Title
On Embedded Methods for Crack Propagation, Virtual Surgery, Shattered Objects in Computer Animation, and Elliptic Partial Differential Equations

Author
Hellrung, Jeffrey Lee

Publication Date
2012

Peer reviewed|Thesis/dissertation

University of California
Los Angeles

On Embedded Methods for Crack Propagation, Virtual Surgery, Shattered Objects in Computer Animation, and Elliptic Partial Differential Equations

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mathematics

by

Jeffrey Lee Hellrung, Jr.

2012

Abstract of the Dissertation

On Embedded Methods for Crack Propagation, Virtual Surgery, Shattered Objects in Computer Animation, and Elliptic Partial Differential Equations

by

Jeffrey Lee Hellrung, Jr.
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2012
Professor Joseph M. Teran, Chair

We present a collection of embedded methods to solve a variety of scientific computing problems in both 2 and 3 dimensions. Embedded methods make use of a structured background mesh which does not conform to the irregular geometry, such as the domain boundary, of the problem. Instead, the irregular geometry is embedded within the structured mesh's elements, providing a framework to solve many problems involving crack propagation, progressive fracturing, dynamic interfaces, and shape optimization.

In Part I, we apply the mesh cutting algorithm of Sifakis et al. [SDF07] to investigate the modeling of cracks, surgical incisions, and shattering. Specifically, we present a geometrically flexible and straightforward crack propagation method which combines the *eXtended Finite Element Method* (XFEM) with [SDF07] and an innovative integration scheme which makes use of a subordinate quadrature mesh. We also discuss the application of [SDF07] and other advances in numerical methