and non-symmetric discretization is devised to enforce the appropriate interface conditions. Gibou *et al.* [69] introduced a formulation similar to the GFM to discretize the Poisson equation to solve the Stefan problem. However, in this work, the final discretization is second order accurate and symmetric. Later, Gibou and Fedkiw [70] developed a fourth order accurate version of this discretization, at the cost of giving up symmetry. More recently, the same problem has also been solved, to second order of accuracy, in non-graded adaptive Cartesian grids by Chen *et al.* [71].

The finite element community has also made significant progress in incorporating the IIM and similar techniques to solve the Poisson equation using immersed grids. Finite element formulations have the advantage that they usually produce symmetric discretizations of self-adjoint operators, and that they are (in principle) easily extendible to high orders of accuracy and to higher dimensions. However, they also require (at least for the current approaches for immersed problems) “special” elements and/or integrations over partial elements. Hence, even though most of these methods yield symmetric discretizations, most applications are still restricted to second order accuracy in 2D.

One popular finite element method to solve elliptic problems with immersed interfaces is the penalty method [72–74], where interface conditions are added with penalization parameters to the weak formulation of the problem. Moës *et al.* [75] introduced the extended finite element method (X-FEM) in the context of modelling the growth of cracks. In essence, the X-FEM uses an enriched basis to represent the solution adjacent to cracks, incorporating appropriate enrichment functions that model displacement discontinuities introduced by the cracks. Other more general immersed finite-element formulations include the finite-element embedded interface method by Dolbow and Harari [76], the virtual node method by Bedrossian *et al.* [77], and extensions of the IIM [54, 78–80], and the exact subgrid interface correction method by Huh and Sethian [81].

In the context of the finite volume method, the cut-cell method [82–84] became a

standard approach in fluid flow applications with immersed boundaries. This method was later adapted to finite elements [85], and recently incorporate in a DG discretization of the compressible Navier-Stokes equations [86].

1.3 Contributions

In this thesis a new method is presented – the *correction function method* – to solve incompressible viscous flows to high order of accuracy. This new method is of the “immersed” kind, in which the boundaries and/or interfaces are immersed into underlying regular meshes.

One key step needed to solve the incompressible Navier-Stokes equations (INSE) is the solution of the Poisson equation. Hence, one of the core contributions of this thesis is the development of a correction function method that can be used to solve the Poisson equation to high order of accuracy using immersed grids. Furthermore, within the context of the Poisson equation there is a number of configurations of interest to fluid flow applications. In this thesis the Poisson equation is solved in the context of (i) discontinuous solutions across internal interface, (ii) discontinuous coefficients, and (iii) immersed boundaries. The method used to solve situation (i) is called the “original” correction function method, since it serves as the foundation for the correction function method used in all other applications discussed in this thesis. This original version of the CFM was published in the *Journal of Computational Physics* [87].

Other contribution of the thesis is the extension of the CFM to dynamic problems related to fluid flow phenomena. In particular, the thesis includes the solution of the heat equation and of the nonlinear convection-diffusion equation with immersed boundaries. This contribution involves the creation of a concept of extended solutions and correction functions that stretches over time and is valid for dynamic equations. In addition, this extension is crucial to fluid flow applications because we need to solve a nonlinear convection-diffusion equation to obtain the velocity distribution in incompressible viscous flows.

The final contribution of the thesis is putting the techniques mentioned above together to create a numerical solver for the INSE. This numerical solver is capable of obtaining solutions to high order of accuracy with boundaries that are immersed into regular Cartesian grids. This step also involves writing the INSE in a formulation that is suitable for an accurate CFM discretization.

In summary, the contributions of this thesis can be separated into three categories:

1. **Poisson equation:** development of a new family of immersed schemes – the *correction function method* (CFM). These schemes yield highly accurate discretizations that can be efficiently inverted for the following problems.
 - Constant coefficients Poisson equation with jump discontinuities across an internal interface.
 - Discontinuous coefficient Poisson equation with jump discontinuities across an internal interface.
 - Poisson equation with “immersed boundaries.”
2. **Dynamic problems:** extension of the CFM to dynamic equations of interest in fluid flow applications. In particular, this thesis includes
 - the heat equation.
 - the nonlinear convection-diffusion equation.
3. **Incompressible flow solver:** use of the CFM to solve the incompressible Navier-Stokes equations. This step involves splitting these equations into a nonlinear convection-diffusion equation for the velocity distribution, and a Poisson equation for the pressure distribution. Then, the techniques developed in items 1 and 2 are used to create a numerical scheme that can solve incompressible viscous flows to high order of accuracy with immersed boundaries.

1.4 Organization of the thesis

The remainder of this thesis is organized as follows. In §2 the “original” version of the correction function method is introduced. This version is designed to solve the constant coefficients Poisson equation, with jump discontinuities (with some restrictions) imposed across an interface internal to the solution domain. In this chapter, the basic concepts and machinery that are behind all other versions of the CFM are discussed in detail.

Then in §3 the original CFM is extended to solve (i) the Poisson equation with an immersed boundary, and (ii) the discontinuous coefficient Poisson equation. The biggest difficulty in these cases is that the PDE that defines the correction function is coupled to the underlying solution of the Poisson equation. The approach used to handle this coupling between the equations is discussed in §3. Next, §4 presents an alternative approach to solve these same problems. In this alternative approach, one first solves a boundary integral equation to obtain new “jump conditions.” After this step, a general Poisson equation can be rewritten as an equivalent constant coefficients Poisson equation with discontinuous solution. Then this equivalent problem can be solved with the original CFM.

Chapter §5 is dedicated to the solution of the heat equation and of the nonlinear convection-diffusion equation, both with immersed boundaries. In this chapter the concept of a correction function is extended to the context of dynamic problems. Then, in §6 the techniques discussed in §3 and §5 are combined to solve the incompressible Navier-Stokes equations. Finally, in §7 are the concluding remarks, including a summary the most important features of the methods described in the thesis, and a list of open areas for possible future work.

Chapter 2

The Correction Function Method

In this chapter the original version of the *correction function method* (CFM) [87] is introduced. This original version was designed to solve the constant coefficients Poisson equation in situations where the solution is discontinuous across some arbitrary interface. A formal definition of this problem is presented in §2.1. Extensions of the CFM to solve the Poisson equation under more general circumstances are discussed in in §3 and §4.

The basic idea behind the correction function method is to create smooth extensions of the solution from both sides of the discontinuity into the other side. Once these extensions are known, one can use a standard discretization of the Poisson equation everywhere in the solution domain. This idea, and its relationship to the ghost fluid method, are explained in detail in §2.2. Furthermore, these extensions can be used to define a *correction function*, which is characterized as the solution to a partial differential equation, as shown in §2.3. Next, in §2.4 this concept is applied to build a 4th order accurate scheme in 2D. In addition, the domain of definition of the correction function deserves special attention, so appendix C is dedicated to explaining some details that are left out of §2.4. Finally, in § 2.5 the robustness and accuracy of the 2D scheme are demonstrated by applying it to numerical examples.

2.1 Definition of the problem

The original version of the CFM was designed to solve the constant coefficients Poisson equation in a domain Ω in which the solution is discontinuous across a co-dimension 1 interface Γ . This interface divides the domain into the subdomains Ω^+ and Ω^- , as illustrated in figure 2-1. The notation $(\cdot)^+$ and $(\cdot)^-$ is used to denote values in each of the subdomains. Furthermore, the discontinuities across Γ are given in terms of two functions defined on the interface: $a = a(\vec{x})$ for the jump in the function values, and $b = b(\vec{x})$ for the jump in the normal derivatives. Finally, Dirichlet boundary conditions are imposed on the outer boundary $\partial\Omega$. Thus the problem to be solved is

$$\Delta u(\vec{x}) = f(\vec{x}) \quad \text{for } \vec{x} \in \Omega, \quad (2.1a)$$

$$[u] = a(\vec{x}) \quad \text{for } \vec{x} \in \Gamma, \quad (2.1b)$$

$$[u_n] = b(\vec{x}) \quad \text{for } \vec{x} \in \Gamma, \quad (2.1c)$$

$$u(\vec{x}) = g(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega, \quad (2.1d)$$

where

$$[\cdot] = (\cdot)^+ - (\cdot)^- \quad (2.2a)$$

denotes jumps across an internal interface. Throughout this chapter, $\vec{x} = (x_1, x_2, \dots) \in \mathbb{R}^\nu$ is the spatial vector (where $\nu = 2$, or $\nu = 3$), and Δ is the Laplacian operator defined by

$$\Delta = \sum_{i=1}^{\nu} \frac{\partial^2}{\partial x_i^2}. \quad (2.3)$$

Furthermore,

$$u_n = \hat{n} \cdot \vec{\nabla} u = \hat{n} \cdot (u_{x_1}, u_{x_2}, \dots) \quad (2.4)$$

denotes the derivative of u in the direction of \hat{n} , the unit vector normal to the interface Γ pointing towards Ω^+ (see figure 2-1).

Moreover, note that this version of the CFM is focused on the discretization of the problem in the vicinity of the interface only. Thus, the method is compatible with

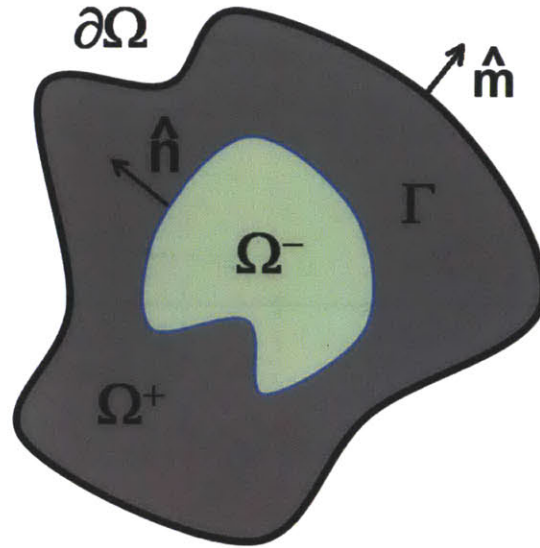


Figure 2-1: Example of solution domain Ω . The solution is discontinuous across the interface Γ .

any set of boundary conditions on $\partial\Omega$, not just Dirichlet, as long as these boundary conditions are properly enforced. In the examples included in this chapter, $\partial\Omega$ is rectangular. Extensions of the CFM to arbitrarily shaped boundaries are discussed in §3 and §4.

2.2 The basic idea

The correction function method is based on the concept of a *correction function*. This concept is a generalization of the ideas of the ghost fluid method (GFM). In essence, one starts with a standard finite-differences discretization of the Laplace operator (*e.g.* 5-point or 9-point stencil). Whenever the discretization stencil straddles the interface Γ , the corrections are used on the right-hand-side (RHS) of the discretized equation to incorporate the jump conditions. As a consequence, the linear system that results from the finite-differences discretization is not altered, only the RHS. Thus, this linear system can be inverted as efficiently as in the case of a solution without discontinuities.

The following example illustrates the key concept in the GFM. Consider the one

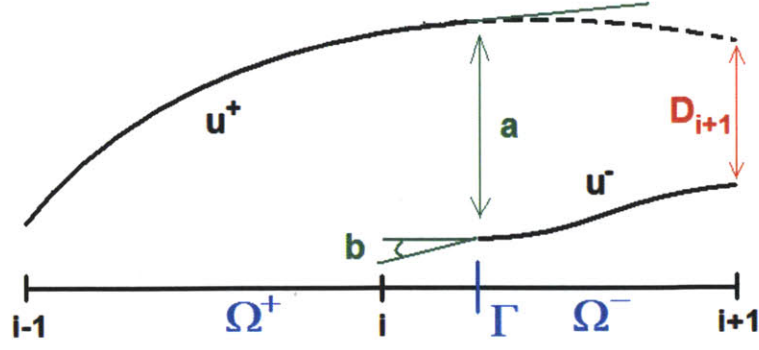


Figure 2-2: Example in 1D of a solution with a jump discontinuity.

dimensional Poisson equation: $u_{xx}(x) = f(x)$, for $x_L < x < x_R$. Then, assume that u is discontinuous at x_Γ , with $\Omega^+ = \{x : x_L < x < x_\Gamma\}$, and $\Omega^- = \{x : x_\Gamma < x < x_R\}$. The standard 2nd order centered differences discretization of the equation is

$$u_{xx_i}^+ \approx \frac{u_{i-1}^+ - 2u_i^+ + u_{i+1}^+}{h^2}, \quad (2.5)$$

where $h = x_{i+1} - x_i$ is the grid spacing – see figure 2-2. However, in the situation depicted in figure 2-2, $x_i < x_\Gamma < x_{i+1}$. Thus, at x_{i+1} only u_{i+1}^- is defined, not u_{i+1}^+ . The idea is to estimate a correction $D_{i+1} = u_{i+1}^+ - u_{i+1}^-$, such that (2.5) becomes

$$u_{xx_i}^+ \approx \frac{u_{i-1}^+ - 2u_i^+ + \overbrace{(u_{i+1}^- + D_{i+1})}^{u_{i+1}^+}}{h^2}, \quad (2.6)$$

Note that, if the correction D_{i+1} is independent on the solution u , then it can be moved to the RHS of the equation, and absorbed into f . That is

$$\frac{u_{i-1}^+ - 2u_i^+ + u_{i+1}^-}{h^2} = f_i - \frac{D_{i+1}}{h^2}. \quad (2.7)$$

With this procedure, the problem with discontinuities can be solved with the same discretization used to solve the continuous problem. Hence, the resulting linear system can be inverted using the same well established and efficient techniques available for

the continuous Poisson equation [43].

Remark 2.1. *The final accuracy of the discretization depends on the quality of D . Liu, Fedkiw, and Kang [59] estimate D using a dimension-by-dimension linear extrapolation of the interface jump conditions – i.e. the functions a and b in (2.1). The result is a first order approximation for D . The CFM is based on generalizing the idea of the correction term to that of a correction function, which can be characterized as the solution to a partial differential equation. Then, high accuracy representations of D follow from solving this equation to high order accuracy, without the complications introduced by dimension-by-dimension Taylor expansions. ♣*

Remark 2.2. *An additional advantage of the correction function approach is that D can be evaluated at any point near the interface Γ . Hence, the CFM can be used with any finite differences discretization of the Poisson equation, without regard to the particulars of the stencil (as would be the case with any approach based on Taylor expansions). ♣*

2.3 The correction function and the equation defining it

As mentioned above, the idea of the CFM is to generalize the concept of correction terms defined at certain grid nodes to that of a *correction function*, and then to find a partial differential equation (PDE) – with appropriate boundary conditions – that uniquely characterizes the *correction function*. Then, at least in principle, one can design algorithms to solve this PDE to obtain the *correction function* to any desired order of accuracy.

Consider a small region Ω_Γ that encloses the interface Γ , defined as the set of all the points within some distance \mathcal{R} of Γ . The value \mathcal{R} is of the order of the grid size h . As explained below, \mathcal{R} must be as small as possible. On the other hand, Ω_Γ has to include all the points where the CFM requires corrections to be computed,

which means that \mathcal{R} depends on the discretization stencil being used¹. In addition, algorithmic considerations (to be seen later) may force \mathcal{R} to be slightly larger than what is needed to include the discretization stencil.

Next, assume that both u^+ and u^- can be extrapolated, so that they are valid everywhere within Ω_Γ and satisfy the following Poisson equations.

$$\Delta u^+(\vec{x}) = f^+(\vec{x}) \quad \text{for } \vec{x} \in \Omega_\Gamma, \quad (2.8a)$$

$$\Delta u^-(\vec{x}) = f^-(\vec{x}) \quad \text{for } \vec{x} \in \Omega_\Gamma, \quad (2.8b)$$

where f^+ and f^- are smooth enough extensions of the source term f to Ω_Γ (see remark 2.3 below). In particular, notice that (2.8) allows for the possibility of a source term that is discontinuous across Γ .

The *correction function* is then defined by $D(\vec{x}) = u^+(\vec{x}) - u^-(\vec{x})$. The PDE that characterize D is obtained by (i) taking the difference between the (2.8a) and (2.8b), and (ii) using the jump conditions (2.1b) and (2.1c). Thus,

$$\Delta D(\vec{x}) = f^+(\vec{x}) - f^-(\vec{x}) = f_D(\vec{x}) \quad \text{for } \vec{x} \in \Omega_\Gamma, \quad (2.9a)$$

$$D(\vec{x}) = a(\vec{x}) \quad \text{for } \vec{x} \in \Gamma, \quad (2.9b)$$

$$D_n(\vec{x}) = b(\vec{x}) \quad \text{for } \vec{x} \in \Gamma. \quad (2.9c)$$

This PDE defines the *correction function* as the solution to a set of equations, with some provisos – see remark 2.4 below. Note that:

1. If the source term is continuous across Ω , then $f_D = 0$.
2. Equation (2.9c) imposes the true jump condition in the normal direction, whereas some versions of the GFM rely on a dimension-by-dimension approximation of this condition [59].

Remark 2.3. *The smoothness requirement on f^+ and f^- is tied up to the desired accuracy for D . For example, in general one can only estimate D to 4th order accuracy*

¹In particular, the 9-point stencil is used in this thesis, so \mathcal{R} cannot be smaller than $\sqrt{2}h$.

if D is at least C^4 . Hence, in this case, $f_D = f^+ - f^-$ must be C^2 . ♣

Remark 2.4. Equation (2.9) is an elliptic Cauchy problem. In general, such problems are ill-posed. However, (2.9) is well posed in the special context of a numerical approximation where

- (a) There is a frequency cut-off in (i) the data $a = a(\vec{x})$ and $b = b(\vec{x})$, and (ii) the description of the curve Γ .
- (b) The solution is needed only a small distance away from the interface Γ , where this distance vanishes simultaneously with the inverse of the cut-off frequency mentioned in point(a).

Because of these conditions, the arbitrarily large growth rate for arbitrarily small perturbations, which is responsible for the ill-posedness of the Cauchy problem, does not occur.

The reason the arbitrarily large growth does not occur is as follows. Consider a perturbation to the solution to Poisson equation along some straight line. Assume a sinusoidal perturbation with wave number $0 < k < \infty$. Then the perturbation grows as $e^{2\pi kd}$, where d is the distance from the line. However, by construction, in the present case conditions (a) and (b) guarantee that kd is bounded. ♣

Remark 2.5. A number characterizing how well posed the discretized version of (2.9) is can be defined as:

$$\alpha = \text{largest growth rate possible,}$$

where growth is defined relative to the size of a perturbation to the solution on the interface. This number is determined by \mathcal{R} (the “radius” of Ω_Γ), as the following calculation shows. First, there is no loss of generality in assuming that the interface is flat, provided that the numerical grid is fine enough to resolve Γ . In this case, consider an orthogonal coordinate system \vec{y} on Γ , and let d be the signed distance to Γ (say, $d < 0$ in Ω^-). Expanding the perturbations in Fourier modes along the

interface, the typical mode has the form

$$\varphi_{\vec{k}} = e^{2\pi i \vec{k} \cdot \vec{y} \pm 2\pi k d},$$

where \vec{k} is the Fourier wave vector, and $k = |\vec{k}|$. As noted in item (a) of remark 2.4, there is a limit to the shortest wave-length that can be represented on a grid with mesh size $0 < h \ll 1$, corresponding to $k = k_{\max} = 1/(2h)$. Hence, an estimate for the maximum growth rate is

$$\alpha \approx e^{\pi \mathcal{R}/h}.$$

Moreover, as noted in item (b) of remark 2.4, the size of \mathcal{R} is proportional to h . Hence, the maximum growth rate is always bounded. ♣

Remark 2.6. Clearly, α is intimately related to the condition number for the discretized problem — see §2.4. In fact, at leading order, the two numbers should be (roughly) proportional to each other — with a proportionality constant that depends on the details of the discretization. For the discretization used described in §2.4, $\sqrt{2}h \leq \mathcal{R} \leq 2\sqrt{2}h$, which leads to the rough estimate $85 < \alpha < 7,200$. On the other hand, the observed condition numbers vary between 5,000 and 10,000. Hence, the actual condition numbers are only slightly higher than α for the ranges of grid sizes used here (the asymptotic limit $h \rightarrow 0$ was not explored). ♣

Remark 2.7. Equation (2.9) depends only on the known inputs of the problem: f^+ , f^- , a , and b . Consequently, D does not depend on the solution u . Hence, after solving (2.9) for D , one can use any standard finite difference discretization of the Poisson equation. Whenever the stencil straddles the interface, D is evaluated where the correction is needed, and these values are transferred to the RHS. ♣

Remark 2.8. When developing an algorithm for a linear Cauchy problem, such as (2.9), the two key requirements are consistency and stability. In particular, when the solution depends on the “initial conditions” globally, stability (typically) imposes stringent constraints on the “time” step for any local (explicit) scheme. This would seem to suggest that, in order to solve (2.9), a “global” (involving the whole domain

Ω_Γ) method will be needed. This, however, is not true: because the solution of (2.9) is needed for one “time” step only – i.e. within an $\mathcal{O}(h)$ distance from Γ , stability is not relevant. Hence, consistency is enough, and a fully local scheme is possible. In the algorithm described in §2.4 it was observed that, for (local) quadrangular patches, the Cauchy problem leads to a well behaved algorithm when the length of the interface contained in each patch is of the same order as the diagonal length of the patch. This result is in line with the calculation in remark 2.5: we want to keep the “wavelength” (along Γ) of the perturbations introduced by the discretization as long as possible. In particular, this should then minimize the condition number for the local problems – see remark 2.6. ♣

2.4 A 4th order accurate scheme in 2D

2.4.1 Overview

In this section the general ideas presented in §2.3 are used to develop a specific example of a 4th order accurate scheme in 2D. The key points of this scheme are

- (a) The Poisson equation is discretized using a compact 9-point stencil – see appendix A. Compactness is important since it is directly related to the size of \mathcal{R} , which has an impact on the problem’s conditioning – see remarks 2.4 to 2.6.
- (b) The interface Γ is represented using the gradient-augmented level set method – see [42]. This method guarantees a local 4th order representation of the interface, as required to keep the overall accuracy of the scheme.
- (c) The domain Ω_Γ is sub-divided into small rectangular regions dubbed $\Omega_\Gamma^{(i,j)}$. Each of these regions is associated with a point in the grid at which the standard discretization of the Poisson equation involves a stencil that straddles the interface Γ . Furthermore, $\Omega_\Gamma^{(i,j)}$ encloses a portion of Γ , and all the nodes where D is needed to complete the discretization of the Poisson equation at the (i, j) -th stencil.
- (d) Within each $\Omega_\Gamma^{(i,j)}$, the correction function D is approximated using a bicubic

interpolation. This approximation guarantees local 4th order accuracy with only 12 interpolation parameters ².

- (e) In each $\Omega_{\Gamma}^{(i,j)}$, the PDE (2.9) is solved in a least squares sense. Namely: First we define an appropriate positive quadratic integral quantity J , equation (2.14), for which the solution is a minimum (actually, zero). Next, we substitute the bicubic approximation for the solution into J , and the integrals are discretized using Gaussian quadrature. Finally, we find the bicubic parameters by minimizing the discretized J .

Remark 2.9. *Solving the PDE in a least squares sense is crucial, since an algorithm is needed that can deal with the myriad ways in which the interface Γ can be placed relative to the fixed rectangular grid used to discretize the Poisson equation. This approach provides a scheme that (i) is robust with respect to the details of the interface geometry, (ii) has a formulation that is (essentially) dimension independent – there are no fundamental changes from 2D to 3D, and (iii) has a clear theoretical underpinning that allows extensions to higher orders, or to other discretizations of the Poisson equation.* ♣

2.4.2 Standard Stencil

In this thesis we use the standard 4th order accurate 9-point discretization of the Poisson equation (see appendix A):

$$L^5 u_{i,j} + \frac{1}{12}(h_x^2 + h_y^2)\hat{\partial}_{xx}\hat{\partial}_{yy}u_{i,j} = f_{i,j} + \frac{1}{12}(h_x^2(f_{xx})_{i,j} + h_y^2(f_{yy})_{i,j}), \quad (2.10)$$

where L^5 is the 2nd order 5-point discretization of Laplace’s operator – see equation (A.1).

In the absence of discontinuities, expression (2.10) provides a compact 4th order accurate representation of the Poisson equation. On the other hand, in the vicinity

²The standard bicubic interpolation requires 16 interpolation parameters. However, in the thesis we use the version introduced in [42], which needs only 12 parameters. For more details, see appendix B.

of the interface Γ we need to compute correction terms to complete the discretization, as described in detail next. To understand how the correction terms affect the discretization, consider the situation depicted in figure 2-3(a). In this case, the node (i, j) lies in Ω^+ while the nodes $(i+1, j)$, $(i+1, j+1)$, and $(i, j+1)$ are in Ω^- . Hence, to be able to use the discretization (2.10), we need to compute $D_{i+1,j}$, $D_{i+1,j+1}$, and $D_{i,j+1}$.

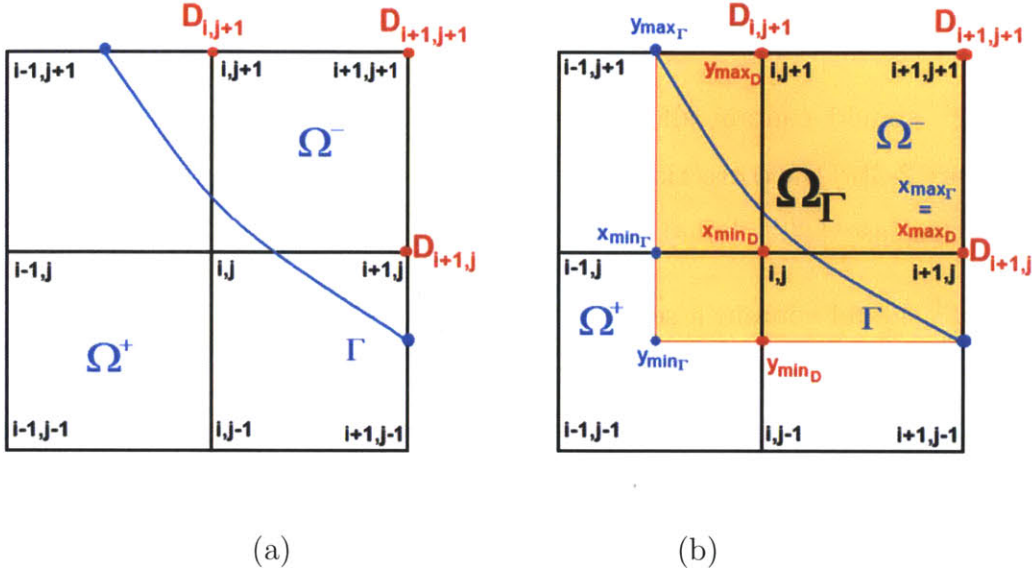


Figure 2-3: (a) The 9-point compact stencil next to the interface Γ . (b) The set $\Omega_\Gamma^{(i,j)}$ for this stencil.

After having solved for D where necessary (see §2.4.3 and §2.4.4), the correction terms modify the RHS on (2.10) as follows.

$$L^5 u_{i,j} + \frac{1}{12} (h_x^2 + h_y^2) \hat{\partial}_{xx} \hat{\partial}_{yy} u_{i,j} = f_{i,j} + \frac{1}{12} (h_x^2 (f_{xx})_{i,j} + h_y^2 (f_{yy})_{i,j}) + C_{i,j}, \quad (2.11)$$

Here the $C_{i,j}$ are the CFM correction terms needed to complete the stencil across the discontinuity at Γ . In the particular case illustrated in figure 2-3(a),

$$C_{i,j} = \left[\frac{1}{6} \frac{(h_x^2 + h_y^2)}{(h_x h_y)^2} - \frac{1}{h_y^2} \right] D_{i+1,j} + \left[\frac{1}{6} \frac{(h_x^2 + h_y^2)}{(h_x h_y)^2} - \frac{1}{h_x^2} \right] D_{i,j+1} - \frac{1}{12} \frac{(h_x^2 + h_y^2)}{(h_x h_y)^2} D_{i+1,j+1}. \quad (2.12)$$

Similar formulas apply for the other possible arrangements of the Poisson equation stencil relative to the interface Γ .

2.4.3 Definition of $\Omega_\Gamma^{(i,j)}$

There is some freedom on how to define $\Omega_\Gamma^{(i,j)}$. The basic requirements are

- (i) $\Omega_\Gamma^{(i,j)}$ should be small, since the problem's condition number increases exponentially with the distance from Γ – see remarks 2.5 and 2.6.
- (ii) $\Omega_\Gamma^{(i,j)}$ should contain all the nodes where D is needed. For the example, in figure 2-3(a) the correction terms $D_{i+1,j+1}$, $D_{i+1,j}$, and $D_{i,j+1}$ are needed. Hence, in this case, $\Omega_\Gamma^{(i,j)}$ should include the nodes $(i+1, j+1)$, $(i+1, j)$, and $(i, j+1)$.
- (iii) $\Omega_\Gamma^{(i,j)}$ should contain a segment of Γ , with a length that is as large as possible — *i.e.* comparable to the length of the diagonal of $\Omega_\Gamma^{(i,j)}$. This follows from the calculation in remark 2.5, which indicates that the wavelength of the perturbations (along Γ) introduced by the discretization should be as long as possible. This should then minimize the condition number for the local problem – see remark 2.6.

In addition, since (2.9) is solved in a least squares sense, integrations over $\Omega_\Gamma^{(i,j)}$ are required. Thus, it is useful to keep $\Omega_\Gamma^{(i,j)}$ as simple as possible. For this reason, we add the extra requirements listed below.

- (iv) $\Omega_\Gamma^{(i,j)}$ should be a rectangle.
- (v) The edges of $\Omega_\Gamma^{(i,j)}$ should be parallel to the grid lines.

In principle, items (iv) and (v) could be traded for improvements in other areas – for example, for better condition numbers for the local problems, or for additional flexibility in dealing with complex geometries. However, for simplicity we enforce (iv) and (v). A discussion of various aspects regarding the definition of $\Omega_\Gamma^{(i,j)}$ can be found in appendix C. For instance, requirement (v) is convenient only when an implicit representation of the interface is used.

With the points above in mind, $\Omega_{\Gamma}^{(i,j)}$ is defined as the smallest rectangle that satisfies the requirements in (ii)-(v); (i) follows automatically. Hence $\Omega_{\Gamma}^{(i,j)}$ can be constructed using the following three easy steps.

1. Find the coordinates $(x_{\min_{\Gamma}}, x_{\max_{\Gamma}})$ and $(y_{\min_{\Gamma}}, y_{\max_{\Gamma}})$ of the smallest rectangle that completely encloses the section of the interface Γ contained by the region covered by the 9-point stencil.
2. Find the coordinates (x_{\min_D}, x_{\max_D}) and (y_{\min_D}, y_{\max_D}) of the smallest rectangle that completely encloses all the nodes at which D needs to be known.
3. Then $\Omega_{\Gamma}^{(i,j)}$ is the smallest rectangle that encloses the two previous rectangles. Its edges are given by

$$x_{\min} = \min(x_{\min_{\Gamma}}, x_{\min_D}), \quad (2.13a)$$

$$x_{\max} = \max(x_{\max_{\Gamma}}, x_{\max_D}), \quad (2.13b)$$

$$y_{\min} = \min(y_{\min_{\Gamma}}, y_{\min_D}), \quad (2.13c)$$

$$y_{\max} = \max(y_{\max_{\Gamma}}, y_{\max_D}). \quad (2.13d)$$

Figure 2-3(b) shows an example of $\Omega_{\Gamma}^{(i,j)}$ defined using these specifications.

Remark 2.10. Notice that a distinct domain $\Omega_{\Gamma}^{(i,j)}$ is defined for each stencil that straddles the interface. When doing so, domains overlap. For example, the domain $\Omega_{\Gamma}^{(i,j)}$ shown in figure 2-3(b) is used to determine $C_{i,j}$. It should be clear that $\Omega_{\Gamma}^{(i-1,j+1)}$ (used to determine $C_{i-1,j+1}$), and $\Omega_{\Gamma}^{(i+1,j-1)}$ (used to determine $C_{i+1,j-1}$), each will overlap with $\Omega_{\Gamma}^{(i,j)}$.

The consequence of these overlaps is that there are multiple values for D at the same node – one for each domain used to solve the local Cauchy problem. However, because we solve for D – within each $\Omega_{\Gamma}^{(i,j)}$ – to 4th order accuracy, any differences that arise from this multiple definition of D lie within the order of accuracy of the scheme. Since it is convenient to keep the computations local, the values of D resulting from the domain $\Omega_{\Gamma}^{(i,j)}$ are used to evaluate the correction term $C_{i,j}$. ♣

Remark 2.11. While rare, cases where a single interface crosses the same stencil multiple times can occur. An example is presented in §2.5.2. A simple approach to deal with situations like this involves two steps: (i) associate each node where the correction function is needed to a particular piece of interface crossing the stencil (say, the closest one), and (ii) define one $\Omega_{\Gamma}^{(i,j)}$ for each of the individual pieces of interface crossing the stencil.

For example, figure 2-4(a) depicts a situation where the stencil is crossed by two pieces of the same interface (Γ_1 and Γ_2), with D needed at the nodes $(i + 1, j + 1)$, $(i + 1, j)$, $(i, j + 1)$, and $(i - 1, j - 1)$. Then, first associate:

- $(i + 1, j + 1)$, $(i + 1, j)$, and $(i, j + 1)$ to Γ_1 .
- $(i - 1, j - 1)$ to Γ_2 .

Second, define

1. $\Omega_{\Gamma_1}^{(i,j)}$ is the smallest rectangle, parallel to the grid lines, that includes Γ_1 and the nodes $(i + 1, j + 1)$, $(i + 1, j)$, and $(i, j + 1)$.
2. $\Omega_{\Gamma_2}^{(i,j)}$ is the smallest rectangle, parallel to the grid lines, that includes Γ_2 and the node $(i - 1, j - 1)$.

After the multiple $\Omega_{\Gamma}^{(i,j)}$ are defined within a given stencil, the local Cauchy problem is solved within each $\Omega_{\Gamma}^{(i,j)}$ separately. For example, in the case shown in figure 2-4(a), the solution for D inside $\Omega_{\Gamma_1}^{(i,j)}$ is completely independent of the solution for D inside $\Omega_{\Gamma_2}^{(i,j)}$. The decoupling between multiple crossings renders the CFM flexible and robust enough to handle complex geometries without any special algorithmic considerations. ♣

Remark 2.12. When multiple distinct interfaces are involved, a single stencil can be crossed by different interfaces – e.g. see §2.5.3 and § 2.5.4. This situation is similar to the one described in remark 2.11, but with an additional complication: there may occur distinct domain regions that are not separated by an interface, but rather by a third (or more) regions between them. An example is shown in figure 2-4(b), where

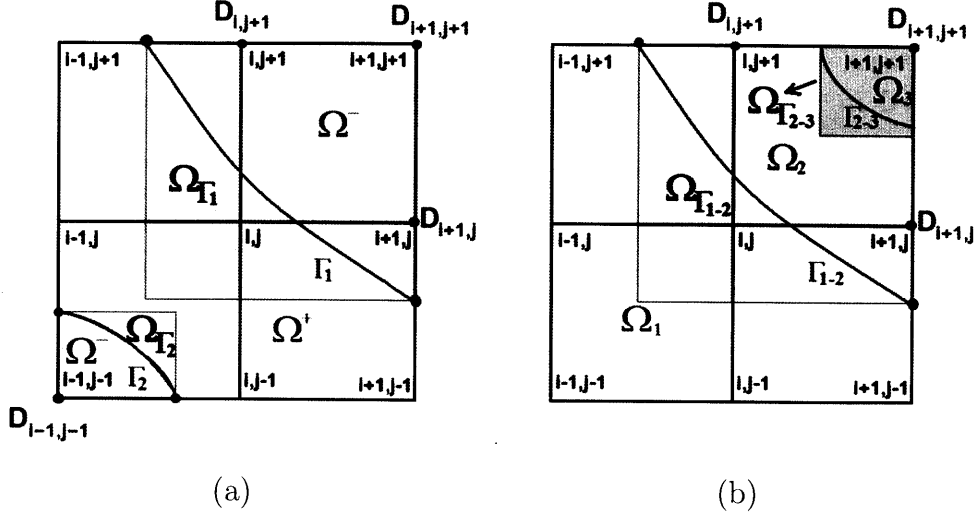


Figure 2-4: Configuration where multiple $\Omega_{\Gamma}^{(i,j)}$ are needed in the same stencil. (a) Same interface crossing the stencil multiple times. (b) Distinct interfaces crossing the same stencil.

Γ_{1-2} and Γ_{2-3} are not part of the same interface. Here Γ_{1-2} is the interface between Ω_1 and Ω_2 , while Γ_{2-3} is the interface between Ω_2 and Ω_3 . There is no interface Γ_{1-3} separating Ω_1 from Ω_3 , hence no jump conditions between these regions are provided. Nonetheless, $D_{1-3} = (u_3 - u_1)$ is needed at $(i + 1, j + 1)$.

Situations such as these can be easily handled by noticing that we can distinguish between primary (e.g. D_{1-2} and D_{2-3}) and secondary correction functions, which can be written in terms of the primary functions (e.g. $D_{1-3} = D_{1-2} + D_{2-3}$) and need not be computed directly. Hence, we can proceed exactly as in remark 2.11, except that we have to make sure that the intersections of the regions where the primary correction functions are computed include the nodes where the secondary correction functions are needed. For example, in the particular case in figure 2-4(b), we define

1. $\Omega_{\Gamma_{1-2}}^{(i,j)}$ is the smallest rectangle, parallel to the grid lines, that includes Γ_{1-2} and the nodes $(i + 1, j + 1)$, $(i + 1, j)$, and $(i, j + 1)$.
2. $\Omega_{\Gamma_{2-3}}^{(i,j)}$ is the smallest rectangle, parallel to the grid lines, that includes Γ_{2-3} and the node $(i + 1, j + 1)$.



2.4.4 Solution of the local PDE

As mentioned in §2.4.1, it is important to solve the PDE that defines the correction function in a least squares sense. By doing so, we are able to build a scheme that (i) does not excite undesirable high frequencies that affect the conditioning of the problem, and (ii) that is flexible enough to deal with the myriad of geometrical configurations that result from the crossing of an arbitrary interface with the regular grid. Specifically, the local PDE are solved by minimizing the functional

$$\begin{aligned}
 J = & \frac{\ell_c^{(i,j)^4}}{2V(\Omega_\Gamma^{(i,j)})} \int_{\Omega_\Gamma^{(i,j)}} \{\Delta D(\vec{x}) - f_D(\vec{x})\}^2 dV \\
 & + c_P \frac{1}{2L(\Gamma)} \int_{\Gamma \cap \Omega_\Gamma^{(i,j)}} \{D(\vec{x}) - a(\vec{x})\}^2 dS \\
 & + c_P \frac{\ell_c^{(i,j)^2}}{2L(\Gamma)} \int_{\Gamma \cap \Omega_\Gamma^{(i,j)}} \{D_n(\vec{x}) - b(\vec{x})\}^2 dS,
 \end{aligned} \tag{2.14}$$

where $V(\Omega_\Gamma^{(i,j)})$ is the “volume” of $\Omega_\Gamma^{(i,j)}$, and $L(\Gamma)$ is the “area” of Γ . Furthermore, $c_P > 0$ is the penalization coefficient used to enforce the interface conditions, and $\ell_c^{(i,j)} > 0$ is a characteristic length associated with $\Omega_\Gamma^{(i,j)}$ – *e.g.* the shortest side length. Clearly J is a quadratic functional whose minimum (zero) occurs at the solution to (2.9).

Hence, computing D in the domain $\Omega_\Gamma^{(i,j)}$, involves the steps described below.

1. Choose a set of basis functions to represent D within $\Omega_\Gamma^{(i,j)}$: $D(\vec{x}) = \sum_{\ell=1}^n d_\ell \phi_\ell(\vec{x})$.
2. Replace this representation of D into the functional J .
3. Approximate the integrals in (2.14) using numerical quadrature.
4. Solve for the weights d_ℓ that minimize J ($n \times n$ self-adjoint linear system).

Since we use a 4th order accurate discretization of the Poisson equation, we need to obtain D with 4th order errors (or better) to keep the overall accuracy of the scheme – see § 2.4.7. Hence, here D is represented using cubic Hermite splines (bicubic in-

terpolants in 2D), which guarantees 4th order accuracy – see [42]³. Note also that, even though the scheme developed here is restricted to 2D, this representation can be easily extended to any number of dimensions. Moreover, the integrals are approximated using Gaussian quadratures – the results presented here were computed with six quadrature points for the 1D line integrals, and 36 points for the 2D area integrals. The resulting discrete problem is then minimized. Because the bicubic representation of D involves 12 basis polynomials, the minimization problem produces a 12×12 self-adjoint linear system.

Remark 2.13. *The option of enforcing the interface conditions using Lagrange multipliers was also explored. While this second approach yields good results, experience showed that the penalization method is more robust.* ♣

Remark 2.14. *The scaling using $\ell_c^{i,j}$ in (2.14) is so that all the three terms in the definition of J behave in the same fashion as the size of $\Omega_\Gamma^{i,j}$ changes with (i, j) , or when the computational grid is refined⁴. This follows because we expect that*

$$\begin{aligned}\Delta D - f &= \mathcal{O}(\ell_c^2), \\ D - a &= \mathcal{O}(\ell_c^4), \\ D_n - b &= \mathcal{O}(\ell_c^3).\end{aligned}$$

Hence each of the three terms in (2.14) should be $\mathcal{O}(\ell_c^8)$. ♣

Remark 2.15. *Once all the terms in (2.14) are guaranteed to scale the same way with the size of $\Omega_\Gamma^{i,j}$, the penalization coefficient c_P should be selected so that the three terms have (roughly) the same size for the numerical solution (they will, of course, not vanish).*

In principle, c_P could be determined from knowledge of the fourth order derivatives of the solution, which control the error in the numerical solution. This approach does not appear to be practical. A simpler method is based on the observation that c_P should

³The basis functions corresponding to the bicubic interpolation can be found in appendix B.

⁴The scaling also follows from dimensional consistency.

not depend on the grid size (at least to leading order, and we do not need better than this). Hence it can be determined empirically from a low resolution calculation. In the examples shown in §2.5 $c_P \approx 50$ produced good results. ♣

Remark 2.16. A more general version of J would involve different penalization coefficients for the two line integrals, as well as the possibility of these coefficients having a dependence on the position of $\Omega_\Gamma^{i,j}$. These modifications could be useful in cases where the solution to Poisson equation has large variations – e.g. a very irregular interface Γ , or a complicated forcing f . Nonetheless, (2.14) worked for all problem considered here. ♣

2.4.5 Computational Cost

We can now infer something about the cost of scheme proposed here. To start with, denote the number of nodes in the x and y directions by

$$N_x = \frac{1}{h_x} + 1, \quad N_y = \frac{1}{h_y} + 1, \quad (2.15)$$

assuming a 1 by 1 computational square. Hence, the total number of degrees of freedom is $M = N_x N_y$. Furthermore, the number of nodes adjacent to the interface is $\mathcal{O}(M^{1/2})$, since the interface is a 1D entity.

The standard discretization of the Poisson equation results in a $M \times M$ linear system. Furthermore, the present method produces changes only on the RHS of the equations. Thus, the basic cost of inverting the linear system is unchanged, and it varies from $\mathcal{O}(M)$ to $\mathcal{O}(M^2)$ operations, depending on the solution method.

Let us now consider the computational cost added by the modifications to the RHS. As presented above, for each node adjacent to the interface, we must

- construct $\Omega_\Gamma^{i,j}$.
- compute the integrals that define the local 12×12 linear system.
- invert this 12×12 self-adjoint linear system.

Note that the cost associated with these tasks is constant: it does not vary from node to node, and it does not change with the size of the mesh. Consequently, the resulting additional cost is a constant times the number of nodes adjacent to the interface. Hence it scales as $M^{1/2}$. Because of the (relatively large) coefficient of proportionality, for small M this additional cost can be comparable to the cost of inverting the $M \times M$ linear system associated with the Poisson equation. Nevertheless, this extra cost becomes less significant as M increases.

2.4.6 Interface Representation

As far as the CFM is concerned, the framework needed to solve the local Cauchy problems is entirely described above. However, there is an important issue that deserves attention: the representation of the interface. This question is independent of the CFM. Many approaches are possible, and the optimal choice is geometry dependent. The discussion below is meant to shed some light on this issue, and motivate the solution adopted here.

In the present work, it is assumed that the interface is not known exactly – since this is what frequently happens. The only exceptions are examples §2.5.3 and §2.5.4, which involve two distinct (circular) interfaces touching at a point. In the generic setting, in addition to a proper representation of the interfaces, one needs to be able to identify the distinct interfaces, regions in between, contact points, as well as distinguish between a single interface crossing the same stencil multiple times and multiple distinct interfaces crossing one stencil. While the CFM algorithm is capable of dealing with these situations once they have been identified (*e.g.* see remarks 2.11 and 2.12), the development of an algorithm with the capability to detect such generic geometries is beyond the scope of this thesis, and a (hard) problem in interface representation. For these reasons, in the examples in §2.5.3 and §2.5.4 we use an exact representation of the interface.

To guarantee the accuracy of the solution for D , the interface conditions must be applied with the appropriate accuracy — see §2.4.7. Since these conditions are imposed on the interface Γ , the location of Γ must be known with the same order of

accuracy desired for D . In the particular case of the 4th order implementation of the CFM algorithm, we need to represent the interface to 4th order accuracy. For this reason, the gradient-augmented level set (GA-LS) method [42] was adopted here. This method allows a simple and completely local 4th order accurate representation of the interface, using Hermite cubics defined everywhere in the domain. The approach also allows the computation of normal vectors in a straightforward and accurate fashion.

Note that the GA-LS method is not the only option for an implicit 4th order representation of the interface. For example, a regular level set method [35], combined with a high-order interpolation scheme, could be used as well. However the GA-LS approach was adopted because of the algorithmic coherence that results from representing both the level set and the correction functions using the same bicubic polynomial base.

2.4.7 Error analysis

A naive reading of the discretized system (2.11) suggests that, in order to obtain a 4th order accurate solution u , one needs to compute the CFM correction terms $C_{i,j}$ to 4th order of accuracy. Thus, from (2.12), it would follow that we need to know the correction function D to 6th order accuracy! This is, however, *not correct*, as explained below.

Since we need to compute the correction function D only at grid points an $\mathcal{O}(h)$ of distance away from Γ , it should be clear that errors in the $D_{i,j}$ are equivalent to errors in the jump conditions a and b . But errors in a and b produce errors of the same order in u - see equations (2.1) and (2.2). Hence, if we desire a 4th accurate solution u , we need to compute the correction terms $D_{i,j}$ to 4th order of accuracy only. This argument is confirmed by the convergence plots shown in §2.5.

2.4.8 Computation of gradients

Some applications require not only the solution to Poisson equation, but also its gradient. Hence, in §2.5 plots of the convergence of the errors in the gradients of the

solutions are also shown. A key question is then: how are these gradients computed?

To compute the gradients near the interface, the correction function can be used to extend the solution across the interface, so that a standard stencil can be used. However, this approach only works if the gradient operator is discretized using the same nodes that are part of the 9-point stencil. If so, we can use the same correction functions computed while solving the Poisson equation. Hence, the gradient operator is discretized with a procedure similar to the one used to obtain the 9-point stencil (see appendix A). Specifically, the following 4th order accurate discretization is used.

$$\partial_x u_{i,j} = \hat{\partial}_x u_{i,j} + \frac{h_x^2}{6} \left[\hat{\partial}_{xx} \hat{\partial}_y u_{i,j} - (f_x)_{i,j} \right], \quad (2.16)$$

$$\partial_y u_{i,j} = \hat{\partial}_y u_{i,j} + \frac{h_y^2}{6} \left[\hat{\partial}_{yy} \hat{\partial}_x u_{i,j} - (f_y)_{i,j} \right], \quad (2.17)$$

where

$$\hat{\partial}_x u_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2h_x}, \quad (2.18)$$

$$\hat{\partial}_y u_{i,j} = \frac{u_{i,j+1} - u_{i,j-1}}{2h_y}, \quad (2.19)$$

and $\hat{\partial}_{xx}$ and $\hat{\partial}_{yy}$ are defined by (A.2) and (A.3), respectively. The terms $(f_x)_{i,j}$ and $(f_y)_{i,j}$ may be given analytically (if known), or computed using appropriate second order accurate discretizations.

This discretization is 4th order accurate. However, since the error in the correction function is (generally) not smooth, the resulting gradient will be less than 4th order accurate (worse case scenario is 3rd order accurate) next to the interface.

2.5 Results

This section shows four examples of computations in 2D using the scheme introduced in §2.4. In the first two examples the discontinuities occur along one single interface, whereas the last two examples involve two distinct interfaces touching at one single point. As discussed in §2.4.6, these last two examples involve the difficult problem of

identifying the pieces of interface cutting each stencil by each particular curve. For this reason, in examples 3 and 4 an exact representation of the interfaces is used. On the other hand, in examples 1 and 2 the interface is represented implicitly using the GA-LS method.

2.5.1 Example 1

This example involves the Poisson problem associated with the exact solution

$$\begin{aligned} u^+(x, y) &= \sin(\pi x) \sin(\pi y), \\ u^-(x, y) &= \sin(\pi x) \{\sin(\pi y) - \exp(\pi y)\}. \end{aligned}$$

The solution domain is the square $[0, 1] \times [0, 1]$, and the interface is defined by the zero contour of the level set function

$$\phi(x, y) = r^2(x, y) - r_0^2,$$

where

$$r(x, y) = \sqrt{(x - x_0)^2 + (y - y_0)^2}. \quad (2.20)$$

Here, $x_0 = 0.5$, $y_0 = 0.5$, and $r_0 = 0.1$. Figure 2-5(a) shows the interface immersed in a Cartesian grid. The subdomain Ω^- is the region contained inside the circular interface, whereas Ω^+ is the region exterior to this interface.

Figure 2-5(b) shows the numerical solution obtained with a fine grid (193×193 nodes). As we can observe, the discontinuity is captured very sharply, and it causes no oscillations in the solution. In addition, the convergence of the error in the solution and its gradient are plotted in figure 2-6. Both the L_2 and L_∞ norms are shown. As expected, the error in the solution converges to 4th order in both norms as the grid is refined. Moreover, the error in the gradient converges to 3rd order in the L_∞ norm and to 4th order in the L_2 norm, which is a reflection of the fact that the error in the solution is not smooth only in a narrow region close to the interface.

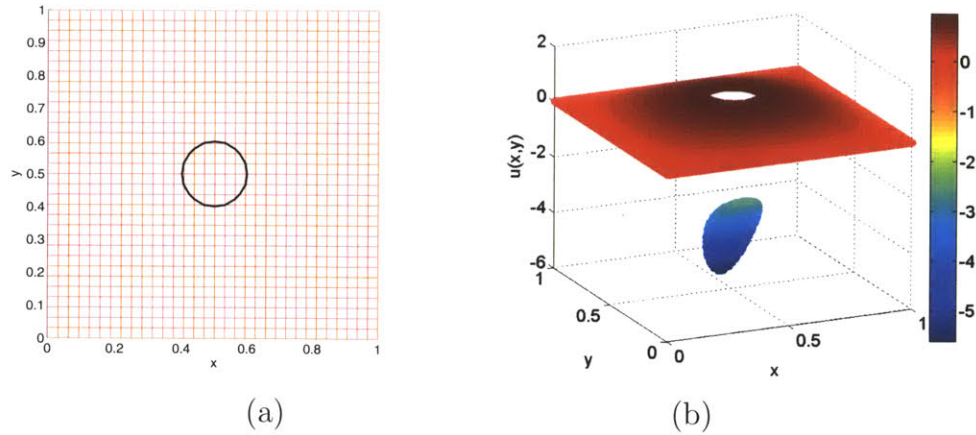


Figure 2-5: Example 1. (a) Solution domain embedded in a 33×33 Cartesian grid. (b) Solution obtained with a 193×193 grid.

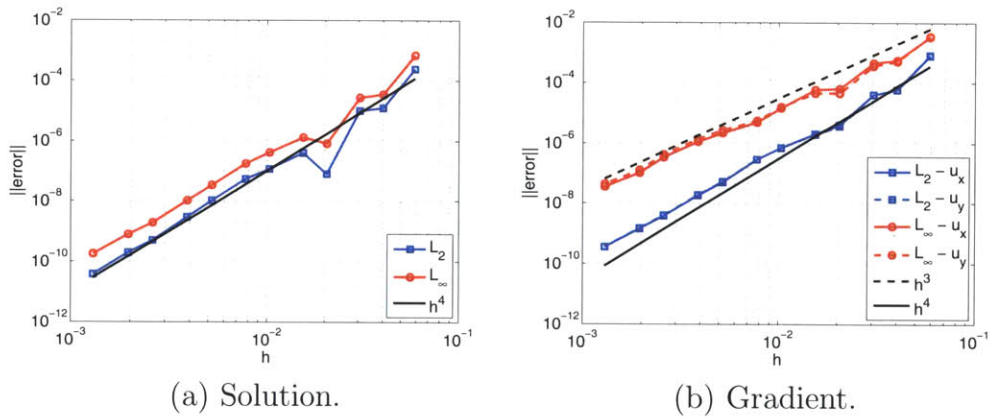


Figure 2-6: Example 1. Convergence of the error in the solution and its gradient in the L_2 and L_∞ norms.

Once the solution and its gradient are known on the computational grid, we can evaluate the solution anywhere in the domain to 4th order accuracy using the bicubic interpolation. For points close to the interface, the correction function can be used to “correct” the interpolation parameters on the grid nodes lying on the opposite side. In many applications, the solution evaluated on the interface is particularly important. Hence, to demonstrate the quality of the solution over the interface, figure 2-7 shows the convergence of the error along the interface. Namely, for each patch $\Omega_\Gamma^{(i,j)}$, the solution and its gradient are computed on the quadrature nodes used to approximate

the integral over the interface in (2.14) (either u^+ or u^- , since the jump conditions are known). The errors in the solution and the gradient are then measured, and the maximum errors over all quadrature nodes and all patches are plotted in figure 2-7 (a rough estimate of the continuous L_∞ norm). As expected, the error in the solution converges to 4th order, while the error in the gradient converges to 3rd order.

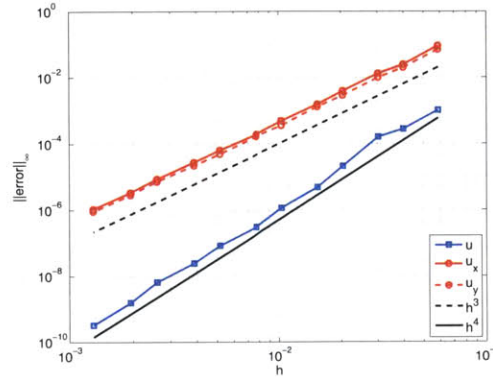


Figure 2-7: Example 1. Convergence of the error in the solution and its gradient evaluated along the interface.

2.5.2 Example 2

This example involves the Poisson problem associated with the exact solution

$$\begin{aligned} u^+(x, y) &= 0, \\ u^-(x, y) &= \exp(x) \cos(y). \end{aligned}$$

The solution domain is the $[0, 1] \times [0, 1]$ square, and the interface is defined by zero contour of the following level set function.

$$\begin{aligned} \phi(x, y) &= r^2(x, y) - r^2(\theta(x, y)), \\ r(\theta) &= r_0 + \epsilon \sin(5\theta), \\ \theta(x, y) &= \arctan\left(\frac{y - y_0}{x - x_0}\right), \end{aligned}$$

where $r(x, y)$ is defined by (2.20). Moreover, $x_0 = 0.5$, $y_0 = 0.5$, $r = 0.25$, and $\epsilon = 0.05$. Figure 2-8(a) shows the interface immersed in a Cartesian grid. The subdomain Ω^- is the region contained inside the interface, whereas Ω^+ is the region exterior to this interface.

Figure 2-8(b) shows the numerical solution obtained with a fine grid (193×193 nodes). Once again, the overall quality of the solution is very satisfactory. Figure 2-9 shows the convergence of the error of the solution and its gradient in the L_2 and L_∞ norms. As we can observe, the solution converges to 4th order, while the gradient converges to 3rd order in the L_∞ norm and close to 4th order in the L_2 norm. However, unlike what happens in example 1, small wiggles are observed in the error convergence plots. This behavior can be explained by the construction of the sets $\Omega_\Gamma^{i,j}$ – see §2.4. The approach used to construct $\Omega_\Gamma^{i,j}$ is highly dependent on the way in which the grid points are placed relative to the interface. Thus, as the grid is refined, the arrangement of the $\Omega_\Gamma^{i,j}$ can vary quite a lot – specially for a “complicated” interface such as the one in this used example. What these variations mean is that, while one can guarantee that the correction function D is obtained with 4th order precision, the proportionality coefficient is not constant – it may vary a little from grid to grid. This variation is responsible for the small oscillations observed in the convergence plot. Nevertheless, despite these oscillations, the overall convergence is clearly 4th order.

The convergence of the error along the interface is shown in figure 2-10. As expected, the error in the solution converges to 4th order, while the error in its gradient converges to 3rd order.

2.5.3 Example 3

In this example two interfaces are so close together as to touch at one single point. Consequently, this example involves the issue discussed in remark 2.12: the possibility of more than one interface crossing the stencil at the same time.

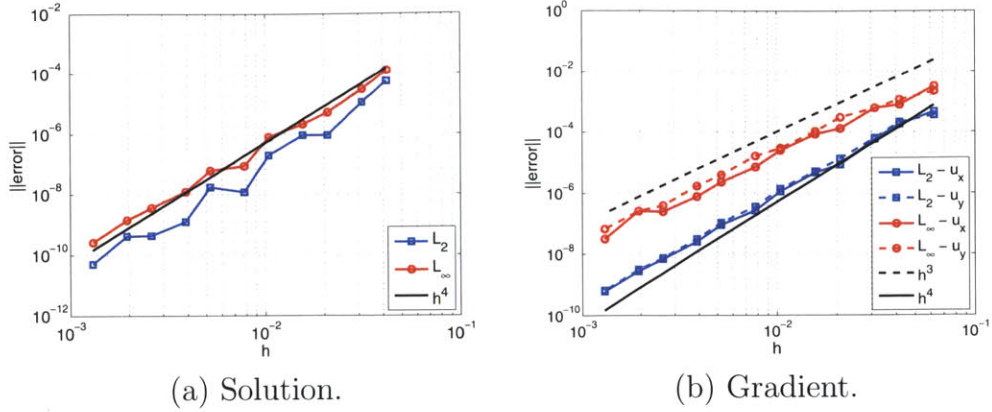


Figure 2-12: Example 3. Convergence of the error in the solution and its gradient in the L_2 and L_∞ norms.

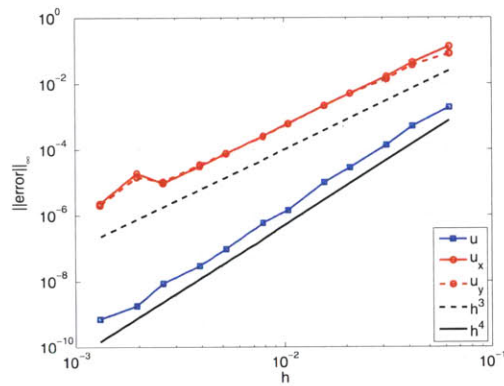


Figure 2-13: Example 3. Convergence of the error in the solution and its gradient evaluated along the interface.

2.5.4 Example 4

This example complements example 3, the sole difference being that here the small circle is placed inside the big circle. The Poisson problem to be solved in this case is

- Region 3: inside of the small circle.

Figure 2-11(b) shows the numerical solution with a fine grid (193×193 nodes). In this example the big circle is centered within the square integration domain and the small circle is external to it, with a common point of tangency. This setting guarantees that, as the grid is refined, a wide variety of configurations involving two distinct interfaces crossing the same stencil occurs in a neighborhood of the contact point. Figure 2-12 shows the convergence of the error in the L_2 and L_∞ norms. Once again we observe 4th order convergence (with small superimposed oscillations) for the solution. In addition, the gradient converges to 3rd order in the L_∞ norm and close to 4th order in the L_2 norm. This example shows that the CFM is robust even in situations where distinct interfaces can get arbitrarily close (tangent at a point).

In this example the solution and its gradient are also evaluate along the interface. Figure 2-13 shows the convergence of the errors. As expected, the error in the solution converges to 4th order, while the error in its gradient converges to 3rd order.

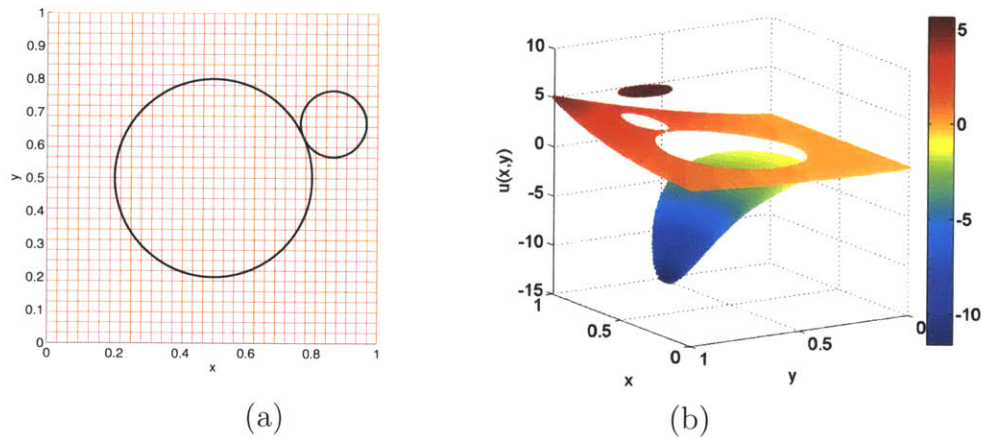


Figure 2-11: Example 3. Convergence of the error in the solution and its gradient in the L_2 and L_∞ norms.

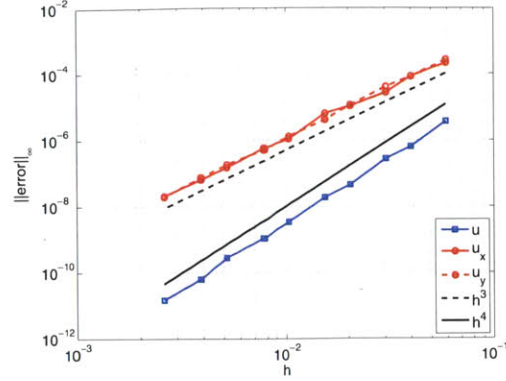


Figure 2-10: Example 2. Convergence of the error in the solution and its gradient evaluated along the interface.

this example the exact location of the interface is used. These interfaces are circles defined with the following parameters.

- Interface 1-2 (Big circle):

$$r_B = 0.3, \quad x_{0_B} = 0.5, \quad y_{0_B} = 0.5.$$

- Interface 2-3 (Small circle):

$$\begin{aligned} r_S &= 0.3, \\ x_{0_S} &= x_{0_B} + r_B \cos(\pi/e^2) - r_S \cos(\pi(1/e^2 + 1)), \\ y_{0_S} &= y_{0_B} + r_B \sin(\pi/e^2) - r_S \sin(\pi(1/e^2 + 1)). \end{aligned}$$

The point of contact is placed along the boundary of the big circle at the polar angle $\theta = \pi e^2$ – use the center of the big circle as the polar coordinates’ origin. This value of θ guarantees that no special alignments of the grid with the local geometry near the contact point can happen. Figure 2-11(a) shows the interface immersed in a Cartesian grid. Finally, these interfaces sub-divide the solution domain into

- Region 1: inside of the big circle.
- Region 2: outer region.

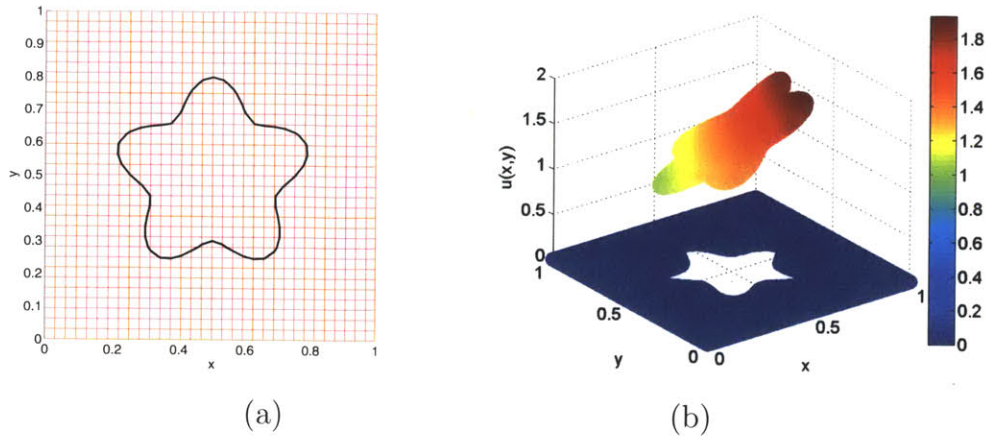


Figure 2-8: Example 2. Convergence of the error in the solution and its gradient in the L_2 and L_∞ norms.

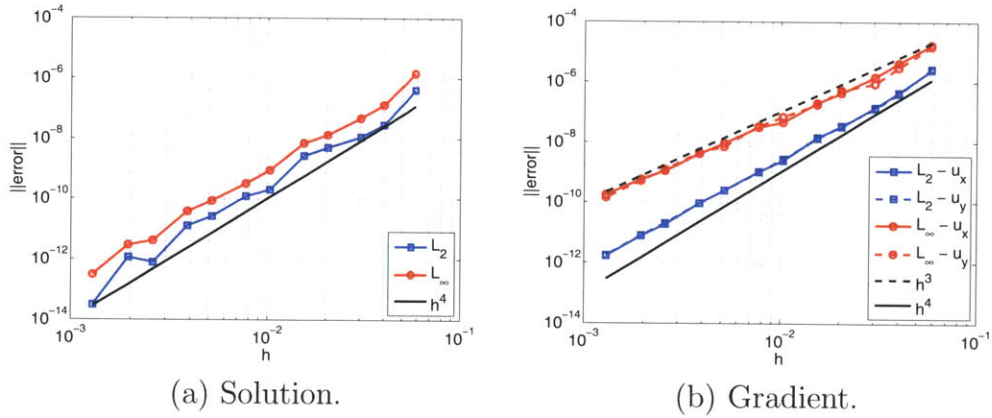


Figure 2-9: Example 2. Convergence of the error in the solution and its gradient in the L_2 and L_∞ norms.

The Poisson problem to be solved is the one associated with the exact solution

$$\begin{aligned}
 u_1(x, y) &= \sin(\pi x) \sin(\pi y) + 5, \\
 u_2(x, y) &= \exp(x) \{x^2 \sin(y) + y^2\}, \\
 u_3(x, y) &= \sin(\pi x) \{ \sin(\pi y) - \exp(\pi y) \}.
 \end{aligned}$$

Because of the presence of two interfaces, the solution domain is split into three distinct regions. The solution in each region is denoted by u_i , $i = 1, 2, 3$. The solution domain is the unit square $[0, 1] \times [0, 1]$. Furthermore, as mentioned above, in

the one associated with the exact solution

$$\begin{aligned}u_1(x, y) &= \sin(\pi x) \sin(\pi y) + 5, \\u_2(x, y) &= \sin(\pi x) \{\sin(\pi y) - \exp(\pi y)\}, \\u_3(x, y) &= \exp(x) \{x^2 \sin(y) + y^2\}.\end{aligned}$$

The interfaces are circles defined with the following parameters.

- Interface 2–3 (Big circle):

$$r_B = 0.3, \quad x_{0_B} = 0.5, \quad y_{0_B} = 0.5.$$

- Interface 1–2 (Small circle):

$$\begin{aligned}r_S &= 0.3, \\x_{0_S} &= x_{0_B} + (r_B - r_S) \cos(\pi/e^2), \\y_{0_S} &= y_{0_B} + (r_B - r_S) \sin(\pi/e^2).\end{aligned}$$

Figure 2-14(a) shows the interface immersed in a Cartesian grid. Finally, the regions defined by these interfaces are as follows.

- Region 1: inside small circle.
- Region 2: region between circles.
- Region 3: outer region.

The results are similar to example 3. Figure 2-14(b) shows the numerical solution obtained with a fine grid (193×193 nodes), while figure 2-15 shows the convergence of the error in the L_2 and L_∞ norms. The convergence of the error evaluated on the interface is shown in 2-16.

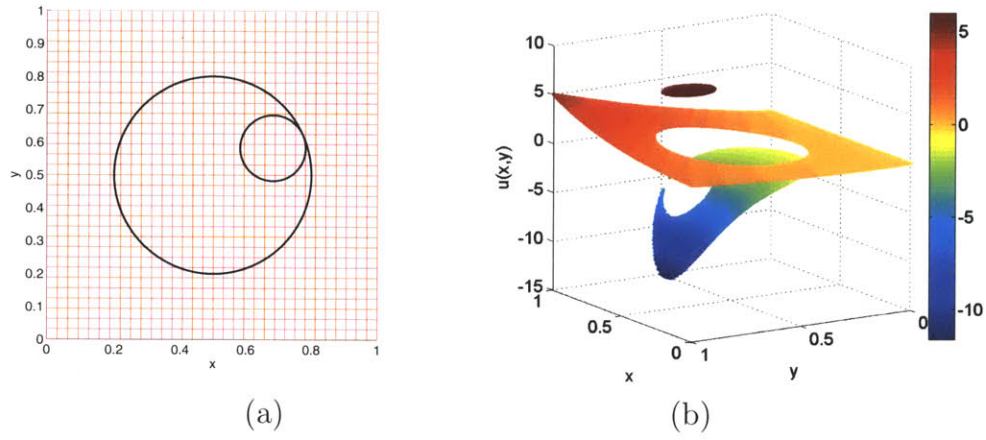


Figure 2-14: Example 4. Convergence of the error in the solution and its gradient in the L_2 and L_∞ norms.

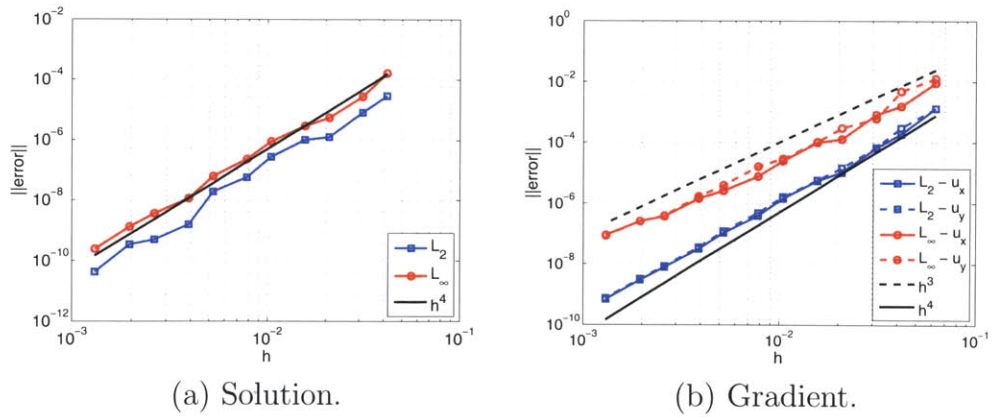


Figure 2-15: Example 4. Convergence of the error in the solution and its gradient in the L_2 and L_∞ norms.

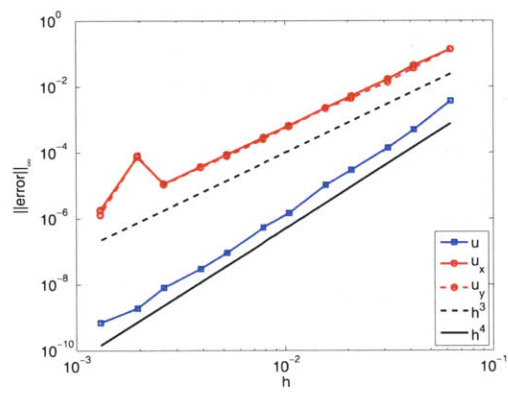


Figure 2-16: Example 4. Convergence of the error in the solution and its gradient evaluated along the interface.

Chapter 3

Extensions of the Correction Function Method

This chapter presents extensions of the correction function method to solve the Poisson equation in general settings. Specifically, this chapter is focused on (a) problems with arbitrarily shaped boundaries, and (b) the discontinuous coefficients Poisson equation. The basic concept behind the solution of these problems is the same introduced in §2: we define a correction function that is used to complete a standard finite differences discretization at nodes across the interface or the boundary. Moreover, this correction function is also the solution to a PDE defined locally in a neighborhood of the interface/boundary. The major challenge in these situations is that this PDE is no longer independent on the underlying solution to the Poisson equation. The approach to handle this coupling between the Poisson equation and the PDE that defines the correction function is discussed here.

This chapter is divided into two sections. In §3.1, the extension of the CFM to problems involving complex geometries is discussed, including results with Dirichlet and Neumann boundary conditions. Then, §3.2 is dedicated to the extension of the CFM to the discontinuous coefficients Poisson equation.

3.1 Boundary conditions on complex geometries

3.1.1 Overview

The objective here is to solve the following Poisson problem.

$$\Delta u(\vec{x}) = f(\vec{x}) \quad \text{for } \vec{x} \in \Omega, \quad (3.1)$$

with

$$u(\vec{x}) = g_D(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega, \quad (3.2a)$$

$$\text{or } u_m(\vec{x}) = g_N(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega. \quad (3.2b)$$

Here, u_m denotes the derivative of u in the direction of \hat{m} , the unit vector normal to the boundary $\partial\Omega$ pointing outwards (see figure 3-1). Furthermore, two types of boundary conditions are considered: Dirichlet (3.2a), or Neumann (3.2b).

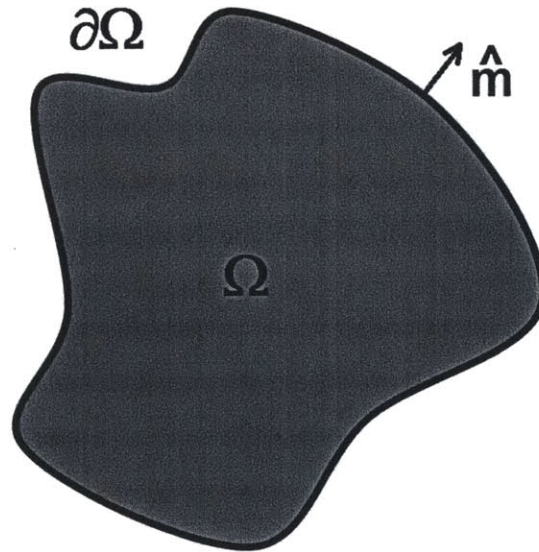


Figure 3-1: Example of a solution domain with arbitrary shape.

The goal is to be able to completely immerse the arbitrarily shaped solution domain in a regular Cartesian grid, while still imposing the appropriate boundary

conditions to high order accuracy. In §2 high order accuracy is achieved by defining a *correction function* as the solution to a partial differential equation. This section is dedicated to extending these ideas to solve the problem mentioned above. In §3.1.2 the extended concept of the correction function is presented, as well as the coupling that occurs between the correction function and the solution to the underlying Poisson equation. Next, §3.1.3 discusses how this coupling affects the definition of the rectangular regions where we solve for the correction function. Then, in §3.1.4 the solution procedure for problems with Dirichlet boundary condition is described. The solution to Neumann problems is similar, and is discussed in §3.1.5. Finally, §3.1.6 shows some results involving Dirichlet and Neumann boundary conditions.

3.1.2 The correction function and the equation defining it

The solution procedure adopted here is similar to that presented in §2.3 and §2.4. For simplicity, first consider the problem with Dirichlet boundary conditions. Then, define a narrow band Ω_Γ^1 , which is the set of all nodes within a distance \mathcal{R} from $\partial\Omega$. This distance \mathcal{R} depends on the details of the discretization, but is of the order of the grid spacing. Next, assume that u can be extended smoothly into Ω_Γ . Since there is no solution on the other side of $\partial\Omega$, the definition of a *correction function* must be adapted. In this context, the correction function, denoted by D , is defined to be the same as the extended solution within Ω_Γ^2 . Thus, the correction function is characterized as the solution to the following PDE.

$$\Delta D(\vec{x}) = f(\vec{x}) \quad \text{for } \vec{x} \in \Omega_\Gamma, \quad (3.3a)$$

$$D(\vec{x}) = g_D(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega, \quad (3.3b)$$

$$D_m(\vec{x}) = u_m(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega. \quad (3.3c)$$

¹There is no interface Γ in this problem, but the notation Ω_Γ is kept because the concept is the same as in §2.

²This distinction between the solution to the Poisson equation, u , and the correction function, D , is important for clarity in the discussion that follows.

Remark 3.1. *In (3.3), it becomes evident that, in general, the correction function depends on the underlying solution to Poisson's equation. In this case, the dependence occurs through (3.3c). ♣*

In principle, it is possible to approximate u_m on $\partial\Omega$ based on a linear combination of the unknown solution u at a set of grid points close to the boundary (*e.g.* using Taylor expansion). If we do so, the solution to (3.3) becomes a (linear) function of these unknowns. However, coming up with a systematic approximation that can handle the myriad of ways the boundary can be placed with respect to a Cartesian grid is not easy. Instead, the approach adopted here is to replace (3.3c) by another condition that maintains the effect of coupling the correction function to the values of the unknown solution. Namely, this condition is such that D must match u at a given set of grid nodes, *i.e.*

$$D(\vec{x}_k) = u_k \quad \text{for } k \in \mathcal{N}, \quad (3.4)$$

where \mathcal{N} is a pre-determined set of grid nodes.

Remark 3.2. *The net effect of making D a function of the unknowns at a set of grid nodes is a modification to the discretization of the Poisson equation next to the boundary. This modification is the result of using this correction function that depends on the unknown solution to complete the discretization stencil close to the boundary.*

Because the discretization is modified, the linear system that we must invert to solve Poisson's equation also changes. In fact, the linear system is different for each configuration of the computational grid and the solution domain. In addition, there are no guarantees that the modified linear system is self-adjoint (in general it is not). Chapter §4 presents an alternative approach that maintains the linear system untouched, at the price of solving an additional boundary integral equation. ♣

Remark 3.3. *Similar to the approach described in §2.4, Ω_Γ is sub-divided in a series of rectangular regions, $\Omega_\Gamma^{(i,j)}$, one for each stencil that straddles the boundary. By doing so, we can constrain the correction function to a local set of grid points, $\mathcal{N}^{(i,j)}$, and thus keep a compact discretization of Poisson's equation.*

The number of grid nodes to be included in $\mathcal{N}^{(i,j)}$ depends on how accurately we want to represent the correction function within $\Omega_{\Gamma}^{(i,j)}$. For the particular implementation discussed here, which is based on the bicubic interpolation to represent the correction function, experience shows that $\mathcal{N}^{(i,j)}$ can be the set of nodes that are both part of (a) the 9-point discretization stencil at node (i, j) , and (b) the solution domain. By defining $\mathcal{N}^{(i,j)}$ in this fashion, the final discretization stencil remains the same; only the weights are modified. As a consequence, the sparsity pattern of the linear system that must be inverted to solve the Poisson equation remains the same, which makes it easier to devise suitable pre-conditioners. ♣

3.1.3 Definition of $\Omega_{\Gamma}^{(i,j)}$

Here $\Omega_{\Gamma}^{(i,j)}$ is defined in a similar fashion to that used in §2.4.3. There are, however, two important differences that must be taken into account:

- (a) $\Omega_{\Gamma}^{(i,j)}$ must include the nodes contained in $\mathcal{N}^{(i,j)}$.
- (b) In some cases, $\Omega_{\Gamma}^{(i,j)}$ must extend beyond the area involved in the discretization stencil – see remark 3.4.

Remark 3.4. Item (a) limits the minimum size of $\Omega_{\Gamma}^{(i,j)}$ ³. In addition, $\Omega_{\Gamma}^{(i,j)}$ must include a piece of $\partial\Omega$ of length comparable to the diagonal of $\Omega_{\Gamma}^{(i,j)}$, as explained in §2.4.3. Therefore, if the minimum size dictated by item (a) does not contain enough of the boundary $\partial\Omega$, we must extend $\Omega_{\Gamma}^{(i,j)}$ in such a way as to include more of the boundary. Note that the extended $\Omega_{\Gamma}^{(i,j)}$ may involve additional grid nodes that are not part of $\mathcal{N}^{(i,j)}$. However, the condition (3.4) is not enforced on these extra nodes.

There is no unique way to extend $\Omega_{\Gamma}^{(i,j)}$. The solution adopted here is a mix between the approaches presented in §C.2 and §C.4, as discussed below.

From remarks 3.3 and 3.4, we conclude that the particular form of $\Omega_{\Gamma}^{(i,j)}$ depends on the choice of $\mathcal{N}^{(i,j)}$, which in turn may depend on the stencil used to discretize

³In the implementation presented here – based on the 9-point stencil and bicubic interpolation – the minimum size is a $2h_x \times 2h_y$ area.

the Poisson equation. For the the particular case of (a) using the 9-point stencil discretization of the Poisson equation (see appendix A), and (b) defining $\mathcal{N}^{(i,j)}$ as discussed in remark 3.3, $\Omega_{\Gamma}^{(i,j)}$ is constructed using the procedure below.

1. Find P , the point along the boundary that is closest to node (i, j) . We do not need to determine P very accurately. Small errors in P result only in small shifts in $\Omega_{\Gamma}^{(i,j)}$, which do not affect the quality of the solution.
2. Determine the piece of boundary of length $2\sqrt{h_x^2 + h_y^2}$ centered on P .
3. Find the coordinates (x_{\min_b}, x_{\max_b}) and (y_{\min_b}, y_{\max_b}) of the smallest rectangle that completely encloses the section of boundary mentioned in item 2 – see figure 3-2(a).
4. Find the coordinates (x_{\min_s}, x_{\max_s}) and (y_{\min_s}, y_{\max_s}) of the smallest rectangle that completely encloses all the nodes in the discretization stencil – see figure 3-2(b).
5. Then $\Omega_{\Gamma}^{(i,j)}$ is the smallest rectangle that encloses the two previous rectangles – see figure 3-2(c). Its edges are given by

$$x_{\min} = \min(x_{\min_b}, x_{\min_s}), \quad (3.5a)$$

$$x_{\max} = \max(x_{\max_b}, x_{\max_s}), \quad (3.5b)$$

$$y_{\min} = \min(y_{\min_b}, y_{\min_s}), \quad (3.5c)$$

$$y_{\max} = \max(y_{\max_b}, y_{\max_s}). \quad (3.5d)$$

Note that step 2 requires an explicit representation of the boundary in a neighborhood of node (i, j) . When an an implicit representation of the boundary is used (*e.g.* level set method), this information is not readily available. Hence, one must use information from neighboring grid cells to pre-compute an explicit representation of the boundary. For more details, see appendix C.

3.1.4 Solution of the Local PDE

Just as in the original version of the CFM (§2.4.4), (3.3) is solved within each $\Omega_{\Gamma}^{(i,j)}$ in a least squares sense. Namely, the solution procedure involves searching for a local

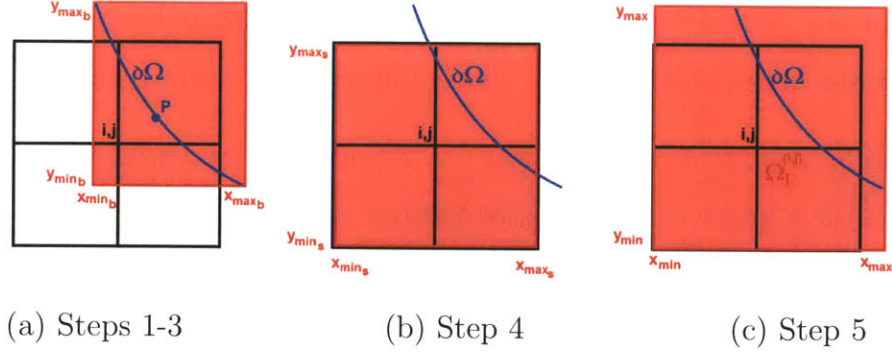


Figure 3-2: Steps involved in forming $\Omega_{\Gamma}^{(i,j)}$.

solution that minimizes the functional

$$\begin{aligned}
 J_b = & \frac{\ell_c^{(i,j)^4}}{2V(\Omega_{\Gamma}^{(i,j)})} \int_{\Omega_{\Gamma}^{(i,j)}} \{\Delta D(\vec{x}) - f(\vec{x})\}^2 dV \\
 & + c_P \frac{1}{2L(\Gamma)} \int_{\Gamma \cap \Omega_{\Gamma}^{(i,j)}} \{D(\vec{x}) - g_D(\vec{x})\}^2 dS \\
 & + c_N \frac{1}{2} \sum_{k \in \mathcal{N}^{(i,j)}} \{D(\vec{x}_k) - u_k\}^2,
 \end{aligned} \tag{3.6}$$

where $c_P > 0$ is the penalization coefficient used to enforce the Dirichlet boundary condition, and $c_N > 0$ is the penalization coefficient responsible to couple the correction function, D , to the underlying solution of the Poisson equation, u – see remark 3.6. Minimizing J_b , and therefore solving for D within each domain $\Omega_{\Gamma}^{(i,j)}$, involves the steps described below.

1. Choose a set of basis functions to represent D within $\Omega_{\Gamma}^{(i,j)}$: $D(\vec{x}) = \sum_{\ell=1}^n d_{\ell} \phi_{\ell}(\vec{x})$.
2. Replace this representation of D into the functional J_b .
3. Approximate the integrals in (3.6) using numerical quadrature.
4. Solve for the weights d_{ℓ} that minimize J_b .

In the particular scheme implemented for this thesis, the Poisson equation is discretized using the 4th order accurate 9-point stencil (see appendix A). Hence, the solution to the local PDE must be 4th order accurate (or better) to keep the overall

accuracy of the scheme. For this reason, D is approximated using cubic Hermite splines (bicubic interpolants in 2D), which guarantee 4th order accuracy – see [42].

Remark 3.5. *Note that, although (3.6) is a quadratic expression, the minimization of J_b results in a linear correspondence between the coefficients d_ℓ and the unknowns u_k . Hence, $d_\ell = d_\ell^{(0)} + \sum_{k=1}^{n_k} d_\ell^{(k)} u_k$, where n_k is the number of elements of $\mathcal{N}^{(i,j)}$. Then, one can solve for these coefficients as follows.*

- *Solving for $c_\ell^{(0)}$: set $u_k = 0$, $k = 1, \dots, n_k$.*
- *Solving for $c_\ell^{(k)}$: set $f = 0$, $g_D = 0$, $u_k = 1$, and $u_q = 0$, $q \neq k$.*

Consequently, if n basis functions are used to represent the solution, the minimization of (3.6) involves solving the same $n \times n$ self-adjoint linear system a total of $nk + 1$ times. In practice, it is useful to compute the LU decomposition of this linear system and solve for each coefficient performing a forward and backward substitution [43]. ♣

Remark 3.6. *As noted in remark 2.15, in principle the values of c_P and c_N can be determined from knowledge of derivatives of the solution. However, the different terms in J_b are scaled such that these coefficients should be $\mathcal{O}(1)$ quantities. Hence, in practice it is easy to manually adjust these values by observing the condition number of the minimization problem in a low resolution experiment. In the examples shown in §3.1.6 $c_P \approx 50$ and $c_N \approx 1$ produced good results. ♣*

3.1.5 Neumann boundary condition

The scheme described in §3.1.2 to §3.1.4 is an extension of the original CFM to enforce Dirichlet-type conditions on arbitrarily shaped boundaries immersed in a Cartesian grid. The procedure for Neumann-type boundary conditions is completely analogous.

Note that the accuracy of the solution scheme depends on how well (3.2b) is satisfied. For instance, when a bicubic interpolation is used as the basis to represent the correction function, condition (3.2b) is only imposed to 3rd order accuracy. Hence, when using bicubic interpolation to impose Neumann conditions, the accuracy of the CFM is 3rd order.

3.1.6 Results

This section shows two examples of computations in 2D using the scheme introduced in §3.1. The exact solution is the same in both examples, but they involve two different types of boundary conditions: Dirichlet for the first example, and Neumann for the second.

Both examples involve the Poisson problem associated with the exact solution

$$u(x, y) = \exp(x) \cos(y).$$

The boundary of the solution domain is represented as the zero contour of the level set function

$$\begin{aligned}\phi(x, y) &= r^2(x, y) - r^2(\theta(x, y)), \\ r(x, y) &= \sqrt{(x - x_0)^2 + (y - y_0)^2}, \\ r(\theta) &= r_0 + \epsilon \sin(5\theta), \\ \theta(x, y) &= \arctan\left(\frac{y - y_0}{x - x_0}\right),\end{aligned}$$

where, $x_0 = y_0 = 0.02\sqrt{5}$, $r_0 = 0.5$, and $\epsilon = 0.2$. Figure 3-3(a) shows the boundary immersed in a Cartesian grid. The solution domain Ω is the region contained inside this curve.

Figure 3-3(b) shows the numerical solution to the Dirichlet problem obtained with a fine grid (193×193 nodes). The solution outside Ω is simply set to zero. The solution to the Neumann problem is not shown because it is visually indistinguishable from the solution presented in figure 3-3(b). In addition, the convergence of the error for both cases are plotted in figure 3-4. Both the L_2 and L_∞ norms are shown. As expected, the error in the Dirichlet case converges to 4th order in both norms, whereas the error converges to 3rd order in the Neumann case.

In this version of the CFM, the correction function offers an approximate value of the solution in a neighborhood of the boundary. Hence, we can easily evaluate the solution and its gradient along the boundary. Figure 3-5 shows the convergence of

the error evaluated along the boundary.

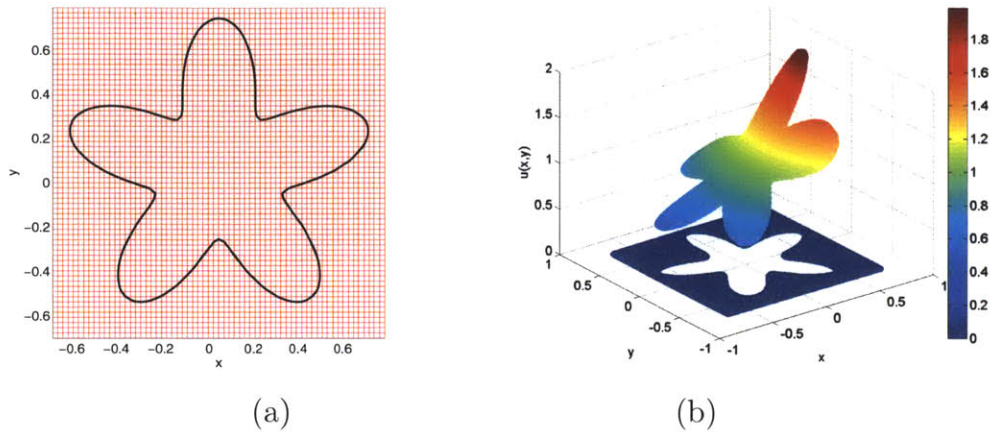


Figure 3-3: (a) Solution domain embedded in a 65×65 Cartesian grid. (b) Solution obtained with a 193×193 grid.

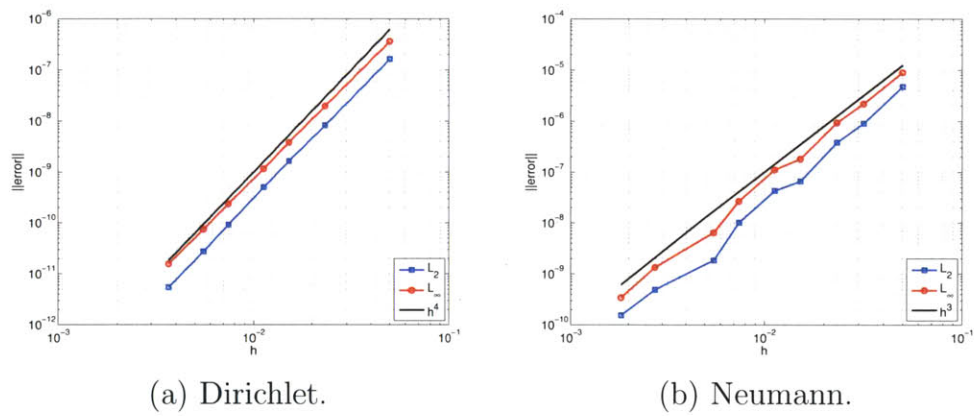


Figure 3-4: Convergence of the error in the L_2 and L_∞ norms.

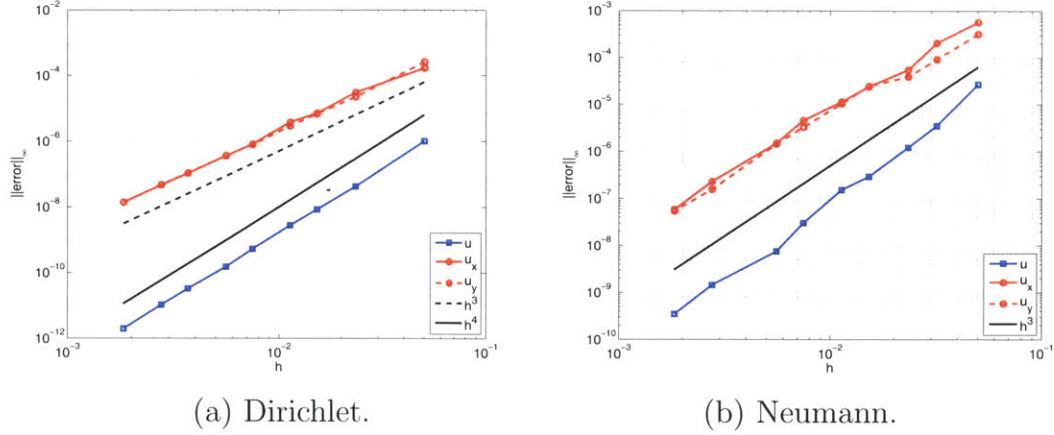


Figure 3-5: Convergence of the error evaluated along the interface.

3.2 Poisson's equation with discontinuous coefficient

3.2.1 Overview

Chapter §2 presents the correction function method to solve the constant coefficients Poisson equation with a prescribed discontinuity across an internal interface. This section discusses an extension of this method to solve the problem when, in addition, there is a discontinuity in the coefficients across the internal interface. For simplicity, assume Dirichlet-type condition on the boundary $\partial\Omega$ that delimits the solution domain Ω . Then, the problem to be solved is

$$\vec{\nabla} \cdot (\beta^+(\vec{x}) \vec{\nabla} u^+(\vec{x})) = f^+(\vec{x}) \quad \text{for } \vec{x} \in \Omega^+, \quad (3.7a)$$

$$\vec{\nabla} \cdot (\beta^-(\vec{x}) \vec{\nabla} u^-(\vec{x})) = f^-(\vec{x}) \quad \text{for } \vec{x} \in \Omega^-, \quad (3.7b)$$

$$[u] = a(\vec{x}) \quad \text{for } \vec{x} \in \Gamma, \quad (3.7c)$$

$$[\beta u_n] = b(\vec{x}) \quad \text{for } \vec{x} \in \Gamma, \quad (3.7d)$$

$$u(\vec{x}) = g(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega. \quad (3.7e)$$

The interface Γ is a co-dimension 1 manifold that subdivides the solution domain into Ω^+ and Ω^- (see figure 2-1). In addition, $\beta > 0$ is the coefficient of Poisson's equation. As mentioned above, in this section we are interested in the situation where β^+ and β^- are discontinuous along the interface Γ . Thus, in (3.7) the meaning of the brackets is

$$[\beta u_n] = \beta^+ u_n^+ - \beta^- u_n^-. \quad (3.8)$$

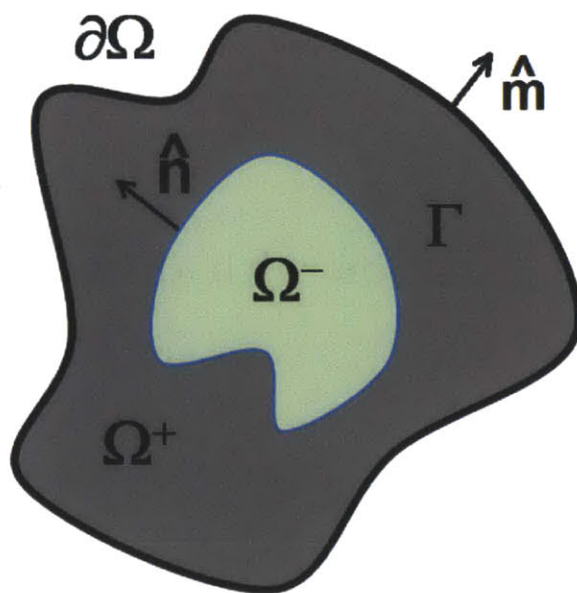


Figure 2-1: Example of solution domain with arbitrary shape.

We solve (3.7) following the basic CFM idea: evaluate smooth extensions of the solution across the interface such that we can apply the discretization stencil to the extended solution. Furthermore, the extensions are characterized as solutions to PDE satisfying appropriate conditions at the interface. However, in the present case it is not possible to define a PDE in terms of a single *correction function*, as in §2 and §3.1. In §3.2.2 the PDE that define the extensions of the solution are presented. Then §3.2.3 describes the procedure adopted to solve these PDE. Finally, §3.2.4 shows some results.

3.2.2 Smooth extensions of the solution

Once again, we define a narrow band Ω_Γ that is the set of all nodes within a distance \mathcal{R} from Γ . Similarly to other problems, \mathcal{R} depends on the details of the discretization, but is of the order of the grid spacing. Then, assume that u can be extended smoothly into Ω_Γ . For clarity, the extensions of the solution are denoted by v . Thus, v must solve the following PDE.

$$\vec{\nabla} \cdot (\beta^+(\vec{x}) \vec{\nabla} v^+(\vec{x})) = f^+(\vec{x}) \quad \text{for } \vec{x} \in \Omega_\Gamma, \quad (3.9a)$$

$$\vec{\nabla} \cdot (\beta^-(\vec{x}) \vec{\nabla} v^-(\vec{x})) = f^-(\vec{x}) \quad \text{for } \vec{x} \in \Omega_\Gamma, \quad (3.9b)$$

$$[v] = a(\vec{x}) \quad \text{for } \vec{x} \in \Gamma, \quad (3.9c)$$

$$[\beta v_n] = b(\vec{x}) \quad \text{for } \vec{x} \in \Gamma. \quad (3.9d)$$

Note that (3.9) is not written in terms of $D = u^+ - u^-$. In this case, the condition on the normal derivative of D becomes

$$D_n = \frac{b(\vec{x})}{\langle \beta \rangle} - \frac{[\beta]}{\langle \beta \rangle} \langle u_n \rangle \quad \text{for } \vec{x} \in \Gamma, \quad (3.10)$$

where

$$\langle \cdot \rangle = \frac{(\cdot)^+ + (\cdot)^-}{2} \quad (3.11)$$

is the mean value across a discontinuity. Equation (3.10) shows that, as long as the coefficient β is discontinuous, D depends on the solution to the underlying Poisson equation. Furthermore, the coupling occurs through the mean value of u_n .

Because of this coupling between D and the underlying solution to the Poisson equation, it is not convenient to characterize solve a PDE for D . Instead, here we solve (3.9) in terms of the solution extensions v^\pm .

Remark 3.7. *Note that (3.9) involves two unknown functions v^\pm , and that it alone does not determine them. To complete the PDE, we can use the same approach adopted in §3.1. Namely, extra conditions arise from the requirement that v^+ and v^- must match the solution to the Poisson problem on each side of the interface, as*

follows

$$v^+(\vec{x}_k) = u_k^+ \quad \text{for } k \in \mathcal{N}^+, \quad (3.12a)$$

$$v^-(\vec{x}_k) = u_k^- \quad \text{for } k \in \mathcal{N}^-, \quad (3.12b)$$

where \mathcal{N}^+ and \mathcal{N}^- are pre-determined set of grid nodes lying on Ω^+ and Ω^- , respectively. ♣

Remark 3.8. *The solution method described in §3.1 can be viewed as a particular case of solving (3.9) with (3.12). The difference is that in the present case the extensions of the solution define two correction functions, whereas in §3.1 we must solve a single correction function.*

Remark 3.9. *Similarly to what occurs in §3.1, the constraints (3.12) result in modifications to the discretization of the Poisson problem next to the interface, which is reflected in changes to the linear system that must be inverted to solve the Poisson equation (see remark 3.2). Chapter §4 presents an alternative approach that maintains the linear system untouched, at the price of solving an additional boundary integral equation.* ♣

Remark 3.10. *Just as in §2.4, Ω_Γ is sub-divided in a series of quadrangular regions. However, since (3.9) involves two unknowns, which interact only through the interface Γ , we can define two rectangular regions for each stencil: (i) $\Omega_\Gamma^{(i,j)^+}$, where v^+ is defined, and (ii) $\Omega_\Gamma^{(i,j)^-}$, where v^- is defined. The only requisite is that both regions involve the same piece of the interface Γ .*

This “splitting” of $\Omega_\Gamma^{(i,j)}$ into two pieces is not strictly necessary. However, it allows us to define v^+ and v^- in regions that are as small as possible. This feature contributes to reduce the condition number of the discretized version of (3.9) (see remarks 2.4 to 2.6). ♣

Remark 3.11. *By splitting $\Omega_\Gamma^{(i,j)}$ as described in remark 3.10, we can also apply the constraints (3.12) to local sets of grid points: $\mathcal{N}^{(i,j)^+}$ and $\mathcal{N}^{(i,j)^-}$. As a consequence, we are able to keep a compact discretization of the Poisson equation.*

The number of grid nodes to be included in the sets $\mathcal{N}^{(i,j)^+}$ and $\mathcal{N}^{(i,j)^-}$ depends on how accurately we want to represent the solution extensions. In particular, when representing the solution extensions using local bicubic interpolants, experience shows that $\mathcal{N}^{(i,j)^\pm}$ can be defined as the set of nodes that are both part of (i) the 9-point discretization stencil at node (i, j) and (ii) Ω^\pm . By doing so, the final discretization stencil remains the same; only the weights are modified. As a consequence, the sparsity pattern of the linear system that must be inverted to solve the Poisson equation remains the same, which makes it easier to devise suitable pre-conditioners. ♣

The definition of $\Omega_\Gamma^{(i,j)^\pm}$ implemented here is analogous to the case described in §3.1.3. The only difference is that only one of these regions must enclose all the nodes that are part of the discretization stencil (item 4 in §3.1.3). For instance, assume that node (i, j) lies in Ω^+ . Then $\Omega_\Gamma^{(i,j)^+}$ must enclose all the nodes that are part of the discretization stencil. On the other hand, $\Omega_\Gamma^{(i,j)^-}$ must only enclose the nodes that are part of $\mathcal{N}^{(i,j)^-}$. Figure 3-6 shows an example of how these regions are defined.

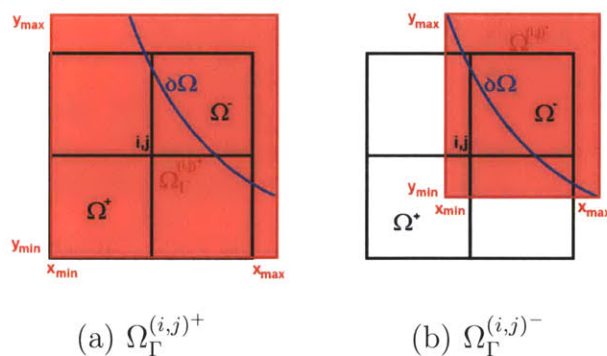


Figure 3-6: Regions where the extended solutions are defined. This example assumes that node (i, j) lies in Ω^+ .

3.2.3 Solution of the Local PDE

As pointed out in §3.2.2, the partial differential equation that characterizes the solution extensions v^+ and v^- – equation (3.9) – is similar to the PDE that characterizes the correction function in §3.1. Thus, we can solve for the solution extensions using

a procedure analogous to the one presented in §3.1.4. Namely, we search for a local solution that minimizes the potential

$$\begin{aligned}
J_{dc} &= \frac{(\ell_c^{(i,j)^+})^4}{2\beta^{+2}V(\Omega_\Gamma^{(i,j)^+})} \int_{\Omega_\Gamma^{(i,j)^+}} \left\{ \vec{\nabla} \cdot (\beta^+ \vec{\nabla} v^+(\vec{x})) - f^+(\vec{x}) \right\}^2 dV \\
&+ \frac{(\ell_c^{(i,j)^-})^4}{2\beta^{-2}V(\Omega_\Gamma^{(i,j)^-})} \int_{\Omega_\Gamma^{(i,j)^-}} \left\{ \vec{\nabla} \cdot (\beta^- \vec{\nabla} v^-(\vec{x})) - f^-(\vec{x}) \right\}^2 dV \\
&+ c_P \frac{1}{2L(\Gamma)} \int_{\Gamma \cap \Omega_\Gamma^{(i,j)}} \{v^+(\vec{x}) - v^-(\vec{x}) - a(\vec{x})\}^2 dS \\
&+ c_P \frac{1}{2L(\Gamma)} \left(\frac{\ell_m^{(i,j)}}{\langle \beta \rangle} \right)^2 \int_{\Gamma \cap \Omega_\Gamma^{(i,j)}} \{\beta^+ v_n^+(\vec{x}) - \beta^- v_n^-(\vec{x}) - b(\vec{x})\}^2 dS \\
&+ c_N \frac{1}{2} \left(\sum_{k \in \mathcal{N}^{(i,j)^+}} \{v^+(\vec{x}_k) - u_k^+\}^2 + \sum_{k \in \mathcal{N}^{(i,j)^-}} \{v^-(\vec{x}_k) - u_k^-\}^2 \right).
\end{aligned} \tag{3.13}$$

where $c_P > 0$ is the penalization coefficient used to enforce the jump conditions, and $c_N > 0$ is the penalization coefficient used to enforce (3.12) – see remark 3.12. Furthermore, $\ell_c^{(i,j)^+}$ and $\ell_c^{(i,j)^-}$ are characteristic lengths of $\Omega_\Gamma^{(i,j)^+}$ and $\Omega_\Gamma^{(i,j)^-}$, respectively – *e.g.* the smallest side length. In addition,

$$\ell_m^{(i,j)} = \min(\ell_c^{(i,j)^+}, \ell_c^{(i,j)^-}) \tag{3.14}$$

is a characteristic length of the intersection between the two rectangular regions. Finally, we solve for the minimum of J_{dc} following the steps described below.

1. Choose a set of basis functions to represent the solution extensions, each within its own rectangular domain of definition:

$$v^+(\vec{x}) = \sum_{\ell=1}^n q_\ell^+ \phi_\ell^+(\vec{x}), \quad v^-(\vec{x}) = \sum_{\ell=1}^n q_\ell^- \phi_\ell^-(\vec{x}).$$

2. Replace these representations of v^+ and v^- into the functional J_{dc} .
3. Approximate the integrals in (3.13) using numerical quadrature.
4. Solve for the weights q_ℓ^\pm that minimize J_{dc} .

Remark 3.12. *In principle, the penalization coefficients c_P and c_N depend on the knowledge of the solution. However, as pointed out in remarks 2.14 and 3.6, because of the scaling of the terms in J_{dc} , these coefficients are $\mathcal{O}(1)$ quantities that we can adjust manually using low resolution experiments. In the examples shown in §3.2.4 $c_P \approx 10$ and $c_N \approx 1$ produced good results.* ♣

Remark 3.13. *The minimization of J_{dc} is carried out as described in §3.1.4. Specifically, the dependence on the unknowns u_k can be handled by considering the influence from each grid node individually – see remark 3.5. Each of the two unknown functions is represented by a linear combination of n basis function. Therefore, the minimization procedure involves solving the same $2n \times 2n$ self-adjoint linear system $n_k + 1$ times, where n_k is the total number of grid nodes that are part of $\mathcal{N}^{(i,j)^+} \cup \mathcal{N}^{(i,j)^-}$. In practice, it is useful to compute the LU decomposition of this linear system and solve it performing a forward and backward substitution [43].* ♣

Remark 3.14. *After solving for the solution extensions, one must compute $D = v^+ - v^-$ to complete the discretization. However, (3.10) shows that D depends on $\langle u_n \rangle$. Just as in the case of the Neumann boundary condition in §3.1.5, the dependence on $\langle u_n \rangle$ produces one order of accuracy loss. In particular, if v^+ and v^- are represented using bicubic interpolants, $\langle u_n \rangle$ can be calculated with 3^{rd} order accuracy only, so that the overall scheme will be 3^{rd} order.* ♣

Remark 3.15. *Here the Poisson equation is discretized using the 9-point stencil (see appendix A). However, this compact discretization of the Poisson equation is only 4^{th} order accurate if the coefficient β is piece-wise constant (i.e. β^+ and β^- are two distinct constants). This is the case in the examples shown in §3.2.4. In more general applications one must resort to wider discretizations of the Poisson equation to obtain high order of accuracy.* ♣

Remark 3.16. *This version of the CFM can be seen as a generalization of the immersed interface method (IMM) [50]. The IIM uses Taylor expansions around one particular point on the interface, together with the appropriate jump conditions, to devise a 2^{nd} order local discretization of the Poisson equation. In contrast, in the CFM*

the Taylor expansion is replaced by the PDE (3.9). As a result, whereas manipulating Taylor expansions to obtain high order discretizations is very complicated, the CFM can be used to produce high order discretizations a straightforward fashion.

Remark 3.17. *Many applications of multiphase flows involve the solution of the discontinuous coefficient Poisson equation with a large ratio between coefficients (e.g. in air-water interface the ratio is 1:1,000). The solution procedure discussed above is general enough to deal with arbitrary ratios between coefficients.*

Note that the case $\beta^-/\beta^+ \gg 1$ is special. In the limit $\beta^-/\beta^+ \rightarrow \infty$ equation (3.7d) tends to $u_n^- \rightarrow 0$. As a consequence, u^- becomes the solution to the Poisson equation with Neumann boundary condition on Γ , which is only defined up to an arbitrary constant. If the solution is defined only up to a constant in the entire domain (such as when we impose Neumann or periodic boundary conditions on $\partial\Omega$), then this situation poses no difficulty. We can set this arbitrary constant by imposing additional constraints to the solution. This is the most common situation that occurs in physical problems.

On the other hand, when Dirichlet-type conditions are imposed on $\partial\Omega$ the solution is not arbitrary. Nonetheless, as β^-/β^+ grows, the solution to (3.7) becomes increasingly ill conditioned. This issue is intrinsic to the equation being solved and is not related to the numerical method used to solve it. The same problem is observed in Ref. [77]. In this paper, high condition numbers are listed for the discontinuous coefficient Poisson equation, even though the authors do not make further comments about it. An example where we can observe this issue is presented in 3.2.4. In addition, the boundary integral formulation discussed in 4 allows us to “fix” this situation by enforcing redundant integral conditions – see appendix D. ♣

3.2.4 Results

This section presents four examples of computations in 2D using the scheme introduced in §3.2. All the examples use the same solution, but with different choices for β (and corresponding changes in the jump conditions).

The examples involve the Poisson problem associated with the exact solution⁴

$$u(x, y)^+ = 0.1(x^2 + y^2)^2 - 0.01 \log(2\sqrt{x^2 + y^2}),$$

$$u(x, y)^- = x^2 + y^2.$$

The boundary of the solution domain is represented as the zero contour of the of the level set function

$$\phi(x, y) = r^2(x, y) - r^2(\theta(x, y)),$$

$$r(x, y) = \sqrt{(x - x_0)^2 + (y - y_0)^2},$$

$$r(\theta) = r_0 + \epsilon \sin(5\theta),$$

$$\theta(x, y) = \arctan\left(\frac{y - y_0}{x - x_0}\right),$$

where, $x_0 = 0.03\sqrt{5}$, $y_0 = 0.02\sqrt{5}$, $r_0 = 0.5$, and $\epsilon = 0.2$. Figure 3-7(a) shows the boundary immersed in a Cartesian grid. The sub-domain Ω^- is the region contained inside this curve, while Ω^+ is the region that lies outside this curve. Finally, the examples considered here are associated to the coefficients

- (i) $\beta^+ = 10$, $\beta^- = 1$.
- (ii) $\beta^+ = 1$, $\beta^- = 10$.
- (iii) $\beta^+ = 1 \times 10^6$, $\beta^- = 1$.
- (iv) $\beta^+ = 1$, $\beta^- = 1 \times 10^6$.

Figure 3-7(b) shows the numerical solution to problem (i) obtained with a fine grid (193×193 nodes). The solution to problems (ii) to (iv) are similar and visually indistinguishable. Convergence plots of the error are shown in figure 3-8. Both the L_2 and L_∞ norms are shown. As expected, as the grid is refined the error decays to 3rdorder in both norms for all examples.

⁴One can obtain an accurate and smooth extension of u^+ because the interface is always more than a grid space away from the singularity at $(x, y) = (0, 0)$.

These plots show that the solution procedure discussed in §3.2 produces good results even for large ratios of coefficients. However, note that the error for case (iv) is significantly larger than in other cases. The reason for the larger errors is that $\beta^-/\beta^+ \gg 1$, so case (iv) represents the ill-posed situation described in remark 3.17. In fact, in example (iv) the condition number of the linear system we must invert to solve the Poisson equation is $\mathcal{O}(\beta^-/\beta^+)$. Similar results are observed in the solution evaluated along the interface, as shown in figure 3-9.

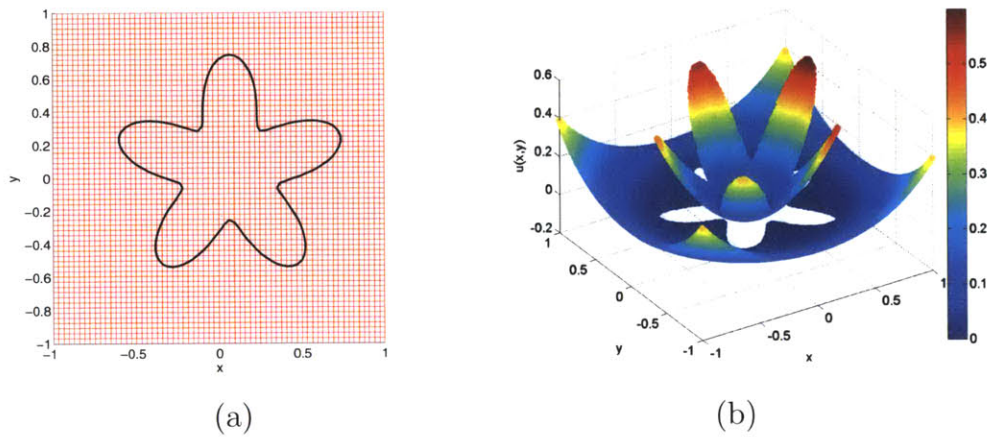
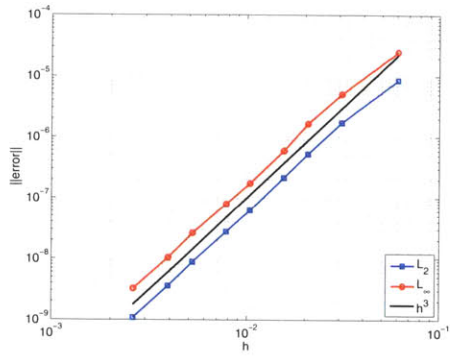
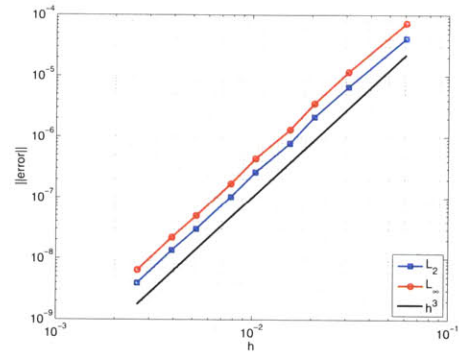


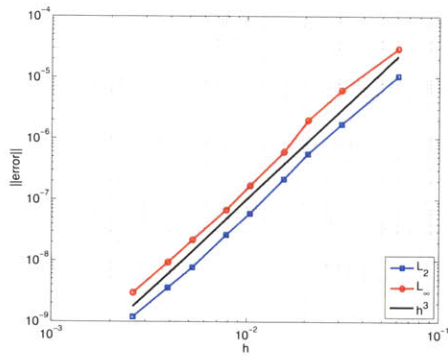
Figure 3-7: (a) Solution domain embedded in a 65×65 Cartesian grid. (b) Solution obtained with a 193×193 grid.



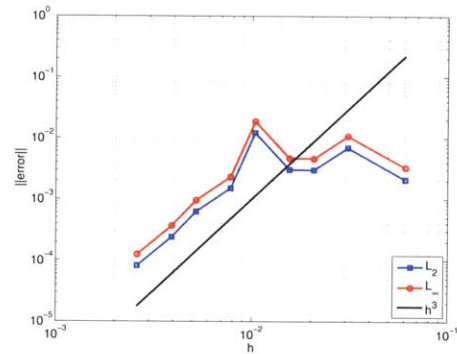
(i) $\beta^+ = 10, \beta^- = 1.$



(ii) $\beta^+ = 1, \beta^- = 10.$

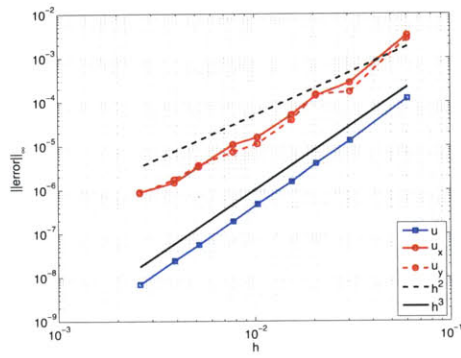


(iii) $\beta^+ = 1 \times 10^6, \beta^- = 1.$

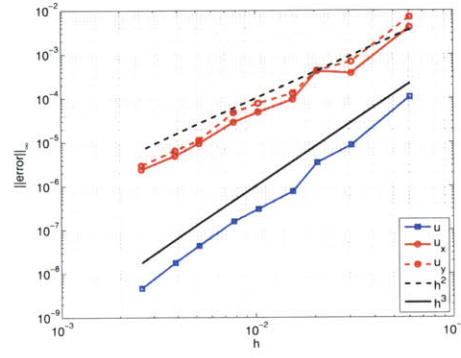


(iv) $\beta^+ = 1, \beta^- = 1 \times 10^6.$

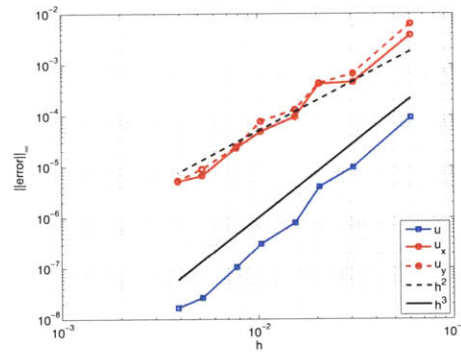
Figure 3-8: Convergence of the error in the L_2 and L_∞ norms.



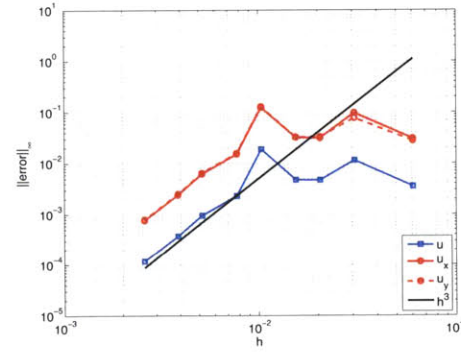
(i) $\beta^+ = 10, \beta^- = 1.$



(ii) $\beta^+ = 1, \beta^- = 10.$



(iii) $\beta^+ = 1 \times 10^6, \beta^- = 1.$



(iv) $\beta^+ = 1, \beta^- = 1 \times 10^6.$

Figure 3-9: Convergence of the error along the interface.

Chapter 4

Alternative Method to Solve the Poisson equation – using boundary integral equations

This chapter presents an alternative to §3 to solve the Poisson equation involving complex geometries using regular Cartesian grids. This algorithm can be applied to a wide variety of Poisson problems, including those where the coefficients and the solution are discontinuous across some arbitrary interface.

The solution procedure described here is based on the combination of the original version of the correction function method and boundary integral equations. As discussed in §2, the CFM can be used to solve, to high order of accuracy, the constant coefficients Poisson equation with discontinuities across an arbitrary interface. In more general situations, one can use the ideas introduced by Mayo [31] to rewrite the original problem as a series of constant coefficients Poisson equations with discontinuous solutions. In short, the problem is split into (i) a simple constant coefficients Poisson equation to handle the source term, and (ii) a Laplace equation where the appropriate interface and boundary conditions are satisfied. Next, the Laplace equation is solved using the corresponding boundary integral formulation. There is a number of well established, fast, and accurate numerical methods – boundary integral methods (BIMs) [31, 88, 89] – that can be used to solve this problem. Then, the solution

obtained with the BIM offers all the information needed to rewrite this problem as a constant coefficients Laplace equation with known discontinuities across an interface, which can be accurately solved using a finite differences discretization and the CFM. The solution procedure is discussed in detail in §4.1.

The accuracy of the solution procedure proposed here results from the fact that, in principle, BIMs and the CFM can be implemented to any desired order of accuracy. Moreover, with the exception of solving the BIM, all other steps involve problems defined in rectangular domains, which can be solved rapidly using the fast Fourier transform (FFT). Hence, the solution procedure is also fast, since the costliest steps involve fast solutions of boundary integral equations [90–92], and a fast Poisson solver based on FFT.

The remainder of this chapter is organized as follows. In §4.1 the solution procedure is explained using a series of problems, increasing gradually in complexity. Then, §4.2 shows some examples where this procedure is applied to problems involving complex geometries and the Poisson equation with piece-wise constant coefficients. Finally, in appendix D a possible fix for a particular configuration that makes the Poisson equation with piece-wise constant coefficients ill-conditioned is discussed.

4.1 Solution procedure

The algorithm presented here builds on the ideas introduced by Mayo [31,63] and on the original version of the CFM – see §2. For simplicity, the solution procedure is presented through a series of problems, increasing gradually in complexity. First, §4.1.1 discusses the solution of the simple problem of solving the Laplace equation with Dirichlet boundary conditions. Second, in §4.1.2 a non-homogeneous source term is added and the solution to the Poisson equation is discussed. Third, §4.1.3 shows the solution of the Poisson equation with piece-wise constant coefficients Poisson equation. Finally, in §4.1.4 the solution procedure for general situations is summarized.

4.1.1 Laplace equation

Consider the following Laplace equation with Dirichlet boundary conditions. (In this problem the solution is smooth throughout Ω .)

$$\Delta u(\vec{x}) = 0 \quad \text{for } \vec{x} \in \Omega, \quad (4.1a)$$

$$u(\vec{x}) = g_D(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega. \quad (4.1b)$$

It is well known [88, 93] that the solution of this equation can be expressed as

$$u(\vec{x}) = \frac{1}{2\pi} \int_{\partial\Omega} \mu(\vec{x}_s) G_m(\vec{x}, \vec{x}_s) dS \quad \text{for } \vec{x} \in \Omega, \quad (4.2)$$

where \vec{x}_s denotes the position vector along the integration surface, G is the Green's function for the Laplace equation¹, and μ is a function defined along the boundary $\partial\Omega$, known as dipole or double layer potential.

Equation (4.2) is the boundary integral representation of the solution to (4.1). Following this representation, the solution is completely described in terms of the potential μ , which becomes the unknown of the problem. To solve for this potential, we must enforce the boundary condition (4.1b), resulting in the following boundary integral equation.

$$\frac{1}{2\pi} \int_{\partial\Omega} \mu(\vec{x}_s) G_m(\vec{x}, \vec{x}_s) dS = g_D(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega. \quad (4.3)$$

Equation (4.3) is a Fredholm integral equation of the second kind. This class of integral equations has been extensively studied and there is a number of well established numerical methods – boundary integral methods (BIMs) – that can be used to solve (4.3) [88, 89]. In 2D, when the boundary $\partial\Omega$ and the data g_D are smooth, Nystrom's method with trapezoidal quadrature is known to be very efficient and accurate (see [89, 93]). For this reason, this is the method adopted to obtain the results shown in §4.2. In 3D there are better suited methods, such as Galerkin's method

¹In 2D, $G(\vec{x}, \vec{x}_s) = \log(|\vec{x} - \vec{x}_s|)$; in 3D $G(\vec{x}, \vec{x}_s) = \frac{1}{|\vec{x} - \vec{x}_s|}$.

with Gaussian quadrature.

Nevertheless, although μ can be computed efficiently and accurately, the integral (4.2) still needs to be evaluated to obtain the solution inside Ω . This computation can be relatively expensive when the solution is needed in a large number of points inside Ω . In addition, the integrand of (4.2) becomes singular as $\vec{x} \rightarrow \vec{x}_s$ for $\vec{x} \notin \partial\Omega$, which makes evaluating this integral even more expensive for points close to the boundary. In order to circumvent these difficulties, Mayo [31] introduced an alternative approach to evaluate the solution within Ω . In short, Mayo shows that the integral expression (4.2) is also the solution to the following Laplace equation.

$$\Delta u(\vec{x}) = 0 \quad \text{for } \vec{x} \in \mathcal{B}, \quad (4.4a)$$

$$[u] = -\mu(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega, \quad (4.4b)$$

$$[u_m] = 0 \quad \text{for } \vec{x} \in \partial\Omega, \quad (4.4c)$$

$$u(\vec{x}) = \frac{1}{2\pi} \int_{\partial\Omega} \mu(\vec{x}_s) G_m(\vec{x}, \vec{x}_s) dS \quad \text{for } \vec{x} \in \partial\mathcal{B}, \quad (4.4d)$$

where \mathcal{B} is a rectangular domain that completely involves the original solution domain Ω (see figure 4-1). Hence, in (4.4) $\partial\Omega$ becomes an interface internal to \mathcal{B} , across which the solution is discontinuous. In addition, the discontinuity in the solution depends only on the potential μ . Thus, the solution proposed by Mayo [31] involves two steps:

1. solving (4.3) using a BIM.
2. solving (4.4) using finite differences.

For the second step, the original version of the CFM, introduced in §2, offers a framework to solve (4.4) using finite differences with the solution domain immersed in a regular Cartesian grid. In summary, with the CFM we can compute a correction function in a narrow band surrounding the “interface” $\partial\Omega$. Whenever the discretization stencil straddles this interface, the correction function is used to complete the discretization. This procedure results in modifications only to the right-hand-side (RHS) of the discretized equation, without modifications to the linear system that must be inverted. In addition, in theory this correction function can be evaluated

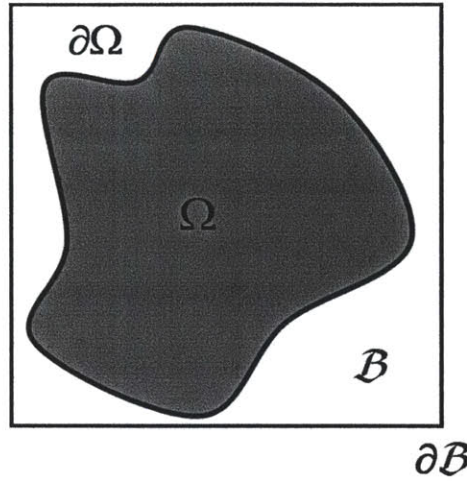


Figure 4-1: Rectangular domain \mathcal{B} that involves Ω .

to arbitrary order of accuracy. Chapter §2 shows a 4th order implementation of the CFM, which is the same one used to obtain the results shown in §4.2.

Remark 4.1. *Mayo [31, 63] also presents a method to solve (4.4) using finite differences in an immersed setting. Furthermore, this method is also based on corrections to the RHS of the discretized equation, and Mayo [31, 63] derived these corrections up to 4th order of accuracy. The major difference is that Mayo's method requires accurate computation of derivatives of μ to evaluate the correction terms.* ♣

Remark 4.2. *The rectangular domain \mathcal{B} is arbitrary. In principle, should \mathcal{B} enclose Ω as tightly as possible to reduce the number of grid points outside the region of interest. However, evaluating (4.4d) too close to $\partial\Omega$ can be difficult. Hence, it is practical to maintain $\partial\mathcal{B}$ within some distance of $\partial\Omega$. The present implementation uses the distance suggested by Mayo [31] of three grid spacings.* ♣

Remark 4.3. *Since (4.4) is defined in a rectangular domain, we can use the fast Fourier transform (FFT) to solve the linear system that comes from the finite differences discretization of the Laplace equation. This fact makes this approach very cost effective².* ♣

²This is not a spectral method. The FFT can be used to invert the linear system because the finite differences discretization of Laplace's equation results in a circular periodic linear system.

Remark 4.4. *The BIM requires a discretization of $\partial\Omega$. However, this discretization of $\partial\Omega$ is completely independent of the computational grid used to solve (4.4) with finite differences. In fact, usually the BIM converges faster than finite differences, so that the discretization of $\partial\Omega$ can be much coarser than the grid used to solve (4.4). ♣*

4.1.2 Including a non-homogeneous source

We can take advantage of the CFM and the linearity of the Laplace operator to include a non-homogeneous source term in (4.1a) in a straightforward fashion. Suppose that (4.1a) is replaced by $\Delta u(\vec{x}) = f(\vec{x})$, $\vec{x} \in \Omega$. Then, we first solve the auxiliary problem

$$\Delta v(\vec{x}) = 0 \quad \text{for } \vec{x} \in \mathcal{B} - \Omega, \quad (4.5a)$$

$$\Delta v(\vec{x}) = f(\vec{x}) \quad \text{for } \vec{x} \in \Omega, \quad (4.5b)$$

$$[v] = 0 \quad \text{for } \vec{x} \in \partial\Omega, \quad (4.5c)$$

$$[v_n] = 0 \quad \text{for } \vec{x} \in \partial\Omega, \quad (4.5d)$$

$$v(\vec{x}) = 0 \quad \text{for } \vec{x} \in \partial\mathcal{B}, \quad (4.5e)$$

where \mathcal{B} is the same rectangular domain defined in (4.4). Moreover, since (4.5) is defined on a rectangular domain, it can also be solved very efficiently using finite differences combined with the CFM and FFT.

Remark 4.5. *Standard finite difference discretizations depend on smooth solutions to guarantee accuracy (e.g. C^3 for 2nd order and C^5 for 4th order). In (4.5) the solution is continuous up to the first derivative, but the Laplacian is discontinuous. Hence, we need an algorithm such as the CFM to handle discontinuities in the source term without losing accuracy. ♣*

After we solve (4.5), we are left to solve a Laplace equation for $w = u - v$:

$$\Delta w(\vec{x}) = 0 \quad \text{for } \vec{x} \in \Omega, \quad (4.6a)$$

$$w(\vec{x}) = g_D(\vec{x}) - v(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega. \quad (4.6b)$$

This equation can be solved by combining a BIM with finite differences and the CFM, as described in §4.1.1. Therefore, the Poisson equation with Dirichlet boundary conditions is solved following the steps described below.

1. Compute v by solving (4.5) using finite differences with the CFM and FFT.
2. Compute w by solving (4.6) using the BIM and finite differences with the CFM and FFT.
3. Set $u = v + w$.

4.1.3 Poisson equation with Piece-wise constant coefficients

Consider now the Poisson equation with piece-wise constant coefficients defined in (3.7), where $\beta^+ > 0$ and $\beta^- > 0$ are two distinct constants. Furthermore, for simplicity, assume Dirichlet boundary conditions – equation (3.7e). The procedure to solve this problem is similar to the one described in §4.1.1 and §4.1.2. We first address the source term by solving

$$\Delta v(\vec{x}) = 0 \quad \text{for } \vec{x} \in \mathcal{B} - \Omega, \quad (4.7a)$$

$$\Delta v(\vec{x}) = f^+(\vec{x})/\beta^+ \quad \text{for } \vec{x} \in \Omega^+, \quad (4.7b)$$

$$\Delta v(\vec{x}) = f^-(\vec{x})/\beta^- \quad \text{for } \vec{x} \in \Omega^-, \quad (4.7c)$$

$$[v] = 0 \quad \text{for } \vec{x} \in \partial\Omega, \quad (4.7d)$$

$$[v_n] = 0 \quad \text{for } \vec{x} \in \partial\Omega, \quad (4.7e)$$

$$[v] = a(\vec{x}) \quad \text{for } \vec{x} \in \Gamma, \quad (4.7f)$$

$$[v_n] = 0 \quad \text{for } \vec{x} \in \Gamma, \quad (4.7g)$$

$$v(\vec{x}) = 0 \quad \text{for } \vec{x} \in \partial\mathcal{B}. \quad (4.7h)$$

Here \mathcal{B} is once again the same rectangular domain defined in (4.4). Note that the problem defined in (4.7) has two “internal interfaces:” (a) Γ , across which the solution and the Laplacian are discontinuous, and (b) $\partial\Omega$, across which the Laplacian

is discontinuous. This problem can be solved accurately and efficiently using finite differences with the CFM and FFT.

The jump in the normal fluxes – equation (3.7d) – is imposed by solving the following Laplace equation.

$$\Delta w(\vec{x}) = 0 \quad \text{for } \vec{x} \in \Omega, \quad (4.8a)$$

$$[w] = 0 \quad \text{for } \vec{x} \in \Gamma, \quad (4.8b)$$

$$[w_n] + \lambda \langle w_n \rangle = b(\vec{x}) / \langle \beta \rangle - \lambda v_n(\vec{x}) \quad \text{for } \vec{x} \in \Gamma, \quad (4.8c)$$

$$w(\vec{x}) = g_D(\vec{x}) - v(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega, \quad (4.8d)$$

where

$$\langle \cdot \rangle = \frac{(\cdot)^+ + (\cdot)^-}{2} \quad (3.11)$$

denotes the mean value, and $\lambda = [\beta] / \langle \beta \rangle$. The solution to this Laplace equation can be written as

$$w(\vec{x}) = \frac{1}{2\pi} \int_{\Gamma} \rho(\vec{x}_s) G(\vec{x}, \vec{x}_s) dS + \frac{1}{2\pi} \int_{\partial\Omega} \mu(\vec{x}_s) G_m(\vec{x}, \vec{x}_s) dS \quad \text{for } \vec{x} \in \mathcal{B}. \quad (4.9)$$

In (4.9), ρ denotes a function defined over Γ , known as the monopole or single layer potential. Similar to the dipole potential, the monopole potential is also commonly used in the solution of the Laplace equation with a boundary integral formulation [88, 89]. Expression (4.9) automatically satisfies conditions (4.8a) and (4.8b). Then, ρ and μ are defined by imposing (4.8c) and (4.8d), which results in the following coupled system of boundary integral equations.

$$\begin{aligned} \rho(\vec{x}) + \frac{\lambda}{2\pi} \int_{\Gamma} \rho(\vec{x}_s) G_n(\vec{x}, \vec{x}_s) dS \\ + \frac{\lambda}{2\pi} \int_{\partial\Omega} \mu(\vec{x}_s) G_{nm}(\vec{x}, \vec{x}_s) dS = \frac{b}{\langle\beta\rangle} - \lambda v_n \end{aligned} \quad \text{for } \vec{x} \in \Gamma, \quad (4.10a)$$

$$\begin{aligned} \mu(\vec{x}) + \frac{\lambda}{\pi} \int_{\Gamma} \rho(\vec{x}_s) G(\vec{x}, \vec{x}_s) dS \\ + \frac{\lambda}{\pi} \int_{\partial\Omega} \mu(\vec{x}_s) G_m(\vec{x}, \vec{x}_s) dS = 2(g_D(\vec{x}) - v(\vec{x})) \end{aligned} \quad \text{for } \vec{x} \in \partial\Omega. \quad (4.10b)$$

After one solves (4.10) for the potentials μ and ρ , w is evaluated by noting that the integral expression (4.9) is also the solution to the following Laplace equation defined in \mathcal{B} :

$$\Delta w(\vec{x}) = 0 \quad \text{for } \vec{x} \in \mathcal{B}, \quad (4.11a)$$

$$[w] = -\mu(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega, \quad (4.11b)$$

$$[w_n] = 0 \quad \text{for } \vec{x} \in \partial\Omega, \quad (4.11c)$$

$$[w] = 0 \quad \text{for } \vec{x} \in \Gamma, \quad (4.11d)$$

$$[w_n] = \rho(\vec{x}) \quad \text{for } \vec{x} \in \Gamma, \quad (4.11e)$$

$$\begin{aligned} w(\vec{x}) = \frac{1}{2\pi} \int_{\Gamma} \rho(\vec{x}_s) G(\vec{x}, \vec{x}_s) dS \\ + \frac{1}{2\pi} \int_{\partial\Omega} \mu(\vec{x}_s) G_{n_s}(\vec{x}, \vec{x}_s) dS \end{aligned} \quad \text{for } \vec{x} \in \partial\mathcal{B}. \quad (4.11f)$$

Equation (4.11) is solved using finite differences with the CFM and FFT.

Therefore, the Poisson equation with piece-wise constant coefficients is solved following the steps described below.

1. Compute v by solving (4.7) using finite differences with the CFM and FFT.
2. Compute w by solving (4.8) using the BIM and finite differences with the CFM and FFT.
3. Set $u = v + w$.

Remark 4.6. Note that (4.10a) involves evaluating v_n along Γ . In general, v_n is one order less accurate than the nominal accuracy of the CFM. Furthermore, since the solution to (4.10) determines the jump conditions for (4.11), the overall solution will be one order less accurate than the nominal accuracy of the CFM. For instance, if a 4th order accurate CFM is used, v_n along Γ is 3rd order accurate. Then it follows that ρ and μ are restricted to 3rd order accuracy, and so are w and u . ♣

Remark 4.7. As noted in remark 3.17, the case where $\beta^-/\beta^+ \gg 1$ leads to an ill-conditioned Poisson equation. This issue is intrinsic to the equation being solved and is not related to the boundary integral formulation nor the numerical method used to solve it. Appendix D discusses a possible fix for this problem, based on enforcing redundant integral conditions. ♣

4.1.4 Summary

The solution procedure in §4.1.1 through §4.1.3 is as follows: we first solve a simplified Poisson equation to handle the source term, and then solve the resulting Laplace equation using a combination of boundary integral formulation and finite differences, with the CFM and FFT. In principle, the Poisson equation with general interface and boundary conditions – including Neumann boundary condition and external problems with finite support – can be solved following these same basic steps. The only difference lies in the boundary integral formulation that solves the Laplace equation with the corresponding interface and boundary conditions. Hence, in summary the solution process involves the steps listed below.

1. **Handling of the source term:** we solve a Poisson equation in the rectangular domain \mathcal{B} . The source term is known in Ω and is set to zero in $\mathcal{B}-\Omega$. In addition, all interface and boundary conditions are also set to zero. This problem can then be solved using finite differences with the CFM and FFT.
2. **Imposing interface and boundary conditions:** we solve the remaining Laplace equation with the appropriate interface and boundary conditions. This step can be split into four sub-steps:

- (a) Write the boundary integral formulation corresponding to the Laplace equation.
 - (b) Solve for the potential distributions using a BIM.
 - (c) Write the equivalent Laplace equation defined in \mathcal{B} . The solution to this equation may be discontinuous across interfaces and boundaries, and the discontinuities depend exclusively on the potential distributions obtained with the BIM.
 - (d) Solve this equivalent Laplace equation using finite differences with the CFM and FFT.
3. **Final solution:** the final solution is the sum of each component described above.

In terms of accuracy, there are four factors must be considered:

1. **Representation of interfaces and boundaries.** The solution is at most as accurate as the interface and boundary conditions. In turn, these conditions are only known as accurately as the position of the interfaces and boundaries. Here interfaces and boundaries are represented using the gradient-augmented level-set (GA-LS) method [42]. This method results in a 4th order accurate representation of these geometries using local grid information.
2. **The accuracy of the BIM.** The accuracy of these methods depends on the smoothness of interfaces and boundaries, and the smoothness of the data provided on these surfaces. For smooth and well resolved surfaces, Nystrom's method is guaranteed to converge as fast as the quadrature rule used to approximate the integrals [89].
3. **Interpolation of the potential distributions.** Some BIM result in the potential distribution defined all along the interfaces (*e.g.* Galerkin's method). However, most BIM result in the potential values over a finite number of nodes along the interface. In these cases, these values must be interpolated to compute the jump conditions where it is required by the CFM. Here only 2D and

smooth geometries are considered, so trigonometric interpolation [94] is used. This interpolation can be computed efficiently with the FFT and has optimal accuracy.

4. **The accuracy of the CFM.** In principle, the CFM can achieve arbitrary order of accuracy, depending only on the smoothness of the data that defines the problem, such as the source term and the jump conditions. The results presented here were obtained with the 4th order implementation of the method discussed in §2.

The accuracy of the solution procedure will be limited by the least accurate of the factors listed above. However, in principle each of these factors can be made as accurate as one needs to; there is no inherent limit to the order of accuracy one can achieve.

When it comes to computational cost, the solution procedure proposed here is also effective. The analysis below is restricted to 2D applications, but it can be extended to any number of dimensions. Consider that the interfaces and boundaries are discretized with a total of k nodes. Furthermore, the computational grid we use in the finite differences steps contains a total of $M = N_x \times N_y$ nodes³ (N_x nodes in the x direction and N_y nodes in the y direction). Then, the cost of the solution procedure can be broken down as follows.

1. **Cost of the BIM.** The cost of the BIM depends on the specific choice of method. In principle, a general BIM requires $\mathcal{O}(k^3)$ operations. However, there is a number of techniques that can be used to reduce this cost to $\mathcal{O}(k^2)$ or even to $\mathcal{O}(k)$ [90–92].
2. **Cost of computing boundary conditions.** In equations (4.4d) and (4.8d) the boundary conditions involve integrals. The integration cost for each node on the boundary is $\mathcal{O}(k)$, such that the total cost of evaluating the boundary conditions is $\mathcal{O}(kM^{1/2})$.

³Once again, note that the discretization of interfaces and boundaries used in the BIM is completely independent of the computational grid used in the finite differences steps.

3. **Cost of interpolation.** Depending on the selected technique, the cost of computing interpolants for the potential distributions over the interfaces and boundaries varies between $\mathcal{O}(k)$ and $\mathcal{O}(k^2)$. After the interpolants are known, the cost of evaluating the potentials at the locations needed by the CFM is $\mathcal{O}(M^{1/2})$.
4. **Cost of the CFM.** Computing the correction function requires the solution of a small linear system (12×12 for 4th order of accuracy in 2D) for each grid node close to the interface or boundary. Hence, this cost is $\mathcal{O}(M^{1/2})$. Furthermore, these linear systems depend only on the geometry of the problem. Hence, even though the CFM is used more than once for each solution, the added cost is basically the same as computing the correction function only once.
5. **Cost of finite differences.** Since all problems that involve finite differences are defined in a rectangular domain, the resulting linear systems can be inverted using FFT. This is one of the fastest methods available to solve the Poisson equation, with a cost of $\mathcal{O}(M \log M)$ operations.

Since the BIM converges rapidly for smooth geometries and data, in many applications we can take $k = \mathcal{O}(M^{1/2})$. In these cases, the cost of the method is somewhere between $\mathcal{O}(M)$ and $\mathcal{O}(M^{3/2})$. In more general situations, $k = \mathcal{O}(M)$ normally suffices to obtain good accuracy. In these situations, the cost of the method varies between $\mathcal{O}(M^{3/2})$ and $\mathcal{O}(M^2)$.

4.2 Results

This section shows two examples of computations in 2D using the procedure described in §4.1. In the first example the Poisson equation is solved in an arbitrarily shaped domain with both Dirichlet and Neumann boundary conditions. The second example involves the Poisson equation with piece-wise constant coefficients. Results are shown for large coefficient ratios (1:1,000,000), including the poorly conditioned problem discussed in §4.1.3 and appendix D.

For completeness, the interfaces and boundaries immersed into regular Cartesian grids are represented using the gradient-augmented level-set method [42]. This method results in 4th order accurate representations of these curves using local grid information. Furthermore, the boundary integral equations are solved with Nystrom’s method and the trapezoidal quadrature rule [31, 89, 90]. For smooth interfaces and data, this method results in optimal convergence and accuracy. The resulting potentials are then interpolated using trigonometric interpolations [94] computed via FFT. In addition, the Poisson equations defined in the rectangular domains are discretized to 4th order accuracy using the standard 9-point stencil – see appendix A. To maintain the overall accuracy of the method, the correction function is computed to 4th order accuracy using the implementation of the CFM detailed in §2.

4.2.1 Example 1. Imposing boundary conditions over arbitrarily shaped surfaces

Consider the Poisson equation associated with the exact solution $u(x, y) = \cos(x) \sin(y)$. The solution domain Ω is the region contained within the surface defined by the zero level of ϕ :

$$\phi(x, y) = r^2(x, y) - r_0^2(\theta(x, y)), \quad (4.12a)$$

$$r(x, y) = \sqrt{x^2 + y^2}, \quad (4.12b)$$

$$r_0(x, y) = 1 + 0.3 \sin(5\theta(x, y)), \quad (4.12c)$$

$$\theta(x, y) = \tan^{-1} \left(\frac{y}{x} \right). \quad (4.12d)$$

Figure 4-2(a) shows the solution domain Ω immersed in a regular Cartesian domain. The black line in figure 4-2(a) represents the zero contour of ϕ , which is the boundary of Ω .

The same problem is solved by imposing (i) Dirichlet and (ii) Neumann boundary conditions. Given a suitable arbitrary constant for the Neumann problem, both solutions are very similar and visually indistinguishable. Figure 4-2(b) shows a plot

of the solution obtained with Dirichlet boundary conditions in a refined grid. The solution outside Ω is simply set to zero.

Nystrom's method converges rapidly in this problem, but a reasonable number of points is needed to guarantee accuracy when the results are interpolated. In principle, the discretization of $\partial\Omega$ can be refined as the Cartesian grid is refined to guarantee adequate accuracy. However, for simplicity, in this example $\partial\Omega$ was discretized only once with 500 equally spaced nodes, which results in an accuracy compatible with even the most refined Cartesian grid considered here.

Figure 4-3 shows the convergence of the error in the L_∞ and L_2 norms. The L_∞ norm behaves quite similarly for both problems, and converges to 4th order. The L_2 norm is smaller in the Dirichlet problem, but the convergence is still 4th order for both cases.

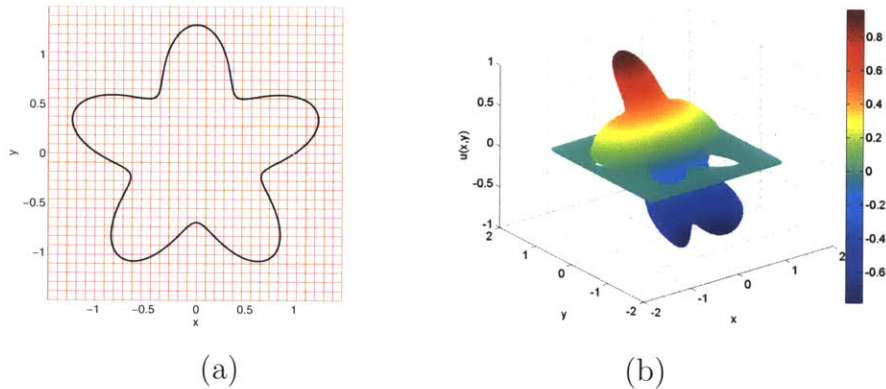


Figure 4-2: (a) Solution domain embedded in a 33×33 Cartesian Grid. (b) Solution obtained with a 193×193 grid.

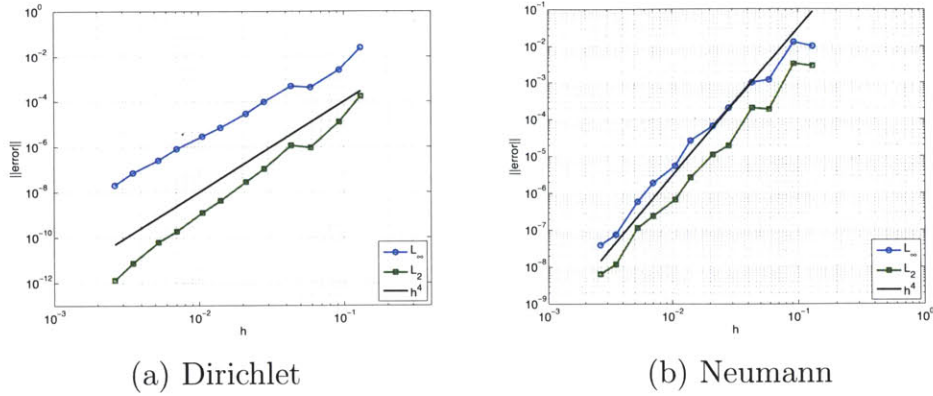


Figure 4-3: Error convergence in L_2 and L_∞ norms.

4.2.2 Example 2. Poisson equation with piece-wise constant coefficients

Consider the Poisson equation with piece-wise constant coefficients associated with the exact solution

$$u^+(x, y) = x^2 + y^2,$$

$$u^-(x, y) = \cos(x) \sin(y) + 2.$$

The solution domain is the area enclosed by the unit circle, which is represented by the zero level of ϕ_Ω :

$$\phi_\Omega(x, y) = r^2(x, y) - 1, \tag{4.13}$$

where $r(x, y)$ and $\theta(x, y)$ are defined by (4.12b) and (4.12d), respectively. In addition, the interface that divides Ω^+ and Ω^- is defined by the zero level of ϕ_Γ :

$$\phi_\Gamma(x, y) = r^2(x, y) - r_0^2(x, y), \tag{4.14a}$$

$$r_0(x, y) = 0.5 + 0.1 \sin(5\theta(x, y)). \tag{4.14b}$$

Figure 4-4(a) shows the solution domain Ω immersed in a regular Cartesian domain. The black lines in figure 4-4(a) represent the zero contours of $\phi_{\partial\Omega}$ and ϕ_Γ . Finally,

Dirichlet boundary conditions are imposed on $\partial\Omega$.

In this example, two situations were considered:

(i) $\beta^+ = 1 \times 10^6$, $\beta^- = 1$.

(ii) $\beta^+ = 1$, $\beta^- = 1 \times 10^6$.

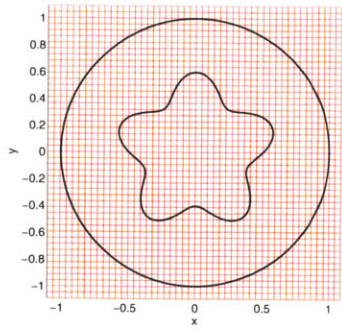
As noted in §4.1.3 and appendix D, case (ii) results in a poorly conditioned problem. Appendix D presents a remedy to this situation. For this purpose, an integral quantity of the solution, which depends only on the problem's data, must be given as an additional input. In this example, the additional data given to the code was the mean value of ρ , which is⁴

$$\bar{\rho} = 0.837801918284980.$$

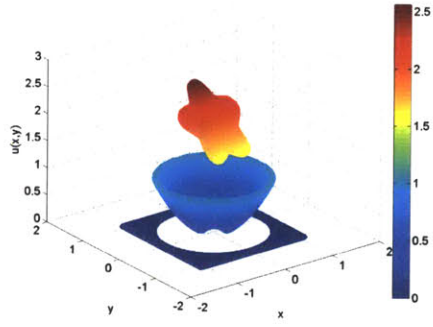
In addition, in this example $\partial\Omega$ was discretized with 200 nodes, while Γ was discretized with 400 nodes.

The solution of (i) and the corrected solution of (ii) are similar and visually indistinguishable. Figure 4-4(b) shows a plot of the solution of (i) in a refined grid. Furthermore, figure 4-5 shows the convergence of the error in the L_∞ and L_2 norms for both situations. Figure 4-5 includes results with and without the correction suggested in appendix D (corrected results are designated by “c”). As expected, the errors converge to 3rd order in both cases. Moreover, the solution of (i) and the corrected solution of (ii) present similarly small errors. On the other hand, the error in the uncorrected solution of (ii) is approximately 1×10^6 bigger than in the corrected version. This number is directly related to the condition number of the boundary integral equation we must solve, which in turn is related to the ratio between coefficients.

⁴The derivation of an expression for $\bar{\rho}$ in terms of the problem's data is shown in appendix D.

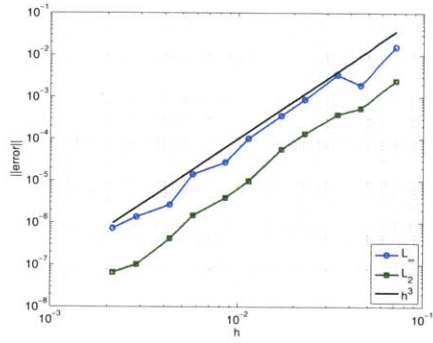


(a)

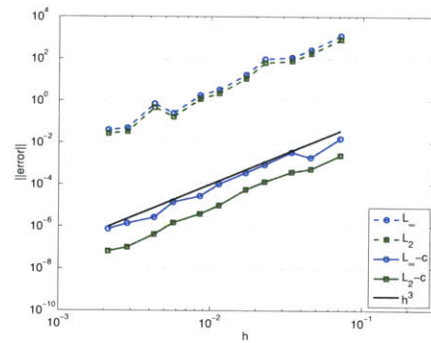


(b)

Figure 4-4: (a) Solution domain embedded in a 33×33 Cartesian Grid. (b) Solution obtained with a 193×193 grid.



(i) $\beta^+ > \beta^-$



(ii) $\beta^- > \beta^+$

Figure 4-5: Error convergence in L_2 and L_∞ norms.

Chapter 5

Dynamic Problems

In this chapter the issue of imposing boundary conditions on complex geometries in dynamic problems is addressed. Specifically, two equations that are relevant in applications are considered: the heat equation, discussed in §5.1, and the convection-diffusion equation, discussed in §5.2.

The method presented in this chapter is based on the same ideas as the version of the correction function method for the Poisson equation introduced in §3. Namely, the *correction function* is defined by a smooth extension of the solution valid in a narrow band surrounding the boundary. This *correction function* is the solution to a partial differential equation that is coupled to the underlying solution to the dynamic equation. The major difference in this extension of the CFM to dynamic problems is the fact that the correction function is now time dependent.

It is possible that one can devise methods to solve dynamic problems that are based on boundary integral equations, such as the method presented in §4 for Poisson's equation. However, this line of research was not pursued for this thesis. Furthermore, the fluid flow applications discussed in this thesis do not involve interface problems. Hence, this chapter is focused on the implementation of the CFM to impose boundary conditions on complex geometries. Extensions of the CFM to interface problems are also possible and are analogous to the method discussed here.

5.1 Heat equation

5.1.1 Overview

This section discusses the solution of the heat equation in an arbitrarily shaped solution domain Ω – see figure 3-1. In general, the shape of Ω may be a function of time, but in this thesis only problems with stationary boundaries were considered. Furthermore, the discussion below is focused on Dirichlet boundary conditions, but the same basic method can be used to other types of boundary conditions, such as Neumann. Then, the problem to be solved is

$$u_t(\vec{x}, t) - \nu \Delta u(\vec{x}, t) = f(\vec{x}, t) \quad \text{for } \vec{x} \in \Omega, \quad t > 0, \quad (5.1a)$$

$$u(\vec{x}, 0) = u_0(\vec{x}) \quad \text{for } \vec{x} \in \Omega, \quad (5.1b)$$

$$u(\vec{x}, t) = g_D(\vec{x}, t) \quad \text{for } \vec{x} \in \partial\Omega, \quad t > 0, \quad (5.1c)$$

where $\nu > 0$ is a diffusivity coefficient.

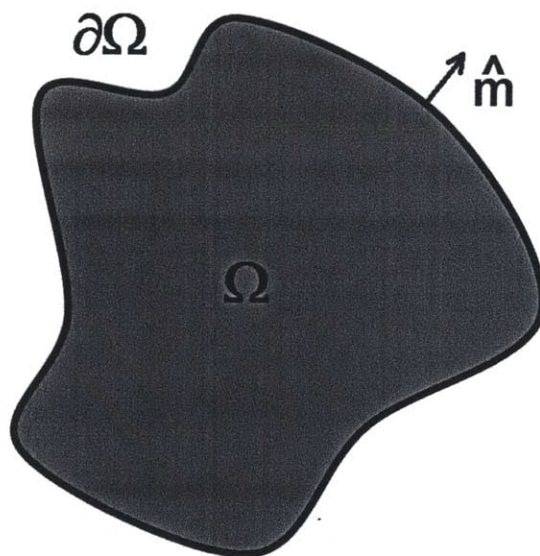


Figure 3-1: Example of solution domain with arbitrary shape.

Equation (5.1) is discretized with a combination of a Crank-Nicholson scheme in time and a compact 9-point discretization in space, as described in §5.1.2. Whenever the resulting discretization stencil straddles the boundary, the correction function is used to complete the discretization, as described in §5.1.3. The result is an implicit scheme that is 4th order accurate in space and 2nd order accurate in time. This scheme is verified in numerical experiments shown in §5.1.4.

Remark 5.1. *To take full advantage of the 4th order accuracy of the discretization in space, and of the CFM to impose boundary conditions, we need to use time steps that are $\mathcal{O}(h^2)$, where h represents the grid spacing. In principle, an implicit discretization in time is not needed if we are to use such small time steps. However, the focus of this thesis is on the implementation of the boundary condition with the correction function method, which is well illustrated by the scheme discussed here. The application of this method to other more accurate discretizations in time does not affect the implementation of the CFM.*

Furthermore, in fluid flow applications, devising time-splitting schemes that are both implicit and high order accurate is not straightforward. This subject still is an active field of research that goes beyond the scope of this thesis. Hence, a simple 2nd order accurate Crank-Nicholson is used scheme in all dynamic applications discussed in this thesis as representative of implicit time-discretization schemes. ♣

5.1.2 A compact discretization

Equation (5.1) is discretized in space using a compact and 4th order accurate 9-point stencil. The motivation behind this discretization stencil is similar to the 9-point stencil used for the Poisson equation, discussed in appendix A. The basic idea is to (i) start with the standard, 2nd order accurate, 5-point stencil discretization of the Laplace operator, and (ii) use derivatives of the heat equation to knock down the leading order errors. Namely, the 2nd order errors of the 5-point stencil discretization are proportional to the fourth derivatives of u : u_{xxxx} and u_{yyyy} . The second derivatives

of the heat equation are

$$u_{xxxx} = \frac{1}{\nu}(u_{xxt} - f_{xx}) - u_{xxyy}, \quad (5.2a)$$

$$u_{yyyy} = \frac{1}{\nu}(u_{yyt} - f_{yy}) - u_{xxyy}. \quad (5.2b)$$

These expressions can be approximated to 2nd order accuracy using standard centered differences. Then, replacing these approximations into the 5-point stencil results in the following discretization in space

$$\hat{T}(u_t)_{i,j} - \nu \hat{S}u_{i,j} = f_{i,j} + \frac{1}{12}(h_x^2(f_{xx})_{i,j} + h_y^2(f_{xx})_{i,j}), \quad (5.3)$$

$$\hat{T} = 1 + \frac{1}{12}(h_x^2 \hat{\partial}_{xx} + h_y^2 \hat{\partial}_{yy}), \quad (5.4)$$

$$\hat{S} = \hat{\partial}_{xx} + \hat{\partial}_{yy} + \frac{1}{12}(h_x^2 + h_y^2) \hat{\partial}_{xx} \hat{\partial}_{yy}, \quad (5.5)$$

where $\hat{\partial}_{xx}$ and $\hat{\partial}_{yy}$ are the centered difference operators (A.2) and (A.3), respectively. The derivatives of the source term, f_{xx} and f_{yy} , can be given analytically if known, or computed using standard 2nd order accurate finite differences.

Furthermore, (5.1) is discretized in time with the Crank-Nicholson scheme, which is implicit and 2nd order accurate. Hence, the fully discretized version of the heat equation becomes

$$\left(\frac{1}{k}\hat{T} - \frac{\nu}{2}\hat{S}\right)u_{i,j}^{n+1} = \left(\frac{1}{k}\hat{T} + \frac{\nu}{2}\hat{S}\right)u_{i,j}^n + f_{i,j}^{n+1/2} + \frac{1}{12}(h_x^2(f_{xx})_{i,j}^{n+1/2} + h_y^2(f_{xx})_{i,j}^{n+1/2}), \quad (5.6)$$

where k is the time step, and $(.)^n$ denotes a quantity evaluated at time $t = nk$.

Away from the boundary, (5.6) provides a compact discretization of the heat equation that is 2nd order accurate in time and 4th order accurate in space. In the vicinity of the boundary, we must compute the corresponding correction function, as described in §5.1.3, and use this information to complete (5.6).

5.1.3 The Correction Function Method

First we define the narrow band Ω_Γ where the correction function exists. Ω_Γ is the set of all points within a distance \mathcal{R} from $\partial\Omega$ in the time interval $nk < t < (n+1)k$, where \mathcal{R} is a value of the order of the grid size. Next, assume that the solution u can be extended smoothly into Ω_Γ . Then, the correction function, D , is defined as being equal to this smooth extension of the solution within Ω_Γ . Thus, the correction function is characterized as the solution to the following PDE.

$$D_t(\vec{x}, t) - \nu\Delta D(\vec{x}, t) = f(\vec{x}, t) \quad \text{for } \vec{x} \in \Omega_\Gamma, \quad nk < t < (n+1)k, \quad (5.7a)$$

$$D(\vec{x}, t) = g_D(\vec{x}, t) \quad \text{for } \vec{x} \in \partial\Omega, \quad nk < t < (n+1)k, \quad (5.7b)$$

$$D(\vec{x}_k, nk) = u_k^n \quad \text{for } k \in \mathcal{N}, \quad (5.7c)$$

$$D(\vec{x}_k, (n+1)k) = u_k^{n+1} \quad \text{for } k \in \mathcal{N}, \quad (5.7d)$$

where \mathcal{N} is a pre-determined set of grid nodes. In principle, this set of grid nodes can vary with time, especially to accommodate changes in the shape of $\partial\Omega$. However, since here only stationary boundaries are considered, \mathcal{N} is kept fixed at all time.

Remark 5.2. *The lack of an initial condition in (5.7) restricts the minimum width of Ω_Γ in time. This restriction occurs because information about the solution propagates away from the boundary with effective speed $\mathcal{O}(\nu/h)$. Hence, to determine the solution at a distance \mathcal{R} at time $t = k$ the time step must be at least $\mathcal{O}(\mathcal{R}h/\nu)$. In the particular scheme implemented here, $\mathcal{R} \approx \sqrt{2}h$, so $k > \sqrt{2}h^2/\nu$. In principle, one can alleviate this restriction by making Ω_Γ span more than one time step. Nevertheless, if one wants to use small time steps, there is probably no need to use an implicit discretization in time.* ♣

Remark 5.3. *Similar to the approach used to determine the correction function in other applications, Ω_Γ is sub-divided into a series of rectangular regions $\Omega_\Gamma^{(i,j)}$, one for each stencil that straddles the boundary. The definition of $\Omega_\Gamma^{(i,j)}$, and the local sets of grid nodes $\mathcal{N}^{(i,j)}$, is the same presented in §3.1.3, except for the fact that in the present case these regions extend over the time dimension with width k .* ♣

Finally, we solve for D within each $\Omega_\Gamma^{(i,j)}$ in a least squares sense. Namely, the solution corresponds to the minimum of the functional

$$\begin{aligned}
J_h &= \frac{\ell_c^{(i,j)^4}}{2\nu V(\Omega_\Gamma^{(i,j)})} \int_{nk}^{(n+1)k} dt \int_{\Omega_\Gamma^{(i,j)}} \{D_t - \nu \Delta D(\vec{x}, t) - f(\vec{x}, t)\}^2 dV \\
&+ c_P \frac{1}{2L(\Gamma)} \int_{nk}^{(n+1)k} dt \int_{\Gamma \cap \Omega_\Gamma^{(i,j)}} \{D(\vec{x}, t) - g_D(\vec{x}, t)\}^2 dS \\
&+ c_N \frac{1}{2} \sum_{k \in \mathcal{N}^{(i,j)}} \{D(\vec{x}_k, nk) - u_k^n\}^2, \\
&+ c_N \frac{1}{2} \sum_{k \in \mathcal{N}^{(i,j)}} \{D(\vec{x}_k, (n+1)k) - u_k^{n+1}\}^2.
\end{aligned} \tag{5.8}$$

In (5.8), c_P is the penalization coefficient used to impose the boundary condition, and c_N is the penalization used to enforce the constraints (5.7c) and (5.7d). The minimization of J_h , and therefore solving for D within each domain $\Omega_\Gamma^{(i,j)}$, involves the steps described below.

1. Choose a set of basis functions to represent D within $\Omega_\Gamma^{(i,j)}$: $D(\vec{x}, t) = \sum_{\ell=1}^{n_b} d_\ell \phi_\ell(\vec{x}, t)$.
2. Replace this representation of D into the functional J_h .
3. Approximate the integrals in (5.8) using numerical quadrature.
4. Solve for the weights d_ℓ that minimize J_h .

In the particular scheme implemented in this thesis, the correction function is represented by a bicubic interpolation in space and linear interpolation in time, with a total of $n_b = 24$ basis functions. This representation is coherent with the discretization described in §5.1.2, which is 4th order accurate in space and 2nd order accurate in time. Moreover, the time integrals in (5.8) are discretized using two Gaussian quadrature nodes. The remainder of the solution procedure is completely analogous to what is described in §3.1.4.

Remark 5.4. *The definition of the correction function for the heat equation with Neumann boundary condition is analogous to the Dirichlet problem described here.*

However, as noted in §3.1.5, the use of bicubic interpolation for the representation of D in space allows us to impose the Neumann boundary condition with only 3rd order accuracy. Hence, this implementation of the correction function method is restricted to 3rd order of accuracy for problems involving Neumann boundary condition. ♣

5.1.4 Results

In this section, the numerical scheme introduced in §5.1 is used to solve the heat equation in 2D with Dirichlet boundary conditions. This example corresponds to the heat equation associated with the exact solution

$$u(x, y, t) = \exp(-(x^2 + y^2)) \cos(2\pi t),$$

with $\nu = 1$. The boundary of the solution domain is represented as the zero contour of the level set function

$$\begin{aligned} \phi(x, y) &= r^2(x, y) - r^2(\theta(x, y)), \\ r(x, y) &= \sqrt{(x - x_0)^2 + (y - y_0)^2}, \\ r(\theta) &= r_0 + \epsilon \sin(5\theta), \\ \theta(x, y) &= \arctan\left(\frac{y - y_0}{x - x_0}\right), \end{aligned}$$

where, $x_0 = y_0 = 0$, $r_0 = 0.5$, and $\epsilon = 0.2$. Figure 5-1(a) shows the boundary immersed in a Cartesian grid. The solution domain Ω is the region contained inside this curve.

The time step used in this example is $k = 16h^2$. Figure 5-2 shows the numerical solution obtained with a fine grid (193×193 nodes) at different times. The solution outside Ω is simply set to zero. In addition, the error was measured at $t = 0.5$. Figure 5-1(b) shows a plot of the convergence of the error in the L_2 and L_∞ norms. As expected, the error converges to 4th order in both norms. The error in the solution and its gradient was also measured along the boundary. Figure 5-3 shows that the error along the boundary converges to the expected 4th order, while the gradient

converges to 3rd order.

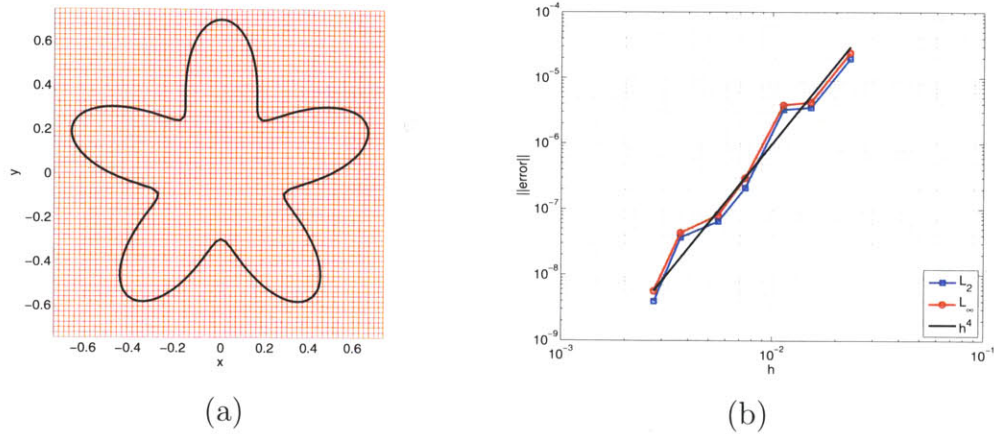


Figure 5-1: (a) Solution domain embedded in a 65×65 Cartesian grid. (b) Convergence of the error in the L_2 and L_∞ norms.

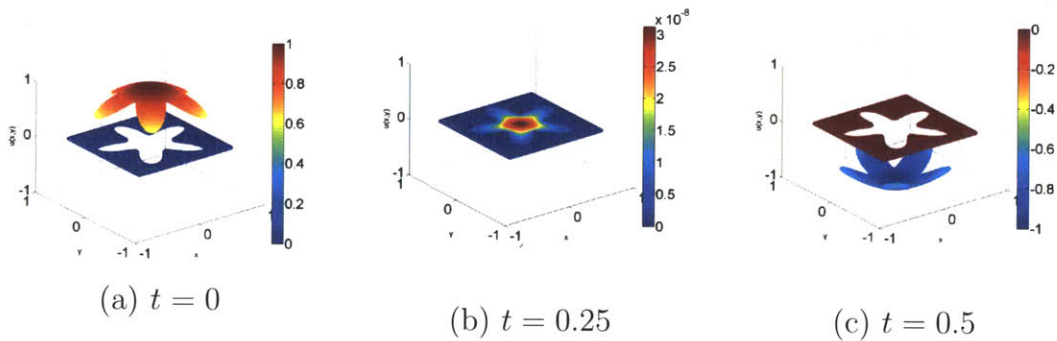


Figure 5-2: Solution at different times.

5.2 Convection-Diffusion equation

5.2.1 Overview

This section is dedicated to the solution of the convection-diffusion equation in an arbitrarily shaped solution domain Ω – see figure 3-1. In principle, the method described here can be applied to any number of dimensions, but in this thesis only the 2D convection-diffusion equation is considered. Furthermore, in the situation studied

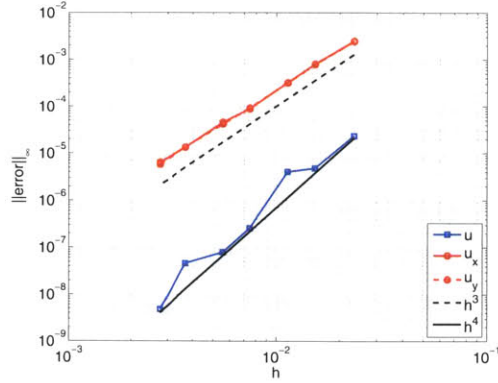


Figure 5-3: Convergence of the error along the boundary.

here the convective velocity is a function of the quantity being transported with the flow, resulting in the nonlinear convection-diffusion equation. In particular, in the equation of conservation of linear momentum the transported quantity is the velocity itself.

In addition, the boundary $\partial\Omega$ is considered to be stationary, and Dirichlet boundary conditions are applied on $\partial\Omega$. Then, we seek the solution to

$$\vec{w}_t(\vec{x}, t) + (\vec{w} \cdot \vec{\nabla})\vec{w} - \nu\Delta\vec{w}(\vec{x}, t) = f(\vec{x}, t) \quad \text{for } \vec{x} \in \Omega, \quad t > 0, \quad (5.9a)$$

$$\vec{w}(\vec{x}, 0) = \vec{w}_0(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega, \quad (5.9b)$$

$$\vec{w}(\vec{x}, t) = \vec{g}_D(\vec{x}, t) \quad \text{for } \vec{x} \in \partial\Omega, \quad t > 0, \quad (5.9c)$$

where $\nu > 0$ is a diffusivity coefficient, and \vec{w} is the convective velocity. The correction function method that solves this problem is similar to the one implemented for the heat equation in §5.1. However, there are two very important differences worth noting:

(a) Expression (5.9) is a vector equation. In 2D it involves two scalar variables:

$$\vec{w} = \{u, v\}.$$

(b) Equation (5.9) is nonlinear because of the convection term $(\vec{w} \cdot \vec{\nabla})\vec{w}$.

In theory, the nonlinear convection term couples both scalar components u and v . Hence, a fully implicit numerical scheme has to solve both components simultaneously

in a large and nonlinear system of equations, which can be a very costly process. For this reason, the most common approach to solve (5.9) is to treat the convection term explicitly, whereas still solving for the diffusive terms in an implicit fashion. The time step constraint added by treating the convection term explicitly is $k < \mathcal{O}(h)$, which is not an important restriction in most practical applications.

The convective term models the propagation of information along the characteristic directions. Hence, an explicit discretization must respect this flow of information (must be upwind), else instabilities arise. There exist methods that can achieve high order of accuracy and are widely used in this context, such as the ENO and WENO schemes [95–97]. However, to obtain high order of accuracy, these schemes involve relatively wide discretization stencils, which is not very attractive from the point of view of the CFM. For this reason, a different approach was adopted here.

Instead of treating the complete convective term in an explicit fashion, only the convective velocity is treated explicitly, while the remainder of the convective term is kept in an implicit form. In other words, the convective term is approximated by

$$(\vec{w} \cdot \vec{\nabla})\vec{w} \approx (\vec{w}^* \cdot \vec{\nabla})\vec{w}, \quad (5.10)$$

where \vec{w}^* is the explicit convective velocity obtained by extrapolation of the solution from previous time steps. Naturally, the quality of this approximation depends on the accuracy of the extrapolation used to estimate \vec{w}^* . The objective of this approach is to be able to use a compact centered differences discretization and still obtain an accurate and stable numerical scheme. For all practical purposes, no restriction to the maximum time step allowed was observed with the discretization scheme described in §5.2.2.

Therefore, (5.9) is solved with a combination of a Crank-Nicholson discretization in time and a compact 9-point discretization in space, as described in §5.2.2. Whenever this discretization stencil straddles the boundary, the correction function is used to complete the discretization. The result is a semi-implicit¹ scheme that is 4th order

¹The discretization is implicit with the exception of (5.10).

accurate in space and 2nd order accurate in time. Finally, §5.2.3 shows some results.

Remark 5.5. *Once the convective term is replaced by the semi-implicit version (5.10), the 2D convection-diffusion equation becomes a system of two linearized equations that can be solved independently for the components u and v . The same procedure is applied to the PDE that defines the correction function, so there is one correction function for each component, independently of the other. In this context, the definition of the correction function is completely analogous to the procedure applied to the heat equation in §5.1.3. For this reason, it will not be discussed in further detail. ♣*

5.2.2 A compact discretization

As discussed above, the convective velocity is replaced by an explicit approximation – see equation (5.10). In particular, here the convective velocity is approximated with a linear extrapolation of the solution in the previous two time steps:

$$\bar{w}^* = \frac{3\bar{w}^n - \bar{w}^{n-1}}{2}. \quad (5.11)$$

By doing so, the equations for the scalar components u and v decouple from each other. Hence, we can solve for each component individually. Furthermore, both components are subject to the same differential operators. Therefore, we can apply the same discretization for both equations.

Equation (5.9) is discretized in space using a compact and 4th order accurate 9-point stencil. To derive this stencil we start with standard, 2nd order accurate, centered differences applied to both the convective and diffusive terms. The error resulting from this discretization is

$$\frac{h_x^2}{6}u^*u_{xxx} + \frac{h_y^2}{6}v^*u_{yyy} - \frac{\nu}{12}(h_x^2u_{xxxx} + h_x^2u_{yyyy}) + \mathcal{O}(h^4), \quad (5.12)$$

where $\vec{w}^* = \{u^*, v^*\}$. The idea is to use derivatives of the equation to estimate this error to 2nd order, leaving only 4th order errors behind. The second derivatives of the

convection-diffusion equation results in

$$u^* u_{xxx} - \nu u_{xxxx} = f_{xx} - u_{xxt} + \nu u_{xxyy} - u_{xx}^* u_x - 2u_x^* u_{xx} - (v^* u_y)_{xx}, \quad (5.13a)$$

$$v^* u_{yyy} - \nu u_{yyyy} = f_{yy} - u_{yyt} + \nu u_{xyyy} - v_{yy}^* u_y - 2v_y^* u_{yy} - (u^* u_x)_{yy}, \quad (5.13b)$$

All terms in the right-hand-side of (5.13) can be discretized to 2nd order of accuracy using standard centered differences. However, replacing (5.13) into (5.12) does not eliminate all of leading order error. The remaining error is given by

$$-\frac{\nu}{6}(h_x^2 u_{xxxx} + h_x^2 u_{yyyy}) + \mathcal{O}(h^4). \quad (5.14)$$

In principle, we could use different manipulations of the equation and its derivatives to eliminate these errors as well. However, this procedure leads to errors that are $\mathcal{O}(h^4/\nu)$, which is not satisfactory for small ν (or large Reynolds number in the Navier-Stokes equations). Hence, instead the error (5.14) is eliminated by using the same expression as in (5.13), but replacing the convective term with a fully explicit version:

$$\nu u_{xxxx} = ((\vec{w}^* \cdot \vec{\nabla}) u^*)_{xx} - f_{xx} + u_{xxt} - \nu u_{xxyy}, \quad (5.15a)$$

$$\nu u_{yyyy} = ((\vec{w}^* \cdot \vec{\nabla}) u^*)_{yy} - f_{yy} + u_{yyt} - \nu u_{xyyy}. \quad (5.15b)$$

Note that this last expression is only used to eliminate the errors left in (5.14). Hence, it is reasonable to expect it does not influence the stability of the scheme. Finally, the complete discretization in space becomes

$$\begin{aligned} \hat{T}(u_t)_{i,j} + \hat{S}_c u_{i,j} - \nu \hat{S}_d u_{i,j} &= f_{i,j} + \frac{1}{12}(h_x^2 (f_{xx})_{i,j} + h_y^2 (f_{yy})_{i,j}) \\ &+ \frac{1}{12}(h_x^2 \hat{\partial}_{xx}(\vec{w}^* \cdot \vec{\nabla}) u_{i,j}^* + h_y^2 \hat{\partial}_{yy}(\vec{w}^* \cdot \vec{\nabla}) u_{i,j}^*), \end{aligned} \quad (5.16)$$

where

$$\hat{T} = 1 + \frac{1}{12}(h_x^2 \hat{\partial}_{xx} + h_y^2 \hat{\partial}_{yy}), \quad (5.17)$$

$$\hat{S}_c = u^* \hat{\partial}_x + v^* \hat{\partial}_y + \frac{1}{6}(h_x^2 \hat{S}_{cx} + h_y^2 \hat{S}_{cy}) \quad (5.18)$$

$$\begin{aligned} \hat{S}_{cx} &= v^* \hat{\partial}_{xx} \hat{\partial}_y + (\hat{\partial}_{xx} u^*) \hat{\partial}_x + (\hat{\partial}_{xx} v^*) \hat{\partial}_y \\ &\quad + 2(\hat{\partial}_x u^*) \hat{\partial}_{xx} + 2(\hat{\partial}_x v^*) \hat{\partial}_x \hat{\partial}_y, \end{aligned} \quad (5.19)$$

$$\begin{aligned} \hat{S}_{cy} &= u^* \hat{\partial}_x \hat{\partial}_{yy} + (\hat{\partial}_{yy} u^*) \hat{\partial}_x + (\hat{\partial}_{yy} v^*) \hat{\partial}_y \\ &\quad + 2(\hat{\partial}_y u^*) \hat{\partial}_x \hat{\partial}_y, + 2(\hat{\partial}_y v^*) \hat{\partial}_{yy} \end{aligned} \quad (5.20)$$

$$\hat{S}_d = \hat{\partial}_{xx} + \hat{\partial}_{yy} + \frac{1}{12}(h_x^2 + h_y^2) \hat{\partial}_{xx} \hat{\partial}_{yy}, \quad (5.21)$$

The derivatives of the source term, f_{xx} and f_{yy} , can be given analytically if know, or computed using standard 2nd order accurate finite differences.

The Crank-Nicholson scheme, which is implicit and 2nd order accurate, is used for the discretization in time. Hence, the final discretized version of the convection-diffusion equation becomes

$$\begin{aligned} \left(\frac{1}{k} \hat{T} + \frac{1}{2}(\hat{S}_c - \nu \hat{S}_d)\right) u_{i,j}^{n+1} &= \left(\frac{1}{k} \hat{T} - \frac{1}{2}(\hat{S}_c - \nu \hat{S}_d)\right) u_{i,j}^n \\ &\quad + f_{i,j}^{n+1/2} + \frac{1}{12}(h_x^2 (f_{xx})_{i,j}^{n+1/2} + h_y^2 (f_{yy})_{i,j}^{n+1/2}) \\ &\quad + \frac{1}{12}(h_x^2 \hat{\partial}_{xx}(\vec{w}^* \cdot \vec{\nabla}) u_{i,j}^* + h_y^2 \hat{\partial}_{yy}(\vec{w}^* \cdot \vec{\nabla}) u_{i,j}^*), \end{aligned} \quad (5.22)$$

where k denotes the time step.

Away from the boundary, (5.22) provides a compact discretization of (5.2) that is 2nd order accurate in time and 4th order accurate in space. In the vicinity of the boundary, we must compute the corresponding correction function and use this information to complete (5.22).

5.2.3 Results

In this section the numerical scheme introduced in §5.2 was used to solve the 2D convection-diffusion equation with Dirichlet boundary conditions. Unlike other examples in this thesis, here the irregular geometry where the Dirichlet boundary conditions are applied is a boundary internal to the solution domain Ω . On the external boundary periodic boundary conditions are imposed. The internal boundary is represented as the zero contour of the of the level set function

$$\begin{aligned}\phi(x, y) &= r^2(x, y) - r^2(\theta(x, y)), \\ r(x, y) &= \sqrt{(x - x_0)^2 + (y - y_0)^2}, \\ r(\theta) &= r_0 + \epsilon \sin(5\theta), \\ \theta(x, y) &= \arctan\left(\frac{y - y_0}{x - x_0}\right),\end{aligned}$$

where, $x_0 = y_0 = \pi + 0.1\sqrt{5}$, $r_0 = 1$, and $\epsilon = 0.4$. The solution domain is the area comprised in the $(0, 2\pi) \times (0, 2\pi)$ square, with the area within the zero level set excluded. Figure 5-4 shows the solution domain discretized with a Cartesian grid and the internal boundary immersed into it. This example corresponds to the convection-diffusion equation associated with the exact solution

$$\begin{aligned}u(x, y, t) &= -\sin(x)^2 \cos(y) \sin(y) \sin(t) + 1, \\ v(x, y, t) &= \cos(x) \sin(x) \sin(y)^2 \sin(t) + 1,\end{aligned}$$

with $\nu = 1$. The time step used is $k = h^2$.

Figures 5-5 and 5-6 show the numerical solutions obtained with a fine grid (193×193 nodes) at different times. The solution outside Ω is simply set to zero. Error convergence plots are shown in figure 5-7. Both the L_2 and L_∞ norms are shown. As expected, errors converge to 4th order in both norms for the two components u and v . Figure 5-8 shows the convergence of the error in the solution and its gradient evaluated along the boundary. Once again, both components u and v converge to 4th order, while the respective gradients convergence to 3rd order.

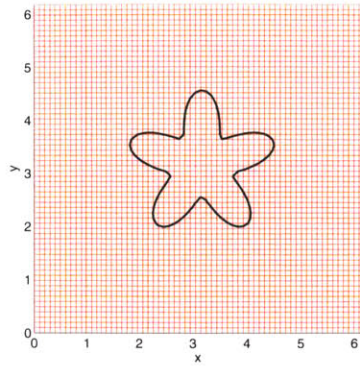


Figure 5-4: Solution domain discretized with a 65×65 Cartesian grid. The internal boundary is immersed in the grid.

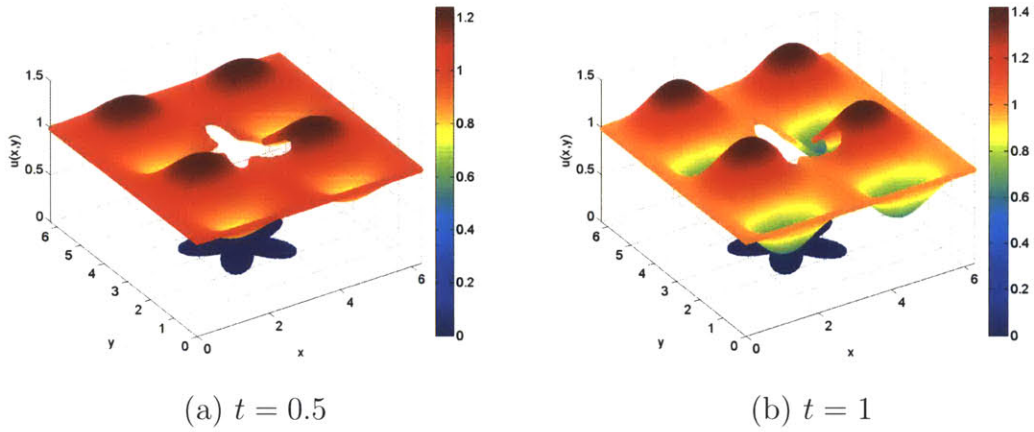


Figure 5-5: Plot of the u component of the solution at different times.

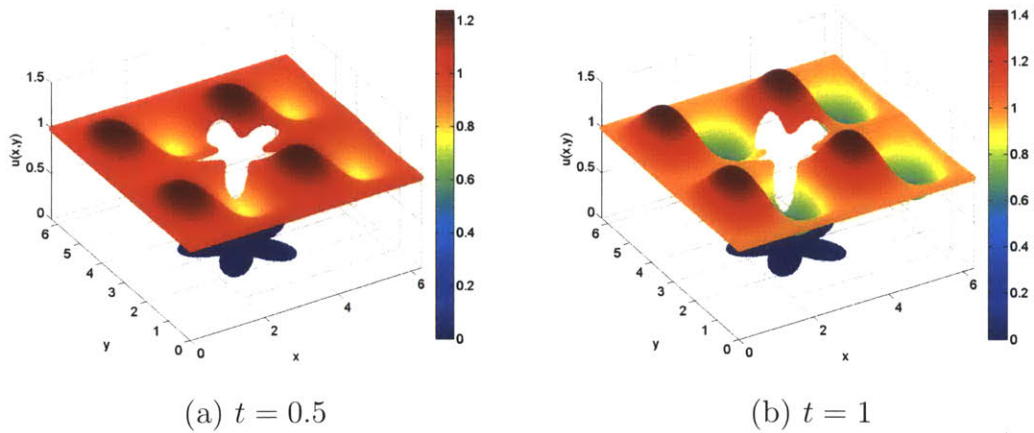
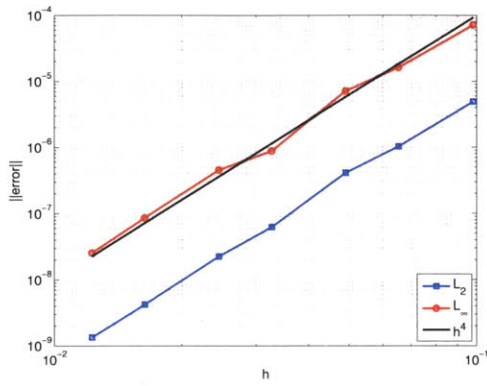
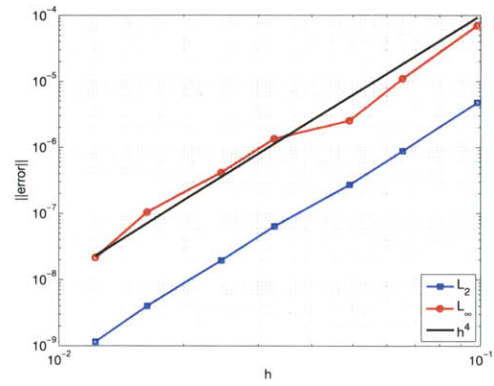


Figure 5-6: Plot of the v component of the solution at different different in time.

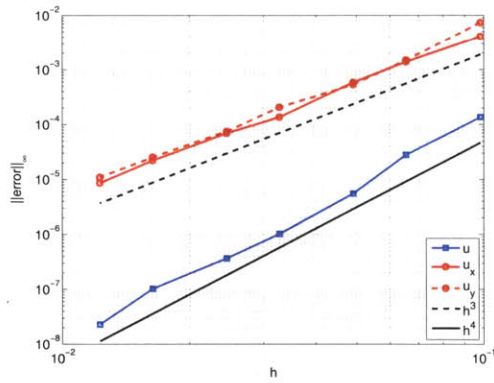


(a) u

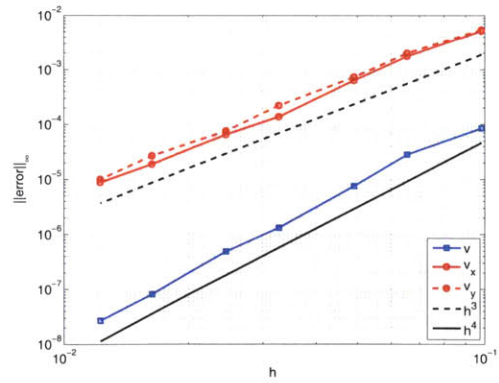


(b) v

Figure 5-7: Convergence of the error in the L_∞ and L_2 norms.



(a) u



(b) v

Figure 5-8: Convergence of the error along the boundary.

Chapter 6

Incompressible Navier-Stokes Equations

In this chapter, the techniques described in previous chapters are combined to solve the incompressible Navier-Stokes equations to high order of accuracy using immersed grids. Most common formulations of the incompressible Navier-Stokes equations (INSE) involve the solution of a convection-diffusion equation for the velocity field, and a Poisson equation for the distribution of pressure. Hence, the versions of the correction function method discussed in §2 to §5 serve as the foundation needed to solve the INSE.

Before developing a CFM to solve the INSE, we need to know what boundary conditions to use. The incompressibility condition included in the INSE can be satisfied in different forms, giving rise to different formulations of the INSE. In turn, each formulation results in a different set of boundary conditions. In §6.1 the particular formulation adopted in this thesis is presented. Next, §6.2 discusses the numerical method used to solve this formulation of the equations. Finally, §6.3 shows some results, including the flows over cylinders in the low Reynolds number regime.

6.1 Formulation

The incompressible Navier-Stokes equations (INSE) are a set of equations that describe the governing dynamics of incompressible viscous flows. By incompressible flow, it should be understood that the density of the fluids involved in the flow remain constant in each phase. These equations describe the conservation of mass¹ and linear momentum. In nondimensional form, the INSE read

$$\text{Conservation of mass:} \quad \vec{\nabla} \cdot \vec{w} = 0, \quad (6.1a)$$

$$\text{Conservation of linear momentum:} \quad \vec{w}_t + (\vec{w} \cdot \vec{\nabla})\vec{w} = -\vec{\nabla}p + \frac{1}{Re}\Delta\vec{w} + \vec{f}, \quad (6.1b)$$

where $\vec{w} = \{u, v\}$ is the velocity vector, p is the pressure, and \vec{f} represents external body forces that act upon the fluid. In addition,

$$Re = \frac{UL}{\nu} \quad (6.2)$$

is the *Reynolds number*. In (6.2), U is a characteristic speed, L is a characteristic length, and ν is the dynamic viscosity of the fluid.

For completeness, (6.1) must be accompanied by suitable initial and boundary conditions. For the flow in a given domain Ω , it is common to impose

$$\vec{w}(\vec{x}, 0) = \vec{w}_0(\vec{x}) \quad \text{for } \vec{x} \in \Omega, \quad (6.3)$$

$$\vec{w}(\vec{x}, t) = \vec{g}_D(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega, \quad t > 0. \quad (6.4)$$

The conditions (6.3) and (6.4) describe the situations studied in this thesis, including the flow over rigid bodies.

Equation (6.1a) is not an evolution equation, but a constraint on the solutions. This creates difficulties in the design of numerical schemes to solve the equations. A very common approach to solve (6.1) is to use one of the many variations of

¹Equation (6.1a) is also commonly referred to as the incompressibility or the divergence-free condition.

the projection method [19, 20, 98–100]. A feature common to all these variations is to split the solution for velocity from the solution for pressure. Pressure (or, in some variations, a variable related to pressure) is characterized as the solution to a Poisson equation. A long standing problem of the projection method is how to properly define boundary conditions for this Poisson equation. As a consequence, the projection method normally produces “numerical boundary layers:” regions close to the boundary where the error decays slower than in the rest of the domain as the computational grid is refined.

There are recent re-formulations of the INSE which re-write them as equivalent systems of evolution equations at the continuum level, and can be used to produce schemes that avoid the numerical boundary layers in projection methods [21–24, 26]. In particular, the formulations developed by Johnston and Liu [23, 24], and by Shirokoff and Rosales [26] have the following properties in common.

- (i) The split between velocity and pressure occurs at the continuum level – the dependence between these variables occurs at the source terms and boundary conditions.
- (ii) The boundary condition for pressure is derived such that (6.1a) is satisfied.

Hence, in principle, we can discretize these new formulations to any desired order of accuracy.

Nevertheless, the formulation proposed by Shirokoff and Rosales [26] results in “unconventional” boundary conditions for the velocity field. As shown in [26], the implementation of these boundary conditions is not straightforward. For this reason, here the formulation introduced by Johnston and Liu [23, 24] was implemented. Namely, the system of equations to be solved is

$$\vec{w}_t + (\vec{w} \cdot \vec{\nabla})\vec{w} - \frac{1}{Re}\Delta\vec{w} = -\vec{\nabla}p + \vec{f}, \quad (6.5a)$$

$$\Delta p = \vec{\nabla} \cdot (-(\vec{w} \cdot \vec{\nabla})\vec{w} + \vec{f} - \lambda\vec{w}), \quad (6.5b)$$

with (6.3), (6.4), and the additional boundary condition

$$p_n = \hat{n} \cdot \left(-(\vec{g}_D)_t - (\vec{w} \cdot \vec{\nabla})\vec{w} - \frac{1}{Re}(\vec{\nabla} \times \vec{\nabla} \times \vec{w}) + \vec{f} \right), \quad (6.6)$$

where \hat{n} is the unit vector normal to the boundary $\partial\Omega$, pointing outwards. Hence, this formulation involves the solution to a convection-diffusion equation with Dirichlet boundary condition for the velocity field, and a Poisson equation with Neumann boundary condition for pressure. The solution to these equations is discussed in §2 to §5. The next section explains how we can use these techniques in the context of (6.5).

Remark 6.1. *The $(-\lambda\vec{\nabla} \cdot \vec{w})$ term in (6.5b) is not part of the formulation presented by Johnston and Liu [23, 24]. This additional term serves only to stabilize the numerical discretization in the high Reynolds number regime, as explained below.*

Assume that the initial condition (6.3) automatically satisfies the divergence-free condition (6.1a). Then, following the discussion in [24], by solving (6.5) with (6.3), (6.4), and (6.6), the variable $\varphi = (\vec{\nabla} \cdot \vec{w})$ indirectly satisfies the following PDE.

$$\varphi_t = \frac{1}{Re}\Delta\varphi - \lambda\varphi \quad \text{for } \vec{x} \in \Omega, \quad (6.7a)$$

$$\varphi(\vec{x}, 0) = 0 \quad \text{for } \vec{x} \in \Omega, \quad (6.7b)$$

$$\varphi_m(\vec{x}, t) = 0 \quad \text{for } \vec{x} \in \partial\Omega, \quad t > 0. \quad (6.7c)$$

Naturally, the solution to this PDE is $\varphi = 0$ for $t > 0$. However, the discretization of (6.5) introduces numerical errors that also affect (6.7). For small Re , the effect of these numerical errors is such that the divergence-free condition is only satisfied up to the order of accuracy of the numerical scheme, which is satisfactory for all practical purposes. However, for large Re the diffusion in (6.7a) is not enough to control the discretization error and keep φ small, which then de-stabilizes the computation. In this case, the additional term with $\lambda > 0$ serves to control the discretization errors and stabilize the solution. This term is inspired by the solution adopted in [26] for a similar problem.

6.2 Numerical Scheme

As discussed in §6.1, the formulation of the INSE adopted in this thesis requires the solution of a convection-diffusion equation with Dirichlet boundary conditions for the velocity field, and a Poisson equation with Neumann boundary conditions for the pressure. Solving these problems to high order accuracy using immersed grids is discussed in §3 to §5. This section explains how we can combine these techniques to develop a version of the correction function method to solve the incompressible Navier-Stokes equations.

Note that the formulation discussed in §6.1 is only stable because it satisfies the divergence-free condition (6.1a) by indirectly enforcing (6.7). Hence, the discretization scheme must also indirectly enforce (6.7) – at least up to the desired order of accuracy. In the context of the correction function method, this condition means that the pressure correction function must depend on the velocity correction function. Specifically, when we solve the PDE that defines the pressure correction function, we must evaluate the source term in (6.5b), and the boundary condition (6.6), using the velocity correction function:

$$\Delta D_p = \vec{\nabla} \cdot \left(-(\vec{D}_w \cdot \vec{\nabla}) \vec{D}_w + \vec{f} - \lambda \vec{D}_w \right) \quad \text{for } \vec{x} \in \Omega_\Gamma, \quad (6.8a)$$

$$(D_p)_m = \hat{n} \cdot \left(-(\vec{g}_D)_t - (\vec{D}_w \cdot \vec{\nabla}) \vec{D}_w - \frac{1}{Re} (\vec{\nabla} \times \vec{\nabla} \times \vec{D}_w) + \vec{f} \right) \quad \text{for } \vec{x} \in \partial\Omega, \quad (6.8b)$$

where D_p denotes the pressure correction function, and \vec{D}_w represents the velocity correction function². In fact, this requirement makes the implementation relatively easy, since we can obtain derivatives of the correction function at any point close the boundary in a straightforward fashion. On the other hand, we must be careful to maintain the high order accuracy.

²As explained in §5.2, the correction function for each component of the velocity field is computed independently from the other. Then $\vec{D}_w = \{D_u, D_v\}$.

If each component of the velocity correction function is represented using a bicubic interpolation in space and a linear interpolation in time, as discussed in §5.2, the solution is 4th order accurate – assuming $k = \mathcal{O}(h^2)$ – but the derivatives are less accurate. In particular, (6.8) involves second derivatives of the velocity. Hence, in the case mentioned above, these conditions can only be enforced to 2nd order of accuracy. As a result, the overall accuracy of the solution scheme is reduced to 2nd order. Therefore, to obtain high order accuracy in the context of the INSE, we must use a richer representation for the correction functions. To obtain 4th order of accuracy, the correction functions are represented with biquintic interpolation in space, and quadratic interpolation in time.

On the other hand, this richer representation of the correction functions requires more information about the solution. In other words, the local sets $\mathcal{N}^{(i,j)}$ used to impose the compatibility conditions (5.7c) and (5.7d) need to include more grid nodes. In this particular case, $\mathcal{N}^{(i,j)}$ is the set of nodes that both (i) lie within the solution domain, and (ii) are part of the 5×5 set of nodes that surround node (i, j) . In addition, this choice of $\mathcal{N}^{(i,j)}$ results in larger rectangular domains $\Omega_{\Gamma}^{(i,j)}$. For this reason, $\Omega_{\Gamma}^{(i,j)}$ must also include a bigger piece of the boundary. Namely, $\Omega_{\Gamma}^{(i,j)}$ is such that it includes the piece of boundary of length $4\sqrt{h_x^2 + h_y^2}$ centered at the point on the interface closest to node (i, j) – see §5.1.3 for further detail. In addition, the richer representation in time also demands more information from past time steps. In this case, $\Omega_{\Gamma}^{(i,j)}$ extends with width $2k$ in time.

Remark 6.2. *A consequence of including more grid nodes in $\mathcal{N}^{(i,j)}$ is a wider discretization stencil in the vicinity of the boundary. This happens because when we use the correction function to complete the discretization stencil, the stencil is modified to include the grid nodes that are part of $\mathcal{N}^{(i,j)}$.*

Remark 6.3. *Although this richer representation of the correction functions results in 4th order accuracy for both velocity and pressure, with the GA-LS representation of the immersed boundary the vectors normal to the boundary are computed to 3rd order accuracy. As a consequence, the the Neumann boundary conditions for the pressure*

are only imposed to 3rd order, which reduces the accuracy of this variable.

Remark 6.4. *I use information from the past two time steps only to compute the velocity correction function. The discretization of the equations on the grid nodes is the same one described in §5.2.2: a combination of Crank-Nicholson in time and centered differences in space.*

Second, although the pressure correction function must depend on the velocity correction function to maintain stability, the opposite is not true. When we compute the source term for the PDE that defines the velocity correction function using (6.5a), we *cannot* use the pressure correction function to compute the gradient of pressure. The reason for this asymmetric coupling between velocity and pressure correction functions comes from the relationship between velocity and pressure in the INSE. In the INSE, velocity is the only independent variable. In contrast, pressure is a function of velocity only – given any velocity distribution, we can obtain a corresponding pressure distribution that “drives” it to a divergence-free field. On the other hand, the opposite is not true. Given any pressure distribution, we cannot necessarily compute a corresponding velocity field that is divergence-free. For this reason, using information from the pressure correction function when solving for the velocity correction function is similar to using the velocity correction function from a previous time step to set an initial condition. As discussed in §5.1.3, this procedure is unstable. The solution adopted here is to (i) compute the gradient of pressure away from the boundary using standard 4th order accurate finite differences, and (ii) use bicubic extrapolation to evaluate the gradient of pressure whenever necessary for the CFM.

Third, the first and second derivatives of the velocity field are needed to evaluate the source term for the pressure on the grid nodes (the actual pressure, not the correction function). These derivatives can be computed with standard 4th order accurate finite differences. For nodes that are close to the boundary, we can use the correction function to complete these finite differences stencils.

Except for the points discussed above, the remainder of the solution procedure follows the schemes introduced in §3.1 and §5.2. Below a summarized description of

the steps involved in this solution procedure is presented. For this purpose, assume that we seek the solution of the INSE at $t = nk$. Moreover, let xqt denote the quadrature nodes used for integration in time (normalized such that $0 < xqt < 1$). Then, the solution procedure involves the steps listed below.

1. Compute the convective velocity in $\Omega_{\Gamma}^{(i,j)}$ at $t = (n - 2 + 2xqt)k$ by linear extrapolation.
2. Compute the pressure gradient on the grid nodes at $t = (n - 2)k$ and $t = (n - 1)k$ using 4th order finite differences. For nodes close to the boundary, use the correction function to complete the discretization.
3. Extrapolate the pressure gradient to $\Omega_{\Gamma}^{(i,j)}$ at $t = (n - 2 + 2xqt)k$ using bicubic extrapolation in space and linear extrapolation in time.
4. Solve for the velocity correction function using the scheme described in §5.2.
5. Compute the convective velocity on the grid nodes at $t = (n + 1/2)k$ by linear extrapolation.
6. Solve the convection-diffusion equation to obtain the velocity on the grid nodes at $t = nk$. In this step we use the Crank-Nicholson scheme described in §5.2.
7. Compute the source term for the pressure correction function using the velocity correction function – see (6.8a).
8. Compute the boundary condition for the pressure correction function using the velocity correction function – see (6.8b).
9. Solve for the pressure correction function using the scheme described in §3.1.
10. Compute the source term for the Poisson’s equation – see (6.5b).
11. Solve the Poisson’s equation to obtain the pressure on the grid nodes at $t = nk$. In this step we use the scheme described in §3.1.

Remark 6.5. *In the vicinity of boundaries, the viscous term in the INSE becomes significant, even in the high Reynolds number regime. For this reason, as explained in remark 5.2, information from the boundary takes some time to propagate towards the edges of Ω_Γ . As a consequence, there is a limitation to the minimum time step we can use with the CFM. In the context of the INSE, this limit becomes $k > \sqrt{2}h^2 Re$. This limit may seem too restrictive in the high Reynolds number regime. However, a reasonable computational grid used to solve the INSE in this regime will be very refined close to the boundaries, while coarser in most of the solution domain. Ideally, the grid spacing close to the boundary will be $1/\sqrt{Re}$ the size of the grid spacing away from the boundary. Hence, in this situation this limit becomes $k > \sqrt{2}h_c^2$, where h_c is the “coarse” grid spacing, which is a reasonable time step for most practical purposes.*

Remark 6.6. *The solution procedure described here requires knowledge of the solution at the two previous time steps. Hence, in principle we must use a special scheme to handle the first time step. However, this issue is not particularly relevant to the correction function method. Therefore, for simplicity, here an additional “initial” condition is imposed at $t = -k$. For situations where the exact solution is known, this information is used to define this additional condition. Otherwise, the same initial condition is set for $t = 0$.*

6.3 Results

This section presents six examples of solutions of the 2D incompressible Navier-Stokes equations using the scheme described in §6. In the first two examples, the domain is periodic in both directions, with no immersed boundaries in the solution domain. These examples serve as validation of the formulation of the INSE adopted here, and of the numerical scheme used to discretize the equations in the interior of the solution domain.

The third example shows the solution of a similar problem, but with the addition of an internal boundary, which is immersed in the regular Cartesian grid. In this example, the correction function method used to solve the INSE is validated.

Finally, the last three examples involve flows over cylinders in the low Reynolds number regime. Unfortunately, moderate to high Reynolds number results could not be computed because the code developed for this thesis supports only uniform Cartesian grids. In the high Reynolds number regime, one must be able to adequately resolve the boundary layer with grids that are relatively fine close to the boundary. Hence, because a uniform grid is used here, the computational requirements increase rapidly as the Reynolds number becomes higher. To surpass this difficulty, one needs to implement the CFM with a numerical scheme that supports grids with variable cell sizes, without loss of accuracy. Developing such a numerical scheme in the context of finite differences is not an easy task, and lies outside the scope of this thesis.

6.3.1 Purely periodic boundaries

Before moving to problems involving immersed boundaries, it is important to validate the formulation of the INSE and the numerical scheme adopted here. For this reason, the first two experiments involve purely periodic boundary conditions: there are no boundaries immersed in the solution domain. The first experiment involves a low Reynolds number situation ($Re = 1$), whereas the second experiment is set in the high Reynolds number regime ($Re = 1 \times 10^6$). This second experiment serves to evaluate the stabilizing term added to Johnston and Liu's formulation of the INSE equations, as discussed in remark 6.1. Namely, in the first example $\lambda = 0$, and in the second experiment $\lambda = \min(1/4k, 10)^3$.

In both examples, the solution domain is the $(0, 2\pi) \times (0, 2\pi)$ square. Moreover, the problem considered here is related to the exact solution

$$u(x, y, t) = -\sin^2(x) \cos(y) \sin(y) \sin(t) + 1, \quad (6.9a)$$

$$v(x, y, t) = \cos(x) \sin(x) \sin^2(x) \sin(t) + 1, \quad (6.9b)$$

$$p(x, y, t) = \cos(x) \sin(y) \sin(t). \quad (6.9c)$$

³The λ term is an artificial correction added to control errors in the high Re regime. So, it is interesting to limit λ to a relatively small value, *e.g.* $\lambda = 10$. However, the theoretical rate of decay imposed by the stabilizing term is $\tau = 1/\lambda$. Hence, for stability reasons, λ must not be larger than $1/k$. Here, $\lambda = 1/4k$ in coarse grids, where the time step is relatively large.

The solution is evaluate up to $t = 10$ with time step $k = 2.5h^2$. Finally note that, since there is no boundary in the interior of the solution domain, it is not necessary to resolve a boundary layer. Hence, we can solve the high Reynolds number regime without using a very refined grid. Furthermore, there is no restriction to the minimum time step allowable, as occurs in problems with boundaries – see remark 6.5.

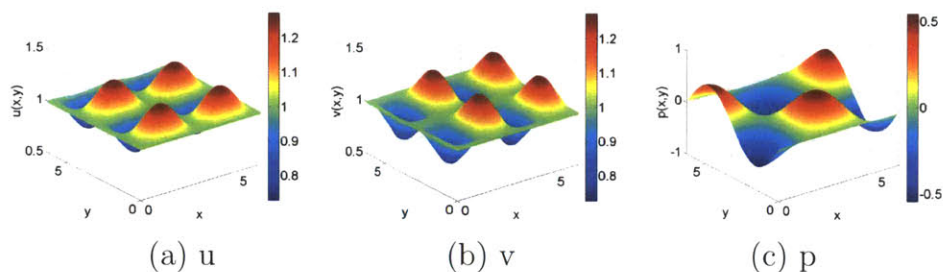


Figure 6-1: Solution at $t = 10$ for $Re = 1$.

Figure 6-1 shows the solution to example 1 at $t = 10$ in a 128×128 grid. The solution to example 2 is similar, and visually indistinguishable. Moreover, figure 6-2 presents the behavior of $\varphi = (\vec{\nabla} \cdot \vec{w})$ over time in example 2 (128×128 grid). Figure 6-2(a) shows the solution computed with $\lambda = 0$, while figure 6-2(b) shows the solution computed with $\lambda = 10$. As we can observe, without the stabilization term, φ tends to grow over time. The stabilization term controls this growth, making φ oscillate periodically with small amplitude. Therefore, we conclude that, in the absence of boundaries, the stabilization term added to the pressure equation is capable of controlling the growth of φ even for Reynolds number as high as 1×10^6 .

Finally, figure 6-3 shows the convergence of the L_∞ norm of the errors at $t = 10$. As expected, in both examples the error converges to 4th order for all variables – including φ .

6.3.2 Immersed Boundary

After validating the formulation and the basic numerical scheme used to solve the INSE, the next step is to combine it with the CFM to impose the appropriate conditions on immersed boundaries. To verify the validity of the scheme proposed here,

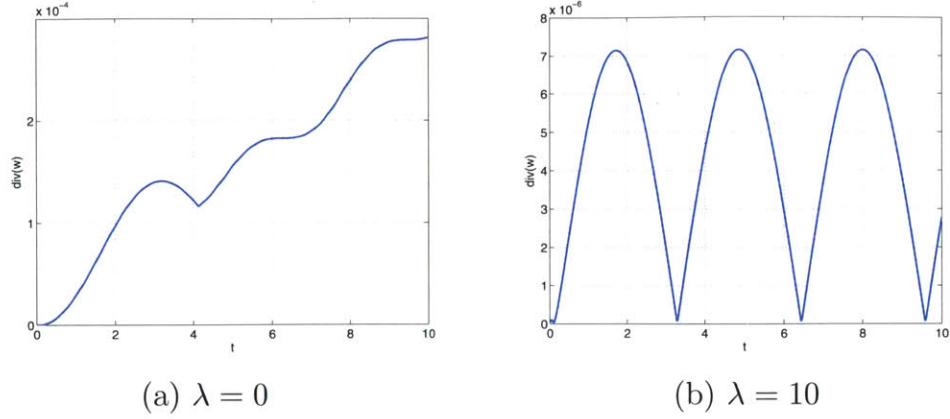


Figure 6-2: L_∞ norm of the divergence of velocity for $Re = 1 \times 10^6$.

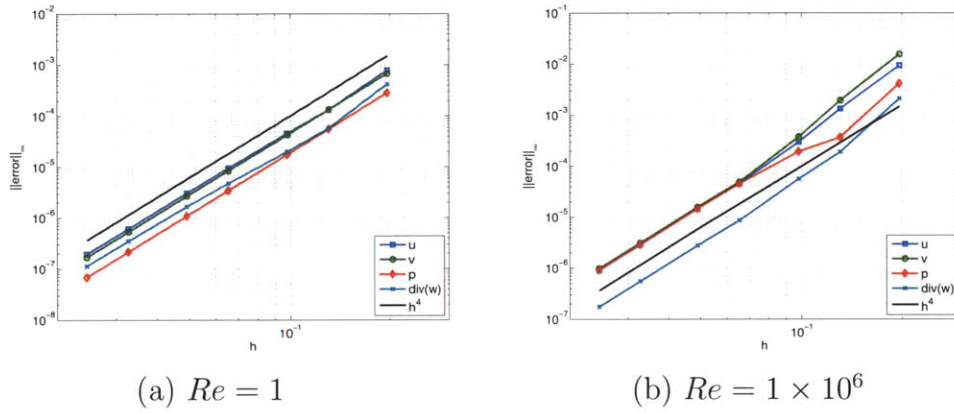


Figure 6-3: Convergence of the error in the L_∞ norm.

a circular boundary is immersed into the solution domain of example 1 and Dirichlet boundary conditions are imposed on it. The circular boundary is represented by the zero level of the the level set function

$$\phi(x, y) = r^2(x, y) - r_0^2,$$

$$r(x, y) = \sqrt{(x - x_0)^2 + (y - y_0)^2},$$

where $x_0 = y_0 = \pi$, and $r_0 = 1$. Figure 6-4 shows the solution domain with the boundary immersed in a Cartesian grid. The solution domain is the region exterior to this circle.

As discussed in remark 6.5, in this case the Reynolds number imposes a limitation

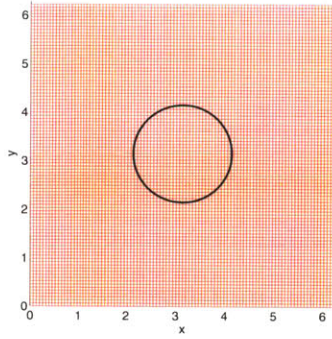


Figure 6-4: Solution domain with the boundary immersed in a 96×96 Cartesian grid.

to the minimum time step allowed. Hence, in this example the INSE associated with the exact solution (6.9) and $Re = 1$ is solved with time step $k = 4h^2$.

Figure 6-5 shows the solution at $t = 1$ in a 192×192 grid. The solution outside the domain is simply set to zero. Similar to the results presented in previous chapters, the boundary condition is properly enforced, without creating oscillations or spurious effects in the remainder of the solution. Moreover, figure 6-6(a) presents the variation of $\varphi = (\vec{\nabla} \cdot \vec{w})$ over time. Once again φ oscillates with bounded amplitude. However, experience shows that a stable solution is only possible after increasing the penalization coefficients used to impose the boundary conditions. Namely, for the velocity correction function, the coefficients were set to $c_P = 1 \times 10^4$, $c_N = 10$, while for the pressure correction function, $c_P = 100$, and $c_N = 10$. This difficulty may be a reflection of the fact the Johnston and Liu formulation imposes conservation of mass in a “marginally stable” fashion, as discussed in [26].

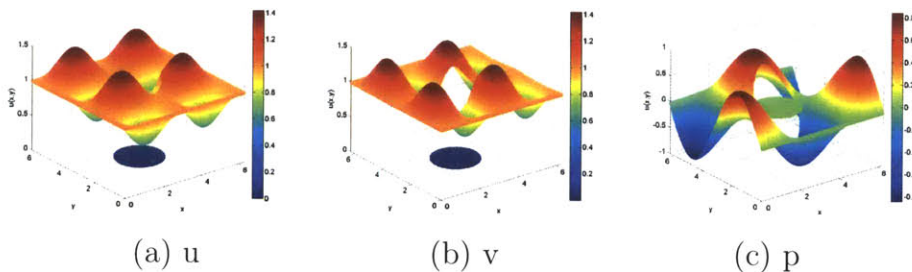


Figure 6-5: Solution at $t = 1$ for $Re = 1$.

Figure 6-6(b) shows the convergence of the error in the L_∞ norm. As expected, the errors converge to 4th order for the velocity components, and 3rd order for pressure and φ – see remark 6.3. Finally, figure 6-7 shows the convergence of the error in the solution and the gradient evaluated along the boundary. As we can observe, the components of velocity computed on the boundary converge to the same order as the solution in the interior of the domain, while the gradient is one order less accurate. On the other hand, the pressure and pressure gradient evaluated along the boundary both converge to 3rd order.

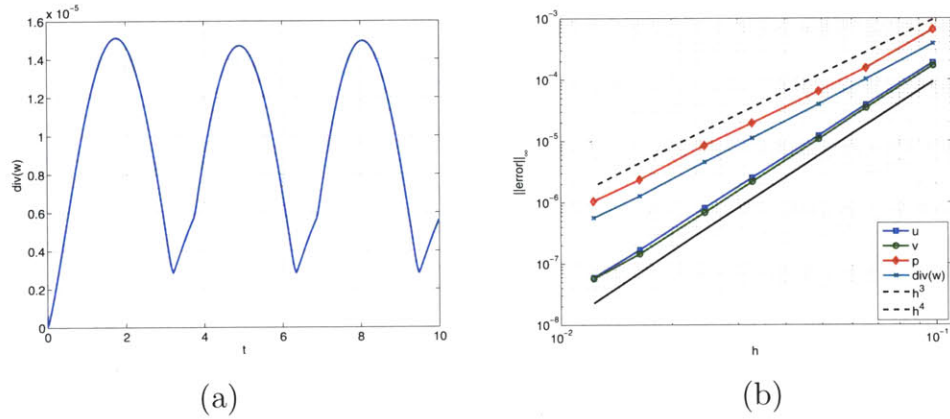


Figure 6-6: (a) Variation of the L_∞ norm of the divergence of the velocity over time. (b) Convergence of the error in the L_∞ norm.

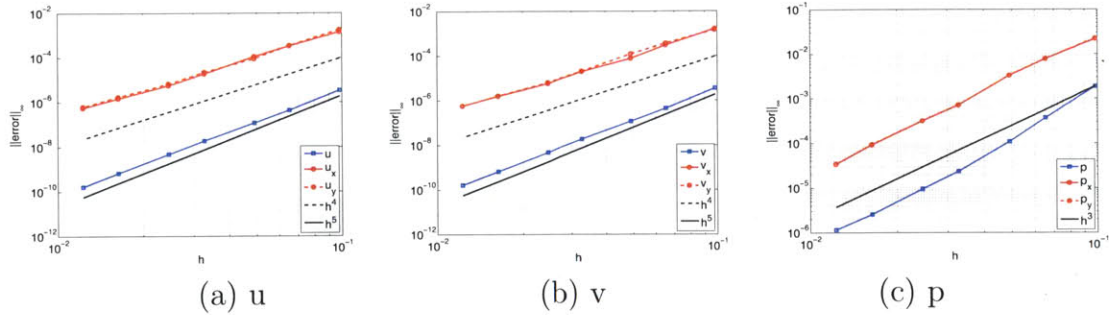


Figure 6-7: Convergence of the error along the boundary.

6.3.3 Flow over a cylinder

The INSE solver discussed in §6 was applied to solve three to fluid flow problems in the low Reynolds number regime. These applications involve the flow around a circular cylinder with $Re = 1$, $Re = 10$, and $Re = 20$. In simulations of flows around a cylinder, it is usual to use inflow/outflow boundary conditions that mimic an open space situation. To avoid the errors introduced by these boundary conditions – especially in a relatively small computational domain such as the one used here – the domain is assumed periodic and the flow is driven by a uniform body force. As a consequence, the definition of Reynolds number that used in these simulations is not the same used in the other examples, and the results presented here do not necessarily match the ones presented in the literature. Nevertheless, the solutions discussed below capture some of the key features of flows around a cylinder in the low Reynolds number regime, such as a well defined “separation bubble” and vortex shedding, which indicate that the numerical scheme is capable of producing quality solutions.

In reality, the periodic boundary conditions correspond to an infinite array of cylinders that extends in both directions. The computational box of size $(0, 6) \times (0, 3)$ represents one period of this infinite array. Figure 6-8 shows the computational box with the cylinder immersed in a regular Cartesian grid. The flow around these cylinders is driven by adding a uniform body force f in the positive x direction. For nondimensionalization purposes, the characteristic length is the diameter of the cylinder, and the characteristic speed is given by⁴

$$U = \sqrt{\frac{Lf}{6\rho}}.$$

As mentioned above, this definition of characteristic speed results in Reynolds numbers that are not compatible with the nondimensionalizations defined with the inflow speed. Hence, we must be careful when comparing the solutions presented here with

⁴The factor of 6 in this expression corresponds to the length of one period. Hence, in the absence of the cylinder, the uniform flow at $Re = 1$ has a unitary pressure drop.

other experiments listed in the literature.

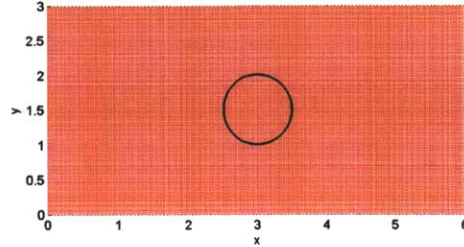


Figure 6-8: One period of the infinite array of cylinders. The boundary is immersed in a 268×128 Cartesian grid.

Moreover, since only the low Reynolds number regime is considered, $\lambda = 0$ in all three examples.

In the first example, the flow around a cylinder with $Re = 1$ is computed for $0 < t \leq 6$. The computational grid has 256×128 nodes and the time step is $k = 6/2731$. Figure 6-9 shows the solution fields at $t = 6$. Note that these contour plots use linear interpolation. Hence, close to the boundary the plot of the pressure distribution shows some oscillations because the pressure is arbitrarily set to zero inside the cylinder. However, these oscillations do not exist in the actual solution. In addition, figure 6-10 shows some streamlines of the flow around the cylinder at $t = 6$. From figures figures 6-9 and 6-10 we see that, at these low Reynolds numbers, the flow flow remains attached all around the cylinder and the solution is very smooth.

Figure 6-11(a) presents the variation of the divergence of the velocity over time. In this plot we can see that the initialization step introduces a spike in the divergence of velocity, which is quickly dissipated. Thus, the divergence-free condition is satisfied up to a small error. Finally, figure 6-11(b) presents the nondimensional forces acting on the cylinder over time. As expected, the lift force remains zero for all time, whereas the drag coefficient approaches the asymptotic value of $f_D = 6$. Moreover, after $t = 4$ the solution seems to reach a steady state, which is reflected in an almost constant

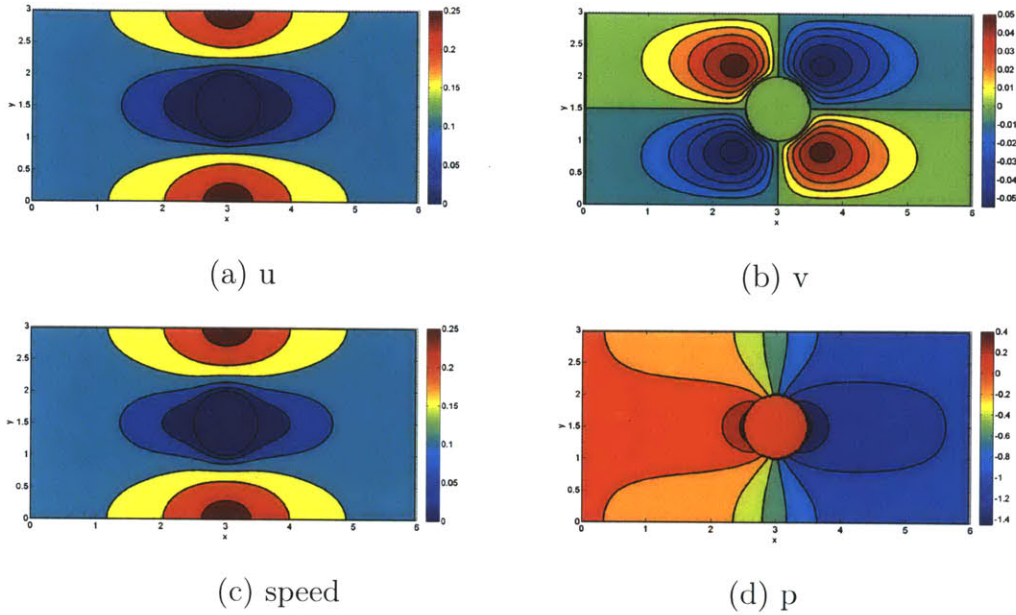


Figure 6-9: Solution at $t = 6$ for $Re = 1$. Speed denotes $s = \sqrt{u^2 + v^2}$.

value of the drag coefficient.

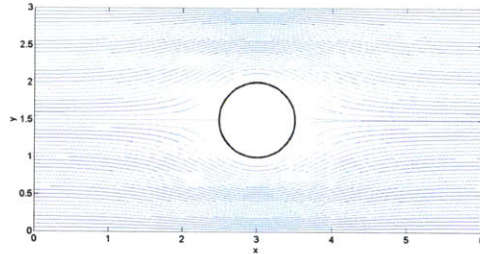


Figure 6-10: Streamlines of the flow around the cylinder at $t = 6$, with $Re = 1$.

In the second example, the flow around a cylinder with $Re = 10$ is computed for $0 < t \leq 30$. The computational grid has 512×256 nodes and the time step is $k = 30/5461$. Figure 6-12 shows contours of the solution at $t = 30$. In the contours of the velocity components we can note a small recirculation region just behind the cylinder, indicating the presence of a bubble caused by flow separation. This bubble is even more noticeable in the plots of the streamlines shown in figure 6-13. As we can observe in this figure, flow separation starts to occur after $t = 15$. By $t = 30$ the flow has practically reached a steady state.

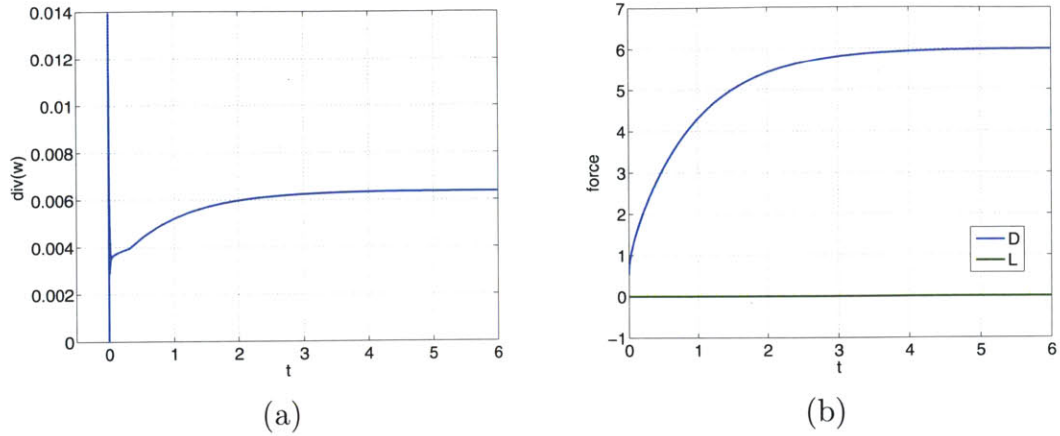


Figure 6-11: (a) Variation of the L_∞ norm of the divergence of velocity over time. (b) Nondimensional forces acting over the cylinder: drag and lift.

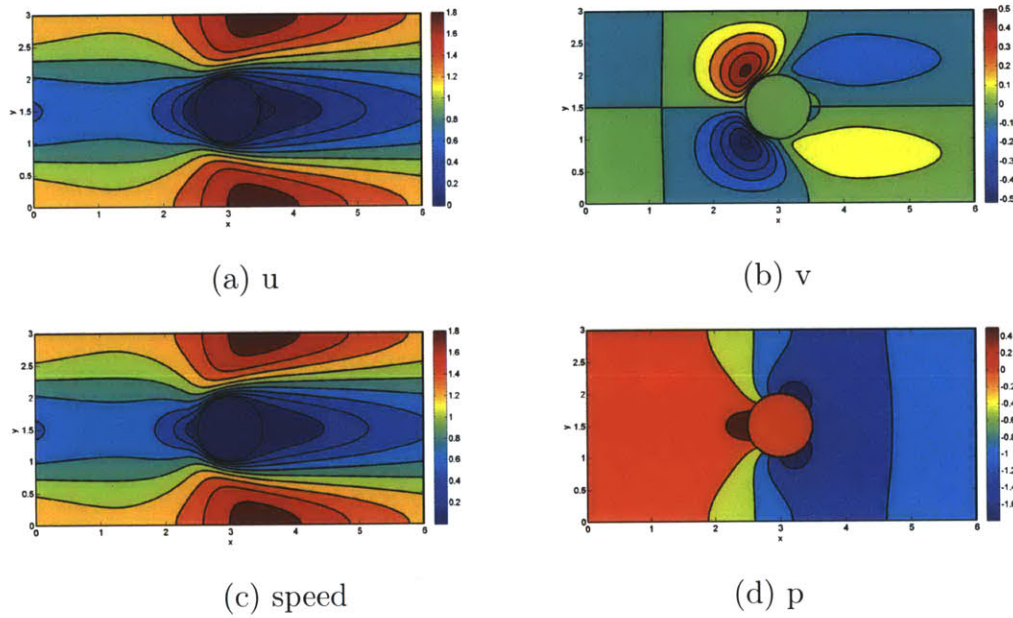


Figure 6-12: Solution at $t = 6$ for $Re = 30$. Speed denotes $s = \sqrt{u^2 + v^2}$.

In addition, figure 6-14(a) shows the variation of the divergence of velocity over time. Once again, after a spike in the first few time steps, the divergence decays and is maintained at a small amplitude. Finally, figure 6-14(b) shows the nondimensional forces acting on the cylinder over time. As expected, the lift force remains zero for all time, and the drag coefficient approaches the asymptotic steady state value of $f_D = 6$.

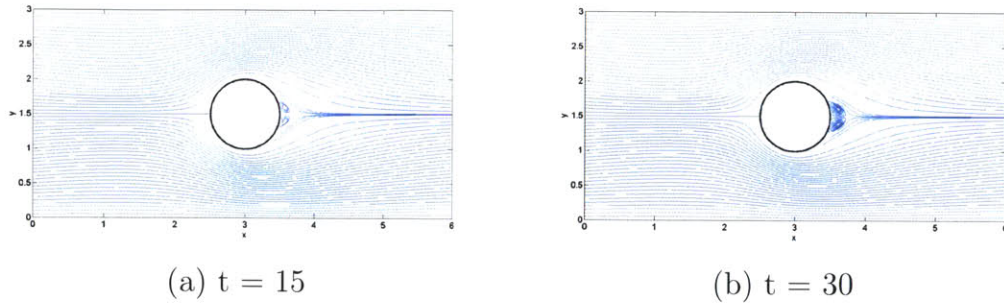


Figure 6-13: Streamlines of the flow around the cylinder with $Re = 10$.

In the last example, the flow around a cylinder with $Re = 20$ is computed for $0 < t \leq 60$. The computational grid has 512×256 nodes and the time step is $k = 60/10923$. Figure 6-15 shows contours of the solution at $t = 30$, while figure 6-16 shows contours of the solution at $t = 60$. As we can observe, the solution changes a lot in this time interval.

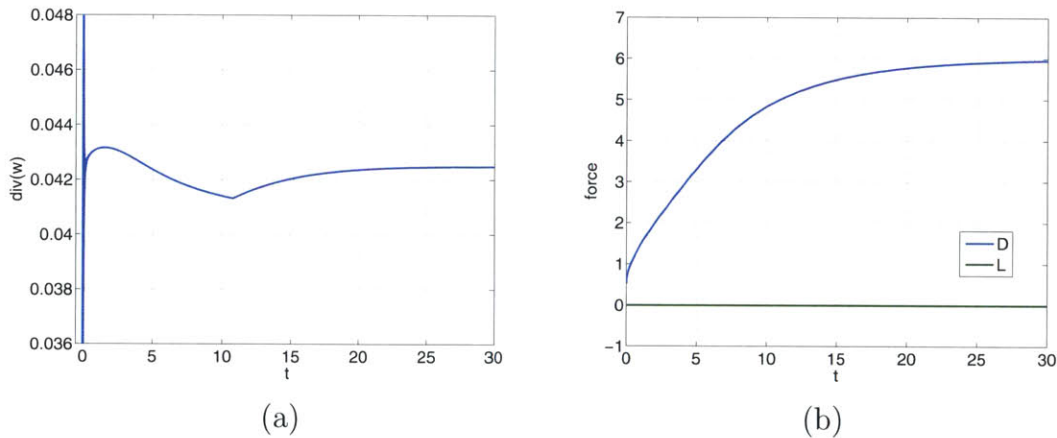


Figure 6-14: (a) Variation of the L_∞ norm of the divergence of velocity over time. (b) Nondimensional forces acting over the cylinder: drag and lift.

We can understand these changes better by looking at some streamlines presented in figure 6-17. Similar to the solution with $Re = 10$, a separation bubble forms behind the cylinder. Up to $t \approx 40$, this bubble grows slowly in length, but remains symmetric. After this time, the symmetric solution becomes unstable and the upper and lower circulation regions start to oscillate. At $t \approx 50$ we can observe the phenomenon of vortex shedding, in which the recirculation regions detach from the cylinder and are

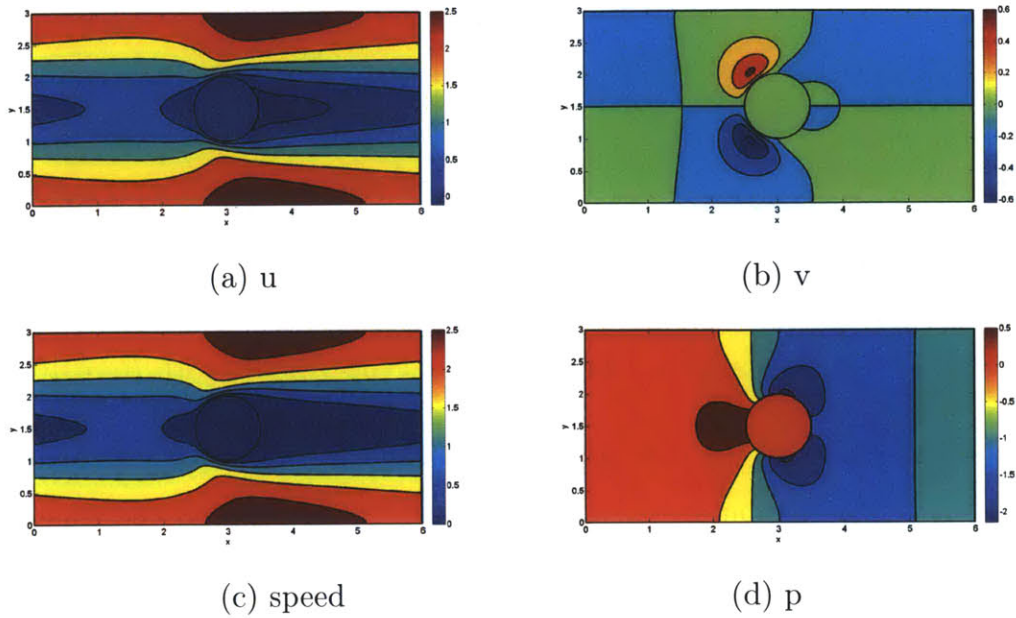


Figure 6-15: Solution at $t = 30$ for $Re = 20$. Speed denotes $s = \sqrt{u^2 + v^2}$.

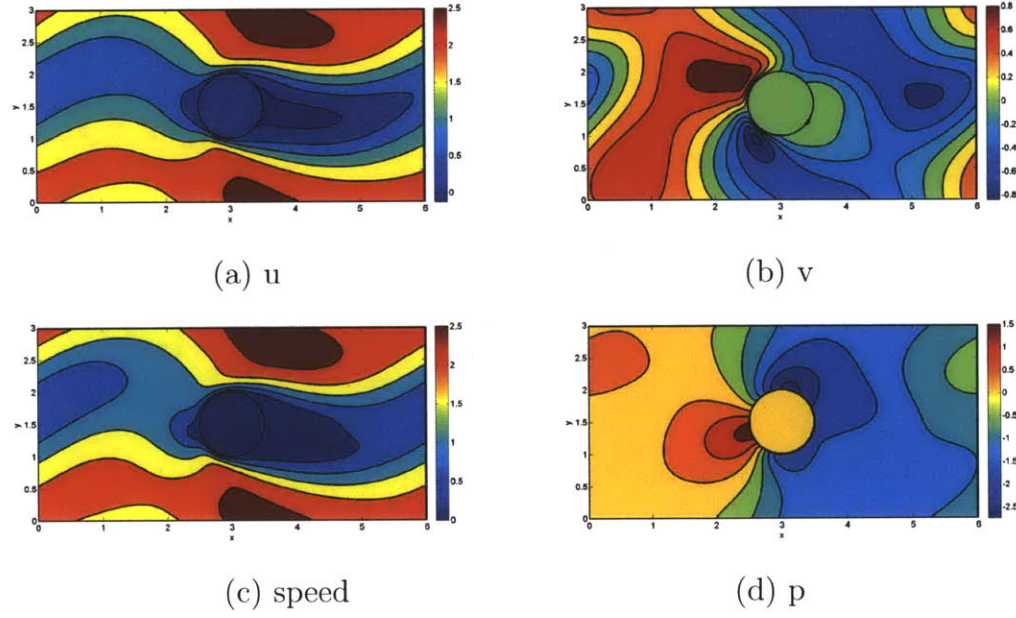


Figure 6-16: Solution at $t = 60$ for $Re = 20$. Speed denotes $s = \sqrt{u^2 + v^2}$.

carried with the flow. This process occurs in an asymmetric fashion: vortices are shed from the upper and lower parts of the cylinder in an alternate order. We can see this process in more detail in figure 6-18. In addition, because of the high viscosity of the

surrounding flow, the vortices shed by the cylinder are quickly dissipated. However, probably due to the small separation between the periodic cylinders, the asymmetric character of the flow is felt downstream by the adjacent cylinders. In fact, it is difficult to determine whether there is a steady solution to this situation, since the disturbances created around one cylinder do not have time to die out before reaching the next cylinder in the infinite array. The solution obtained up to $t = 60$ indicates that the perturbations grow in an unsteady fashion in this time interval.

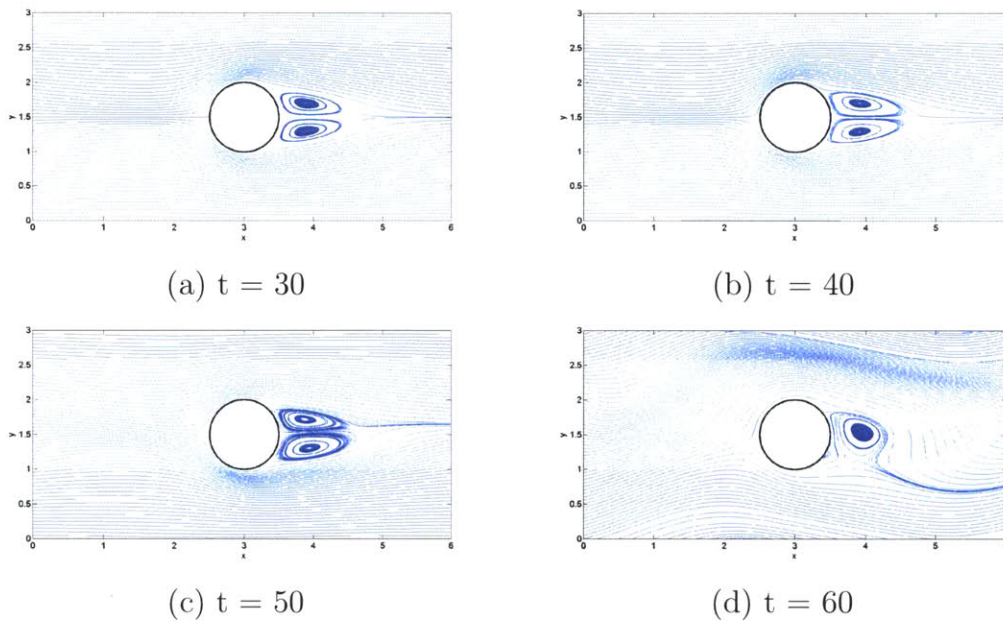


Figure 6-17: Streamlines of the flow around the cylinder with $Re = 20$.

In addition, figure 6-19(a) shows the variation of the divergence of velocity over time. In this example the divergence is also maintained at a small amplitude. However, we can note that the growing perturbations start to affect the divergence of velocity towards the end of the simulation. Finally, figure 6-19(b) shows the nondimensional forces acting on the cylinder over time. Up to $t \approx 40$, the flow is symmetric and the behavior is similar to previous examples: the lift force remains zero while the drag force approaches $f_D = 6$. However, as the asymmetry starts to develop, the lift force starts to oscillate with growing amplitude, and the drag experiences a slight increase.

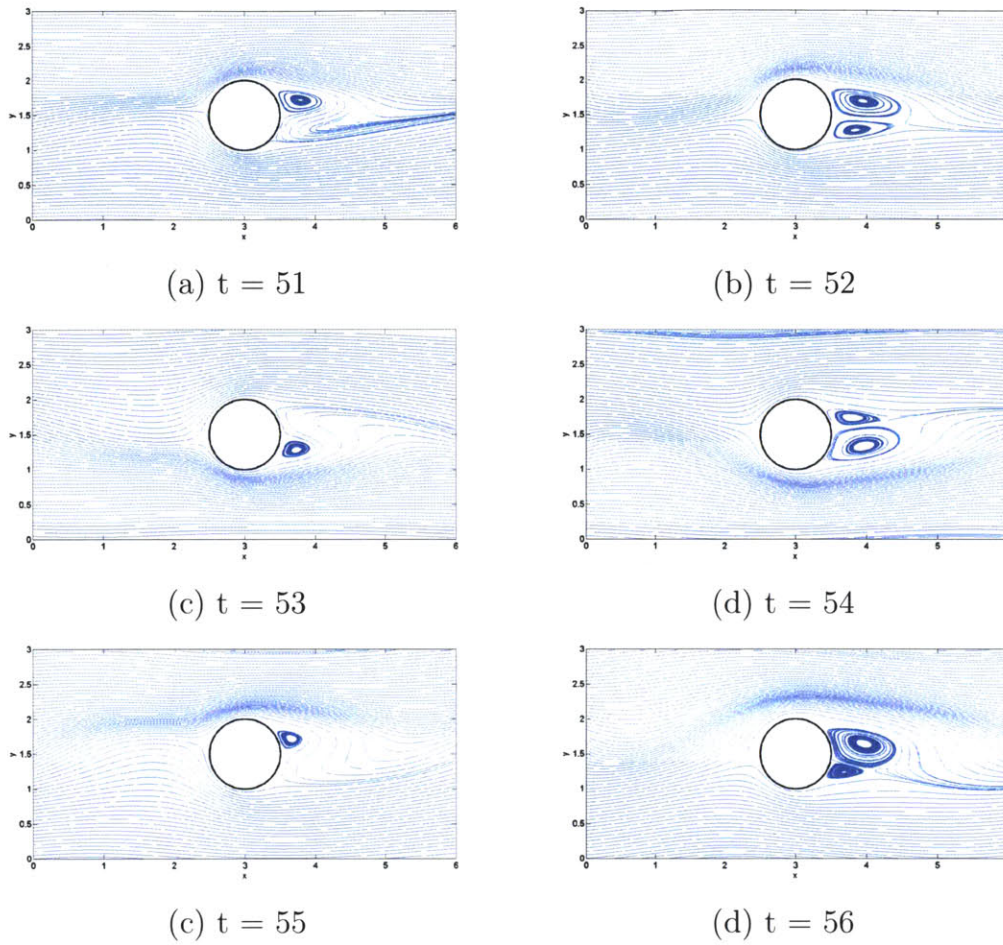


Figure 6-18: Streamlines showing vortex shedding behind cylinder for $t > 50$.

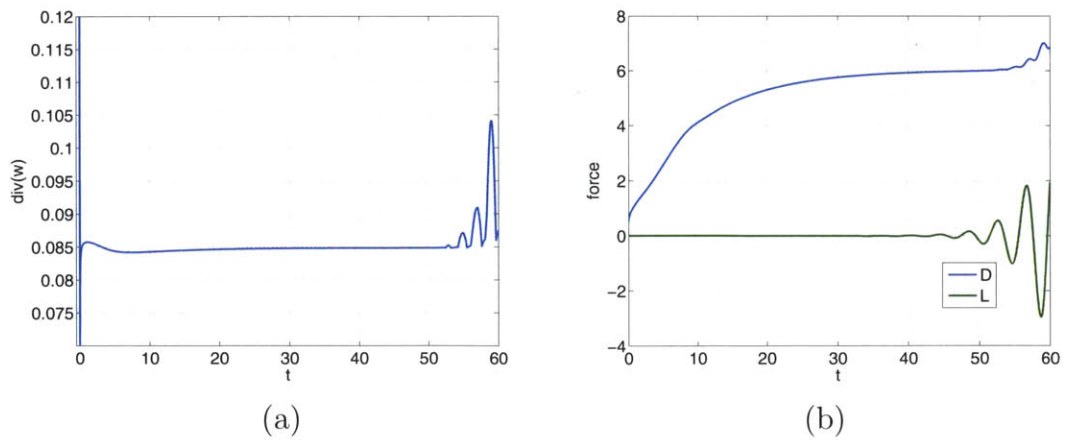


Figure 6-19: (a) Variation of the L_∞ norm of the divergence of velocity over time. (b) Nondimensional forces acting over the cylinder: drag and lift.

Chapter 7

Conclusion

7.1 Final Remarks

This thesis presents the *correction function method* (CFM), a new family family of immersed methods designed to solve the incompressible Navier-Stokes equations, and other related problems, to high order of accuracy. Hence, this method allows us to solve incompressible viscous flows to high order of accuracy, up to the boundary. As a consequence, the solution and its derivatives can be accurately evaluated on surfaces immersed in such flows. This is an important contribution to the area of immersed methods, which, in general, were restricted to 2nd order methods.

In summary, the CFM is based on the construction of *correction functions*, which are used to complete a standard discretization of the equations whenever the stencil straddles a boundary or interface immersed into the computational grid. The key features of the method are

- **Accuracy:** The correction function is characterized as the solution to partial differential equations defined locally in a neighborhood of the boundary and/or interface. Because this PDE can be solved to any desired order of accuracy, the correction function has, in principle, no accuracy limitations. In this thesis 3rd and 4th order implementations of the method are presented.
- **Robustness:** The PDE that defines the correction function is solved in a least

squares sense. Namely, the solution procedure is based on the minimization of a quadratic functional. This feature guarantees smooth solutions and a scheme that is robust in face of the myriad of configurations in which the immersed curves may cross the computational grid.

- **Efficiency:** The narrow band where the correction function is defined is divided into a series of small rectangular domains, usually one for each stencil¹ that straddles the interface. The PDE that defines the correction function is then solved locally within each of these rectangular domains. This feature allows us to split the solution of the correction function into a series of small problems. As consequence, computing the correction function is relatively inexpensive. Furthermore, because these problems are small, involving only a few grid nodes in the vicinity of the discretization stencil, we are able to use compact discretization stencils¹.

In addition, in this thesis a 2D version of the CFM is used to solve

- the constant coefficients Poisson equation with discontinuities across an internal interface.
- the piece-wise constant coefficients Poisson equation with discontinuities across an internal interface.
- the Poisson equation with immersed boundaries.
- the heat equation with immersed boundaries.
- the nonlinear convection-diffusion equation with immersed boundaries.
- the incompressible Navier-Stokes equations in the formulation by Johnston and Liu [23, 24].

Note that only low Reynolds number solutions of the incompressible Navier-Stokes equations are presented. Unfortunately, the code created for this thesis only supports

¹The stencil used to discretize the underlying equations. The PDE that defines the correction function is solved in function space: no additional stencils are used.

uniform computational grids. Hence, it was not possible to run numerical experiments with grids that are refined enough to capture the boundary layer in the high Reynolds number regime. This is one of the issues that remain for future research.

7.2 Future work

The list below include research issues that were encountered during the development of the work discussed here, but went beyond the scope of this thesis. Other extensions of the CFM for a wider range of applications are also mentioned.

- **High Reynolds number regime:** As discussed in §6, solving the INSE in the high Reynolds number regime requires very refined grids close to the boundaries. Hence, to solve flows in this regime, one needs a numerical scheme that supports grids with variable cell sizes (such as octree grids), including adaptations to use the CFM.
- **CFM for other formulations of the INSE:** In this thesis, only the INSE formulation proposed by Johnston and Liu [23,24] was implemented. However, it is not clear whether this formulation is stable in the high Reynolds number regime, even with the fix proposed in §6. On the other hand, the formulation proposed by Shirokoff and Rosales [26] enforces a better control over the gradient-free condition, which should lead to better stability properties. However, the boundary conditions involved in this formulation cannot be implemented with the CFM in the current form of the method. Hence, an adaptation of the CFM to this particular formulation is an interesting line of research for the future.
- **Moving boundaries:** Immersed methods, such as the CFM, are particularly interesting in unsteady simulations, where the boundaries and/or interfaces are moving and deforming over time. In such situations the domain of definition of the correction function must vary with time, and the consequences of these variations deserve careful consideration.

- **Extension to 3D:** Many complex engineering applications can only be properly addressed with fully three-dimensional models. For this reason, the extension of the CFM to 3D is an important line of future research. The conceptual basis of the CFM does not restrict it to 2D in any way. Hence, an extension to 3D should involve only geometrical considerations in the definition of the narrow band where the correction function exists.
- **Extension to higher-accuracy:** As mentioned before, in principle the CFM can be implemented to any desired order of accuracy. Hence, it would be interesting to explore the practical implications of higher order (higher than 4th) implementations.

Appendix A

The 9-point stencil for the Poisson equation

Unlike standard finite-differences discretizations, the 9-point stencil discretization of the Poisson equation is a discretization of the differential equation itself, and not of the differential operator. In essence, the 9-point stencil uses derivatives of the Poisson equation to eliminate the leading 2nd order errors of the standard 5-point stencil discretization of the Laplace operator. The result is a compact and 4th order accurate discretization of the Poisson equation.

The standard 5-point stencil discretization of the Laplace operator is given by

$$L^5 u_{i,j} = \hat{\partial}_{xx} u_{i,j} + \hat{\partial}_{yy} u_{i,j}, \quad (\text{A.1})$$

where $\hat{\partial}_{xx}$ and $\hat{\partial}_{yy}$ denote the centered difference discretization of the second derivatives:

$$\hat{\partial}_{xx} u_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2}, \quad (\text{A.2})$$

$$\hat{\partial}_{yy} u_{i,j} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_y^2}. \quad (\text{A.3})$$

The error resulting from this discretization is

$$\frac{h_x^2}{12}(u_{xxxx})_{i,j} + \frac{h_y^2}{12}(u_{yyyy})_{i,j} + \mathcal{O}(h^4). \quad (\text{A.4})$$

Now consider the Poisson equation $\Delta u = f$. The second derivatives of this equation result in

$$u_{xxxx} = f_{xx} - u_{xxyy}, \quad (\text{A.5a})$$

$$u_{yyyy} = f_{yy} - u_{xxyy}. \quad (\text{A.5b})$$

The mixed derivative $(\cdot)_{xxyy}$ can be computed to 2nd order using the compact 9-point stencil $\hat{\partial}_{xx}\hat{\partial}_{yy}(\cdot)$. Hence, we can eliminate the 2nd order term in (A.4) using this approximation to (A.5). Then, the 9-point discretization the Poisson equation is given by¹

$$L^5 u_{i,j} + \frac{1}{12}(h_x^2 + h_y^2)\hat{\partial}_{xx}\hat{\partial}_{yy}u_{i,j} = f_{i,j} + \frac{1}{12}(h_x^2 (f_{xx})_{i,j} + h_y^2 (f_{yy})_{i,j}). \quad (\text{A.6})$$

The higher derivatives of the source term – $(f_{xx})_{i,j}$ and $(f_{yy})_{i,j}$ – may be given analytically (if known), or computed using appropriate 2nd order discretizations.

¹Notice that here the possibility of different grid spacings in each direction is considered.

Appendix B

Bicubic interpolation

Bicubic interpolation is similar to bilinear interpolation, and can also be used to represent a function in a rectangular domain. However, whereas bilinear interpolation requires one piece of information per vertex of the rectangular domain, bicubic interpolation requires 4 pieces of information: function value, function gradient, and first mixed derivative (*i.e.* f_{xy}). For completeness, the relevant formulas for bicubic interpolation are presented below.

We use the classical multi-index notation, as in Ref. [42]. Thus, we represent the 4 vertices of the domain using the vector index $\vec{v} \in \{0, 1\}^2$. Namely, the 4 vertices are $\vec{x}_{\vec{v}} = (x_1^0 + v_1 \Delta x_1, x_2^0 + v_2 \Delta x_2)$, where (x_1^0, x_2^0) are the coordinates of the left-bottom vertex and Δx_i is the length of the domain in the x_i direction. Furthermore, given a scalar function ϕ , the 4 pieces of information needed per vertex are given by

$$\phi_{\vec{\alpha}}^{\vec{v}} = \partial^{\vec{\alpha}} \phi(\vec{x}_{\vec{v}}), \quad (\text{B.1})$$

where both $\vec{v}, \vec{\alpha} \in \{0, 1\}^2$ and

$$\partial^{\vec{\alpha}} = \partial_1^{\alpha_1} \partial_2^{\alpha_2}, \quad \partial_i^{\alpha_i} = (\Delta x_i)^{\alpha_i} \frac{\partial^{\alpha_i}}{\partial x_i^{\alpha_i}}. \quad (\text{B.2})$$

Then the 16 polynomials that constitute the standard basis for the bicubic interpo-

lation can be written in the compact form

$$W_{\vec{\alpha}}^{\vec{v}} = \prod_{i=1}^2 w_{\alpha_i}^{v_i}(\bar{x}_i), \quad (\text{B.3})$$

where $\bar{x}_i = \frac{x_i - x_i^0}{\Delta x_i}$, and w_{α}^v is the cubic polynomial

$$w_{\alpha}^v(x) = \begin{cases} f(x) & \text{for } v = 0 \text{ and } \alpha = 0, \\ f(1-x) & \text{for } v = 1 \text{ and } \alpha = 0, \\ g(x) & \text{for } v = 0 \text{ and } \alpha = 1, \\ -g(1-x) & \text{for } v = 1 \text{ and } \alpha = 1, \end{cases} \quad (\text{B.4})$$

where $f(x) = 1 - 3x^2 + 2x^3$ and $g(x) = x(1-x)^2$.

Finally, the bicubic interpolation of a scalar function ϕ is given by the following linear combination of the basis functions:

$$\mathcal{H}(\vec{x}) = \sum_{\vec{v}, \vec{\alpha} \in \{0,1\}^2} W_{\vec{\alpha}}^{\vec{v}} \phi_{\vec{\alpha}}^{\vec{v}} \quad (\text{B.5})$$

As defined above (standard bicubic interpolation), 16 parameters are needed to determine the bicubic. However, in Ref. [42] a method (“cell-based approach”) is introduced, that reduces the number of degrees of freedom to 12, without compromising accuracy. This method uses information from the first derivatives to obtain approximate formulae for the mixed derivatives. In the present work, we adopt this cell-based approach.

Appendix C

Issues affecting the construction of

$$\Omega_{\Gamma}^{i,j}$$

As discussed in §2.4, the CFM is based on local solutions to the PDE (2.9) in subregions of Ω_{Γ} – which we call $\Omega_{\Gamma}^{i,j}$. However, there is a certain degree of arbitrariness in how $\Omega_{\Gamma}^{i,j}$ is defined. This appendix discusses several factors that influence the definition of $\Omega_{\Gamma}^{i,j}$. In addition, four distinct approaches are presented, with increasing level of robustness (and, unfortunately, complexity).

The discussion on §2.3 results in only two constraints on $\Omega_{\Gamma}^{i,j}$:

- $\Omega_{\Gamma}^{i,j}$ should be small, since the local problems' condition numbers increase exponentially with distance from Γ – see remarks 2.4 and 2.5.
- $\Omega_{\Gamma}^{i,j}$ should contain all the nodes where the correction function D is needed.

However, when $\Omega_{\Gamma}^{i,j}$ is defined, practical algorithmic constraints must also be considered, as explained below.

First, D is computed by solving a PDE in a weak fashion, and this procedure involves integrations over $\Omega_{\Gamma}^{i,j}$. Thus, it is useful to restrict $\Omega_{\Gamma}^{i,j}$ to an elementary geometrical shape, so that simple quadrature rules can be applied to evaluate the integrals. Second, if $\Omega_{\Gamma}^{i,j}$ is a rectangle, Hermite splines can be used to represent D in

$\Omega_{\Gamma}^{i,j}$ to high order accuracy. For these reasons, here $\Omega_{\Gamma}^{i,j}$ is restricted to be a rectangle¹.

Third, the interface representation must also be considered. In principle, the solution to the PDE (2.9) depends on information given along the interface only, and it is completely independent on the underlying grid. Nevertheless, when the interface is represented implicitly, the location of the interface depends on values defined at grid points. Hence, to construct $\Omega_{\Gamma}^{i,j}$, we need to define a set of grid nodes that are used to reconstruct a local piece of interface. In this thesis the interface is reconstructed using information from the 3×3 set of nodes that defines the 9-point stencil. The approaches discussed in §C.1 through §C.3 are based on this premise.

Although the strategy mentioned above can be easily implemented, it also ties $\Omega_{\Gamma}^{i,j}$ to the underlying grid, whereas it should depend only on the interface geometry. Hence it results in definitions for $\Omega_{\Gamma}^{i,j}$ that cannot track the interface optimally. The approach presented in §C.4, allows $\Omega_{\Gamma}^{i,j}$ to adapt to the local interface geometry, regardless of the underlying grid. The idea is to first identify a piece of the interface based on a pre-determined set of grid cells, and then use this information to construct an optimal $\Omega_{\Gamma}^{i,j}$. This approach leads to a somewhat intricate, but very robust definition for $\Omega_{\Gamma}^{i,j}$.

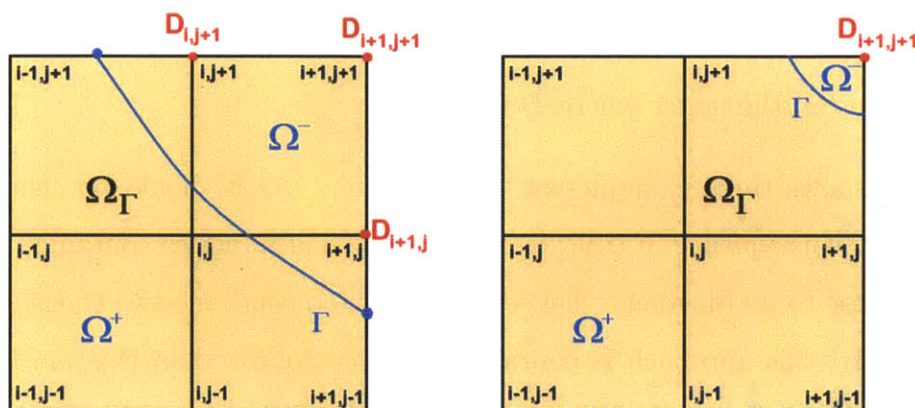
Finally, note that explicit representations of the interface are not constrained by the underlying grid. Moreover, information on the interface geometry is readily available anywhere along the interface. Hence, in this case, we can use the approach discussed in §C.4 in a straightforward fashion. By contrast, the less robust approaches in §C.1 through §C.3 become more involved in this context, because they require the additional work of constraining the explicit representation to the underlying grid.

Obviously, the algorithms presented here (§C.1 through §C.4) represent only a few of the possible ways in which $\Omega_{\Gamma}^{i,j}$ can be defined. Nevertheless, these approaches serve as practical examples of how different factors must be balanced to design robust schemes.

¹Clearly, other simple geometrical shapes, with other types of approximations for D , should be possible.

C.1 Naive Grid-Aligned Stencil-Centered Approach.

In this approach, $\Omega_\Gamma^{i,j}$ is fitted to the underlying grid by defining it as the $2h_x \times 2h_y$ box that covers the 9-point stencil. Figure 3-1 shows two examples.



(a) Well-posed.

(b) Ill-posed.

Figure 3-1: $\Omega_\Gamma^{i,j}$ as defined by the naive grid-aligned stencil-centered approach.

This approach is very appealing because of its simplicity, but it has serious flaws and *it is not recommended*. The reason is that the piece of the interface contained within $\Omega_\Gamma^{i,j}$ can become arbitrarily small – see figure 3-1(b). Then the arguments that make the local Cauchy problem well posed no longer apply – see remarks 2.4 through 2.6. In essence, the biggest frequency encoded in the interface, $k_{\max} \approx 1/\text{length}(\Gamma/\Omega_\Gamma^{i,j})$, can become arbitrarily large – while the characteristic length of $\Omega_\Gamma^{i,j}$ remains $\mathcal{O}(h)$. As a consequence, the condition number for the local Cauchy problem can become arbitrarily large. This approach is described here merely as an example of the problems that can arise from a very simplistic definition of $\Omega_\Gamma^{i,j}$.

C.2 Compact Grid-Aligned Stencil-Centered Approach.

This is the approach described in detail in §2.4.3. In summary, $\Omega_{\Gamma}^{i,j}$ is defined as the smallest rectangle that

- (i) is aligned with the grid.
- (ii) includes the piece of the interface contained within the stencil.
- (iii) includes all the nodes where D is needed.

Figure 3-2 shows three examples of this definition. As it should be clear from this figure, a key consequence of (i-iii) is that the piece of interface contained within $\Omega_{\Gamma}^{i,j}$ is always close to its diagonal – hence it is never too small relative to the size of $\Omega_{\Gamma}^{i,j}$. Consequently, this approach is considerably more robust than the one in §C.1. In fact, this approach is successfully applied to all examples shown in §2.5.

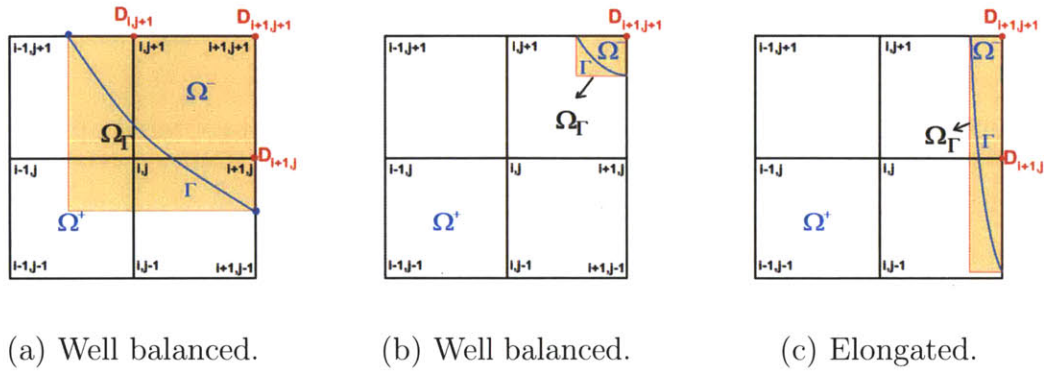


Figure 3-2: $\Omega_{\Gamma}^{i,j}$ as defined by the compact grid-aligned stencil-centered approach.

Unfortunately, the requirements (i-ii) in this approach tie $\Omega_{\Gamma}^{i,j}$ to the grid and the stencil. As mentioned earlier, these constraints may lead to an $\Omega_{\Gamma}^{i,j}$ which is not the best fit to the geometry of the interface. Figure 3-2(c) depicts a situation where this strategy may lead to trouble. This situation happens when there is an almost perfect alignment of the interface with the grid, which can result in an excessively elongated $\Omega_{\Gamma}^{i,j}$ – in the worse case scenario, this set could reduce to a line. Although

the local Cauchy problem remains well conditioned, the elongated sets can interfere with the process used to solve (2.9). However, experience shows that this approach works well with the bicubic representation for a 4th order scheme. This approach only led to trouble in a 2nd order version of the scheme (based on bilinear interpolation) presented in [87].

C.3 Free Stencil-Centered Approach

Here a compromise solution is presented to avoid an elongated $\Omega_{\Gamma}^{i,j}$. In this approach, the constraint (i) in §C.2 is abandoned, but not constraint (ii) – since (ii) is convenient when the interface is represented implicitly. In this approach $\Omega_{\Gamma}^{i,j}$ is defined as the smallest rectangle that

- (i*) Is aligned with the grid rotated by an angle θ_r , where $\theta_r = \theta_{\Gamma} - \pi/4$ and θ_{Γ} characterizes the interface alignment with respect to the grid (*e.g.* the polar angle of the vector tangent to the piece of interface inside the stencil at its mid-point).
- (ii*) Includes the piece of the interface contained within the stencil.
- (iii*) Includes all the nodes where D is needed.

Figure 3-3 shows two examples of this approach.

The implementation of the present approach is very similar to that of the one in §C.2. The only additional work is to compute θ_r and to write the interface and points where D is needed in the rotated frame of reference. In both these approaches the diagonal of $\Omega_{\Gamma}^{i,j}$ is very close to the piece of interface contained within the stencil, which guarantees a well conditioned local problem. However, here the addition of a rotation keeps $\Omega_{\Gamma}^{i,j}$ nearly square, and avoids elongated geometries. The price paid for this regularity is that the sets $\Omega_{\Gamma}^{i,j}$ created using this approach can be a little larger than the ones from §C.2 – with both sets including the exact same piece of interface. In such situations, the present approach results in a somewhat larger condition number.

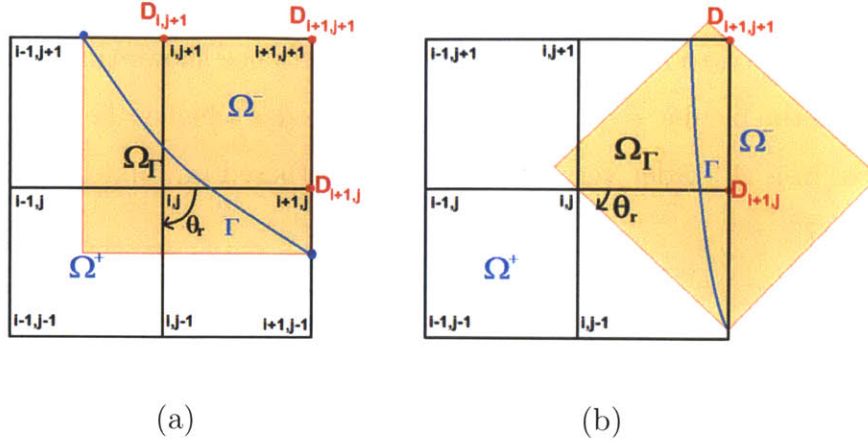


Figure 3-3: $\Omega_\Gamma^{i,j}$ as defined by the free stencil-centered approach.

C.4 Node-Centered Approach

This approach defines $\Omega_\Gamma^{i,j}$ in a fashion that is completely independent from the underlying grid and the discretization stencil. In fact, instead of associating each $\Omega_\Gamma^{i,j}$ to a particular stencil, this approach is based on a different $\Omega_\Gamma^{i,j}$ for each node where the correction is needed – hence the name node-centered, rather than stencil-centered. As a consequence, whereas the prior strategies lead to multiple values of D at the same node (one value per stencil, see remark 2.10), here there is a unique value of D at each node.

In this approach, $\Omega_\Gamma^{i,j}$ is defined by the following steps.

1. Identify the interface in the 4 grid cells that surround a given node. This step is not needed if the interface is represented explicitly.
2. Find the point along the interface that is closest to node $(i,j) = P_0$. This point becomes the center of $\Omega_\Gamma^{i,j}$. There is no need to obtain P_0 very accurately. Small errors in P_0 result only in small shifts in $\Omega_\Gamma^{i,j}$, which do not affect the quality of the solution.
3. Compute \hat{t}_0 , the vector tangent to the interface at P_0 . This vector defines one of the diagonals of $\Omega_\Gamma^{i,j}$. The normal vector \hat{n}_0 defines the other diagonal. Again, high accuracy is not needed.

4. Then $\Omega_\Gamma^{i,j}$ is the square with side length $2\sqrt{h_x^2 + h_y^2}$, centered at P_0 , and diagonals parallel to \hat{t}_0 and $\hat{n}_0 - \Omega_\Gamma^{i,j}$ need not be aligned with the grid.

Figure 3-4 shows two examples of this approach.

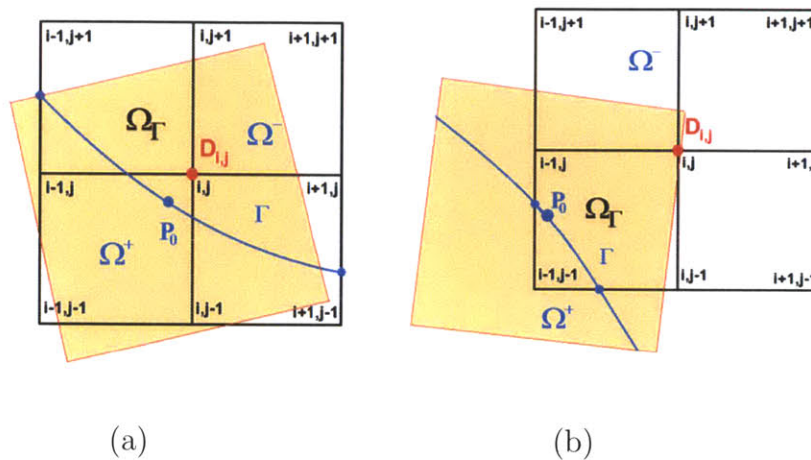


Figure 3-4: $\Omega_\Gamma^{i,j}$ as defined by the node-centered approach.

Note that the piece of interface contained within $\Omega_\Gamma^{i,j}$, as defined by steps 1-4 above, is not necessarily the same found in step 1. Hence, after defining $\Omega_\Gamma^{i,j}$, one still needs to identify the piece of interface that lies within it. For an explicit representation of the interface, this additional step is not particularly costly, but the same is not true for an implicit representation.

This approach is very robust because it always creates a square $\Omega_\Gamma^{i,j}$, with the interface within it close to one of the diagonals – as guaranteed by steps 2 and 3. Hence, the local Cauchy problem is always well conditioned. Furthermore, making $\Omega_\Gamma^{i,j}$ square (to avoid elongation) does not result in larger condition numbers as in §C.3 because the larger $\Omega_\Gamma^{i,j}$ contains an equally larger piece of the interface.

Finally, the small oscillations observed in the convergence plots shown in §2.5 occur because these calculations are carried out with the approach discussed in §C.2 – which produces sets $\Omega_\Gamma^{i,j}$ that are not uniform in size, nor shape, along the interface. However, tests show that these oscillations do not occur with the node-centered approach, for which all the $\Omega_\Gamma^{i,j}$ are squares of the same size. Unfortunately, as pointed

out earlier, the node-centered approach is not well suited for calculations using an interface represented implicitly.

Appendix D

Ill-posed problem

As discussed in §3.2, the discontinuous coefficient Poisson equation can result in an ill-conditioned problem when $\alpha = \beta^-/\beta^+ \gg 1$. To understand this issue, let us look at the jump condition for the normal fluxes – equation (3.7d):

$$\begin{aligned} \beta^+ u_n^+(\vec{x}) - \beta^- u_n^-(\vec{x}) &= b(\vec{x}) \\ \Rightarrow u_n^+(\vec{x}) - \alpha u_n^-(\vec{x}) &= \frac{b(\vec{x})}{\beta^+} \end{aligned} \quad \text{for } \vec{x} \in \Gamma. \quad (\text{D.1})$$

Hence, assuming that u_n^+ and b/β^+ are $\mathcal{O}(1)$ quantities, as α grows u_n^- must tend to zero. In the limit $\alpha \rightarrow \infty$, $u_n^- \rightarrow 0$, which means that (D.1) becomes a Neumann boundary condition for u^- . In other words, the solution for u^- decouples from u^+ . In turn, u^+ becomes the solution to the problem with Dirichlet boundary condition $u^+(\vec{x}) = a(\vec{x}) - u^-(\vec{x})$, $\vec{x} \in \Gamma$.

However, the solution to a Neumann problem is defined only up to an arbitrary constant. In situations where the solution is defined only up to a constant over the entire domain, such as when Neumann or periodic boundary conditions are imposed over $\partial\Omega$, this is not an issue. Some additional constraint is needed to define a unique solution throughout the domain. This is a necessary condition whether α is large or not. On the other hand, if Dirichlet boundary condition is imposed on $\partial\Omega$, the solution is not arbitrary. Nonetheless, as α grows, the discontinuous coefficient Poisson equation becomes ill-conditioned.

Remark D.1. *Note that the numerical solution used to solve (3.7) is not relevant. This conditioning issue is inherent of the problem being considered and is not a result of poor numerical discretization.* ♣

One possible fix for this conditioning issue is to impose an additional constraint on the solution. However, since the solution is not arbitrary, this additional constraint must be redundant, and only play a role as $\alpha \rightarrow \infty$. It must only reiterate information that is weakly enforced due to the unbalance between the different terms in (D.1). For instance, in the context of the boundary integral formulation introduced in §4, we can integrate (4.8c) to obtain

$$\int_{\Gamma} [w_n] dS = \frac{1}{\langle \beta \rangle} \int_{\Gamma} b dS - \lambda \int_{\Gamma} v_n dS - \lambda \int_{\Gamma} \langle w_n \rangle dS, \quad (\text{D.2})$$

where $\lambda = [\beta]/\langle \beta \rangle$. Now note that

$$\int_{\Gamma} v_n dS = \int_{\Gamma} u_n^- dS = \frac{1}{\beta^-} \int_{\Omega^-} f^- dV, \quad (\text{D.3})$$

and $\rho = [w_n]$. Thus

$$\int_{\Gamma} \rho dS = \frac{2}{\lambda + 2} \left(\int_{\Gamma} b dS + \frac{1}{\beta^-} \int_{\Omega^-} f^- dV \right). \quad (\text{D.4})$$

This expression reinforces the argument mentioned above. As α grows, λ approaches the value $\lambda = -2$. Hence, the mean value of ρ becomes very sensitive to any errors in the data inputted to (4.8), especially those coming from the computation of v_n .

One way to force the numerical scheme to “see” the proper mean is to enforce (D.4) as the extra condition and input the correct value to the code. In principle one can compute the correct value using (D.4) since it only involves known parameters of the problem: b and f^-/β^- . However, because of the $(\lambda + 2)$ term in the denominator, the integrals in (D.4) must be computed very accurately, which may not always be easy to do. In the example shown in §4.2, this expression is evaluated using the trapezoidal rule, which resulted in very accurate estimates. One may hope that in real applications, physical reasoning might help to deduce the correct value for the mean

of ρ . (A physical problem that leads to this particular situation was not encountered in this research). Finally, there may be other ways to enforce a redundant condition that is more suitable to the particular problem being solved. Using (D.4) is just one alternative that works when the integrals can be computed accurately.

Bibliography

- [1] S. O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, *Journal of Computational Physics* 100 (1) (1992) 25–37. doi:10.1016/0021-9991(92)90307-K.
- [2] J. Li, Y. Renardy, Numerical study of flows of two immiscible liquids at low reynolds number, *SIAM Review* 42 (3) (2000) 417–439. doi:10.1137/S0036144599354604.
- [3] E. M. Purcell, Life at low reynolds number, *American Journal of Physics* 45 (1) (1977) 3–11. doi:10.1119/1.10903.
- [4] Z. J. Wang, Dissecting insect flight, *Annual Review of Fluid Mechanics* 37 (1) (2005) 183–210. doi:10.1146/annurev.fluid.36.050802.121940.
- [5] K.-B. Lee, J.-H. Kim, C. Kim, Aerodynamic effects of structural flexibility in two-dimensional insect flapping flight, *Journal of Aircraft* 48 (3) (2011) 894–909. doi:10.2514/1.C031115.
- [6] C. S. Peskin, Numerical analysis of blood flow in the heart, *Journal of Computational Physics* 25 (3) (1977) 220–252. doi:10.1016/0021-9991(77)90100-0.
- [7] U.s. air force fact sheet: Mq-1b predator (Jun. 2011).
URL <http://www.af.mil/information/factsheets/factsheet.asp?id=122>
- [8] G. Chesshire, W. D. Henshaw, Composite overlapping meshes for the solution of partial differential equations, *Journal of Computational Physics* 90 (1) (1990) 1–64. doi:10.1016/0021-9991(90)90196-8.
- [9] S. E. Sherer, J. N. Scott, High-order compact finite-difference methods on general overset grids, *Journal of Computational Physics* 210 (2) (2005) 459–496. doi:10.1016/j.jcp.2005.04.017.
- [10] M. A. Yerry, M. S. Shephard, Automatic three-dimensional mesh generation by the modified-octree technique, *International Journal for Numerical Methods in Engineering* 20 (11) (1984) 1965–1990. doi:10.1002/nme.1620201103.
- [11] P.-O. Persson, G. Strang, A simple mesh generator in matlab, *SIAM Review* 46 (2) (2004) 329–345. doi:10.1137/S0036144503429121.

- [12] W. J. Schroeder, M. S. Shephard, A combined octree/delaunay method for fully automatic 3-d mesh generation, *International Journal for Numerical Methods in Engineering* 29 (1) (1990) 37–55. doi:10.1002/nme.1620290105.
- [13] S. Crippa, Improvement of unstructured computational fluid dynamics simulations through novel mesh generation methodologies, *Journal of Aircraft* 48 (3) (2011) 1036–1044. doi:10.2514/1.C031219.
- [14] T. J. Baker, Three decades of meshing; a retrospective view, in: *Proceeding 16th AIAA Computational Fluid Dynamics Conference*, Orlando, Florida, 2003.
- [15] A. N. Marques, J. L. F. Azevedo, Numerical calculation of impulsive and indicial aerodynamic responses using computational aerodynamics techniques, *Journal of Aircraft* 45 (4) (2008) 1112–1135. doi:10.2514/1.32151.
- [16] P.-O. Persson, J. Bonet, J. Peraire, Discontinuous Galerkin solution of the Navier-Stokes equations on deformable domains, *Computer Methods in Applied Mechanics and Engineering* 198 (17-20) (2009) 1585–1595. doi:10.1016/j.cma.2009.01.012.
- [17] S. Keye, Fluid–structure coupled analysis of a transport aircraft and flight–test validation, *Journal of Aircraft* 48 (2) (2011) 381–390. doi:10.2514/1.C000235.
- [18] A. J. Chorin, A numerical method for solving incompressible viscous flow problems, *Journal of Computational Physics* 2 (1) (1967) 12–26. doi:10.1016/0021-9991(67)90037-X.
- [19] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier-Stokes equations, *Journal of Computational Physics* 59 (2) (1985) 308–323. doi:10.1016/0021-9991(85)90148-2.
- [20] J. B. Bell, P. Colella, H. M. Glaz, A second-order projection method for the incompressible Navier-Stokes equations, *Journal of Computational Physics* 85 (2) (1989) 257–283. doi:10.1016/0021-9991(89)90151-4.
- [21] W. D. Henshaw, A fourth-order accurate method for the incompressible Navier-Stokes equations on overlapping grids, *Journal of Computational Physics* 113 (1) (1994) 13–25. doi:10.1006/jcph.1994.1114.
- [22] W. D. Henshaw, H.-O. Kreiss, L. G. Reyna, A fourth-order-accurate difference approximation for the incompressible Navier-Stokes equations, *Computers and Fluids* 23 (4) (1994) 575–593. doi:10.1016/0045-7930(94)90053-1.
- [23] H. Johnston, J.-G. Liu, Finite difference schemes for incompressible flow based on local pressure boundary conditions, *Journal of Computational Physics* 180 (1) (2002) 120–154. doi:10.1006/jcph.2002.7079.

- [24] H. Johnston, J.-G. Liu, Accurate, stable and efficient Navier-Stokes solvers based on explicit treatment of the pressure term, *Journal of Computational Physics* 199 (1) (2004) 221–259. doi:10.1016/j.jcp.2004.02.009.
- [25] R. D. Guy, A. L. Fogelson, Stability of approximate projection methods on cell-centered grids, *Journal of Computational Physics* 203 (2) (2005) 517–538. doi:10.1016/j.jcp.2004.09.005.
- [26] D. Shirokoff, R. Rosales, An efficient method for the incompressible Navier-Stokes equations on irregular domains with no-slip boundary conditions, high order up to the boundary, *Journal of Computational Physics* 230 (23) (2011) 8619–8646. doi:10.1016/j.jcp.2011.08.011.
- [27] F. H. Harlow, J. E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, *Physics of Fluids* 8 (12) (1965) 2182–2189. doi:10.1063/1.1761178.
- [28] F. H. Harlow, J. E. Welch, Numerical study of large-amplitude free-surface motions, *Physics of Fluids* 9 (5) (1966) 842–851. doi:10.1063/1.1761784.
- [29] R. K.-C. Chan, R. L. Street, A computer study of finite-amplitude water waves, *Journal of Computational Physics* 6 (1) (1970) 68–94. doi:10.1016/0021-9991(70)90005-7.
- [30] C. S. Peskin, Flow patterns around heart valves: A numerical method, *Journal of Computational Physics* 10 (2) (1972) 252–271. doi:10.1016/0021-9991(72)90065-4.
- [31] A. Mayo, The fast solution of Poisson’s and the biharmonic equations on irregular regions, *SIAM Journal on Numerical Analysis* 21 (2) (1984) 285–299. doi:10.1137/0721021.
- [32] H. S. Udaykumar, R. Mittal, W. Shyy, Computation of solid-liquid phase fronts in the sharp interface limit on fixed grids, *Journal of Computational Physics* 153 (2) (1999) 535–574. doi:10.1006/jcph.1999.6294.
- [33] A. J. Chorin, Flame advection and propagation algorithms, *Journal of Computational Physics* 35 (1) (1980) 1–11. doi:10.1016/0021-9991(80)90030-3.
- [34] C. W. Hirt, B. D. Nichols, Volume of fluid (vof) method for the dynamics of free boundaries, *Journal of Computational Physics* 39 (1) (1981) 201–225. doi:10.1016/0021-9991(81)90145-5.
- [35] S. Osher, J. A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations, *Journal of Computational Physics* 79 (1) (1988) 12–49. doi:10.1016/0021-9991(88)90002-2.

- [36] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, *Journal of Computational Physics* 183 (1) (2002) 83–116. doi:10.1006/jcph.2002.7166.
- [37] S. Chen, B. Merriman, S. Osher, P. Smereka, A simple level set method for solving Stefan problems, *Journal of Computational Physics* 135 (1) (1997) 8–29. doi:10.1006/jcph.1997.5721.
- [38] C. Min, F. Gibou, A second order accurate level set method on non-graded adaptive Cartesian grids, *Journal of Computational Physics* 225 (1) (2007) 300–321. doi:10.1016/j.jcp.2006.11.034.
- [39] C. Min, F. Gibou, Geometric integration over irregular domains with application to level-set methods, *Journal of Computational Physics* 226 (2) (2007) 1432–1443. doi:10.1016/j.jcp.2007.05.032.
- [40] J. A. Sethian, *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, Cambridge monographs on applied and computational mathematics, Cambridge University Press, 1999.
- [41] S. Osher, R. P. Fedkiw, *Level set methods and dynamic implicit surfaces*, Applied mathematical sciences, Springer, 2003.
- [42] J.-C. Nave, R. R. Rosales, B. Seibold, A gradient-augmented level set method with an optimally local, coherent advection scheme, *Journal of Computational Physics* 229 (10) (2010) 3802–3827. doi:10.1016/j.jcp.2010.01.029.
- [43] L. N. Trefethen, D. Bau, *Numerical Linear Algebra*, SIAM: Society for Industrial and Applied Mathematics, 1997.
- [44] C. Tu, C. S. Peskin, Stability and instability in the computation of flows with moving immersed boundaries: A comparison of three methods, *SIAM Journal on Scientific and Statistical Computing* 13 (6) (1992) 1361–1376. doi:10.1137/0913077.
- [45] C. S. Peskin, B. F. Printz, Improved volume conservation in the computation of flows with immersed elastic boundaries, *Journal of Computational Physics* 105 (1) (1993) 33–46. doi:10.1006/jcph.1993.1051.
- [46] M.-C. Lai, C. S. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, *Journal of Computational Physics* 160 (2) (2000) 705–719. doi:10.1006/jcph.2000.6483.
- [47] C. S. Peskin, The immersed boundary method, *Acta Numerica* 11 (2002) 479–517. doi:10.1017/S0962492902000077.

- [48] D. Goldstein, R. Handler, L. Sirovich, Modeling a no-slip flow boundary with an external force field, *Journal of Computational Physics* 105 (2) (1993) 354–366. doi:10.1006/jcph.1993.1081.
- [49] E. A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, *Journal of Computational Physics* 161 (1) (2000) 35–60. doi:10.1006/jcph.2000.6484.
- [50] R. J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, *SIAM Journal on Numerical Analysis* 31 (4) (1994) 1019–1044. doi:10.1137/0731054.
- [51] R. J. LeVeque, Z. Li, Immersed interface methods for Stokes flow with elastic boundaries or surface tension, *SIAM Journal on Scientific Computing* 18 (3) (1997) 709–735. doi:10.1137/S1064827595282532.
- [52] Z. Li, M.-C. Lai, The immersed interface method for the Navier-Stokes equations with singular forces, *Journal of Computational Physics* 171 (2) (2001) 822–842. doi:10.1006/jcph.2001.6813.
- [53] L. Lee, R. J. LeVeque, An immersed interface method for incompressible Navier-Stokes equations, *SIAM Journal on Scientific Computing* 25 (3) (2003) 832–856. doi:10.1137/S1064827502414060.
- [54] Z. Li, K. Ito, The immersed interface method: numerical solutions of PDEs involving interfaces and irregular domains, *Frontiers in applied mathematics*, SIAM, Society for Industrial and Applied Mathematics, 2006.
- [55] D. V. Le, B. C. Khoo, J. Peraire, An immersed interface method for viscous incompressible flows involving rigid and flexible boundaries, *Journal of Computational Physics* 220 (1) (2006) 109–138. doi:10.1016/j.jcp.2006.05.004.
- [56] X. Zhong, A new high-order immersed interface method for solving elliptic equations with imbedded interface of discontinuity, *Journal of Computational Physics* 225 (1) (2007) 1066–1099. doi:10.1016/j.jcp.2007.01.017.
- [57] R. P. Fedkiw, T. Aslam, S. Xu, The ghost fluid method for deflagration and detonation discontinuities, *Journal of Computational Physics* 154 (2) (1999) 393–427. doi:10.1006/jcph.1999.6320.
- [58] R. P. Fedkiw, T. Aslam, B. Merriman, S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method), *Journal of Computational Physics* 152 (2) (1999) 457–492. doi:10.1006/jcph.1999.6236.
- [59] X.-D. Liu, R. P. Fedkiw, M. Kang, A boundary condition capturing method for Poisson’s equation on irregular domains, *Journal of Computational Physics* 160 (1) (2000) 151–178. doi:10.1006/jcph.2000.6444.

- [60] M. Kang, R. P. Fedkiw, X.-D. Liu, A boundary condition capturing method for multiphase incompressible flow, *Journal of Scientific Computing* 15 (2000) 323–360. doi:10.1023/A:1011178417620.
- [61] D. Q. Nguyen, R. P. Fedkiw, M. Kang, A boundary condition capturing method for incompressible flame discontinuities, *Journal of Computational Physics* 172 (1) (2001) 71–98. doi:10.1006/jcph.2001.6812.
- [62] F. Gibou, L. Chen, D. Nguyen, S. Banerjee, A level set based sharp interface method for the multiphase incompressible Navier–Stokes equations with phase change, *Journal of Computational Physics* 222 (2) (2007) 536–555. doi:10.1016/j.jcp.2006.07.035.
- [63] A. Mayo, The rapid evaluation of volume integrals of potential theory on general regions, *Journal of Computational Physics* 100 (2) (1992) 236–245. doi:10.1016/0021-9991(92)90231-M.
- [64] A. McKenney, L. Greengard, A. Mayo, A fast Poisson solver for complex geometries, *Journal of Computational Physics* 118 (2) (1995) 348–355. doi:10.1006/jcph.1995.1104.
- [65] H. Johansen, P. Colella, A Cartesian grid embedded boundary method for Poisson’s equation on irregular domains, *Journal of Computational Physics* 147 (1) (1998) 60–85. doi:10.1006/jcph.1998.5965.
- [66] Z. Jomaa, C. Macaskill, The embedded finite difference method for the Poisson equation in a domain with an irregular boundary and Dirichlet boundary conditions, *Journal of Computational Physics* 202 (2) (2005) 488–506. doi:10.1016/j.jcp.2004.07.011.
- [67] M. N. Linnick, H. F. Fasel, A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains, *Journal of Computational Physics* 204 (1) (2005) 157–192. doi:10.1016/j.jcp.2004.09.017.
- [68] J. A. Sethian, J. Strain, Crystal growth and dendritic solidification, *Journal of Computational Physics* 98 (2) (1992) 231–253. doi:10.1016/0021-9991(92)90140-T.
- [69] F. Gibou, R. P. Fedkiw, L.-T. Cheng, M. Kang, A second-order-accurate symmetric discretization of the Poisson equation on irregular domains, *Journal of Computational Physics* 176 (1) (2002) 205–227. doi:10.1006/jcph.2001.6977.
- [70] F. Gibou, R. Fedkiw, A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem, *Journal of Computational Physics* 202 (2) (2005) 577–601. doi:10.1016/j.jcp.2004.07.018.

- [71] H. Chen, C. Min, F. Gibou, A supra-convergent finite difference scheme for the Poisson and heat equations on irregular domains and non-graded adaptive Cartesian grids, *Journal of Scientific Computing* 31 (1) (2007) 19–60. doi:10.1007/s10915-006-9122-8.
- [72] I. Babuška, The finite element method for elliptic equations with discontinuous coefficients, *Computing* 5 (1970) 207–213. doi:10.1007/BF02248021.
- [73] J. W. Barret, C. M. Elliott, Fitted and unfitted finite–element methods for elliptic equations with smooth interfaces, *IMA Journal of Numerical Analysis* 7 (3) (1987) 283–300. doi:10.1093/imanum/7.3.283.
- [74] A. Hansbo, P. Hansbo, An unfitted finite element method, based on Nitsche’s method, for elliptic interface problems, *Computer Methods in Applied Mechanics and Engineering* 191 (47–48) (2002) 5537–5552. doi:10.1016/S0045-7825(02)00524-8.
- [75] N. Moës, J. Dolbow, T. Belytschko, A finite element method for crack growth without remeshing, *International Journal for Numerical Methods in Engineering* 46 (1) (1999) 131–150. doi:10.1002/(SICI)1097-0207(19990910)46:1<131::AID-NME726>3.0.CO;2-J.
- [76] J. Dolbow, I. Harari, An efficient finite element method for embedded interface problems, *International Journal for Numerical Methods in Engineering* 78 (2) (2009) 229–252. doi:10.1002/nme.2486.
- [77] J. Bedrossian, J. H. von Brecht, S. Zhu, E. Sifakis, J. M. Teran, A second order virtual node method for elliptic problems with interfaces and irregular domains, *Journal of Computational Physics* 229 (18) (2010) 6405–6426. doi:10.1016/j.jcp.2010.05.002.
- [78] Z. Li, T. Lin, X. Wu, New cartesian grid methods for interface problems using the finite element formulation, *Numerische Mathematik* 96 (2003) 61–98. doi:10.1007/s00211-003-0473-x.
- [79] S. Hou, X.-D. Liu, A numerical method for solving variable coefficient elliptic equation with interfaces, *Journal of Computational Physics* 202 (2) (2005) 411–445. doi:10.1016/j.jcp.2004.07.016.
- [80] Y. Gong, B. Li, Z. Li, Immersed–interface finite–element methods for elliptic interface problems with nonhomogeneous jump conditions, *SIAM Journal on Numerical Analysis* 46 (1) (2008) 472–495. doi:10.1137/060666482.
- [81] J.-S. Huh, J. A. Sethian, Exact subgrid interface correction schemes for elliptic interface problems, *Proceedings of the National Academy of Sciences* 105 (29) (2008) 9874–9879. doi:10.1073/pnas.0707997105.
- [82] D. Clarke, H. Hassan, Euler calculations for multielement airfoils using cartesian grids, *AIAA Journal* 24 (3) (1986) 353–358. doi:10.2514/3.9273.

- [83] M. J. Berger, R. J. LeVeque, An adaptive cartesian mesh algorithm for the Euler equations in arbitrary geometries, in: Proceeding 9th AIAA Computational Fluid Dynamics Conference, Buffalo, NY, 1989.
- [84] H. Ji, F.-S. Lien, E. Yee, An efficient second-order accurate cut-cell method for solving the variable coefficient poisson equation with jump conditions on irregular domains, *International Journal for Numerical Methods in Fluids* 52 (7) (2006) 723–748. doi:10.1002/flid.1199.
- [85] D. P. Young, R. G. Melvin, M. B. Bieterman, F. T. Johnson, S. S. Samant, J. E. Bussoletti, A locally refined rectangular grid finite element method: Application to computational fluid dynamics and computational physics, *Journal of Computational Physics* 92 (1) (1991) 1–66. doi:10.1016/0021-9991(91)90291-R.
- [86] K. J. Fidkowski, D. L. Darmofal, A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations, *Journal of Computational Physics* 225 (2) (2007) 1653 – 1672. doi:10.1016/j.jcp.2007.02.007.
- [87] A. N. Marques, J.-C. Nave, R. R. Rosales, A correction function method for Poisson problems with interface jump conditions, *Journal of Computational Physics* 230 (20) (2011) 7567–7597. doi:10.1016/j.jcp.2011.06.014.
- [88] A. Mikhlin, A. Armstrong, *Integral Equations and their Applications to Certain Problems Mechanics, Mathematical Physics and Technology*, International Series of Monographs in the Science of the Solid State, Elsevier Science & Technology, 1957.
- [89] K. Atkinson, *The numerical solution of integral equations of the second kind*, Cambridge monographs on applied and computational mathematics, Cambridge University Press, 1997.
- [90] V. Rokhlin, Rapid solution of integral equations of classical potential theory, *Journal of Computational Physics* 60 (2) (1985) 187–207. doi:10.1016/0021-9991(85)90002-6.
- [91] F. X. Canning, Sparse approximation for solving integral equations with oscillatory kernels, *SIAM Journal on Scientific and Statistical Computing* 13 (1) (1992) 71–87. doi:10.1137/0913004.
- [92] K. Nabors, F. T. Korsmeyer, F. T. Leighton, J. White, Preconditioned, adaptive, multipole-accelerated iterative methods for three-dimensional first-kind integral equations of potential theory, *SIAM Journal on Scientific Computing* 15 (3) (1994) 713–735. doi:10.1137/0915046.
- [93] G. Evans, J. Blackledge, P. Yardley, *Analytic methods for partial differential equations*, Springer undergraduate mathematics series, Springer, 2000.

- [94] J. Stoer, R. Bulirsch, Introduction to numerical analysis, Texts in applied mathematics, Springer, 2002.
- [95] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, *Journal of Computational Physics* 77 (2) (1988) 439–471. doi:10.1016/0021-9991(88)90177-5.
- [96] X.-D. Liu, S. Osher, T. Chan, Weighted essentially non-oscillatory schemes, *Journal of Computational Physics* 115 (1) (1994) 200–212. doi:10.1006/jcph.1994.1187.
- [97] G.-S. Jiang, C.-W. Shu, Efficient implementation of weighted eno schemes, *Journal of Computational Physics* 126 (1) (1996) 202–228. doi:10.1006/jcph.1996.0130.
- [98] A. J. Chorin, Numerical solution of the navier-stokes equations, *Mathematics of Computation* 22 (104) (1968) 745–762.
- [99] J. Zhu, J. Sethian, Projection methods coupled to level set interface techniques, *Journal of Computational Physics* 102 (1) (1992) 128–138. doi:10.1016/S0021-9991(05)80011-7.
- [100] D. L. Brown, R. Cortez, M. L. Minion, Accurate projection methods for the incompressible Navier-Stokes equations, *Journal of Computational Physics* 168 (2) (2001) 464–499. doi:10.1006/jcph.2001.6715.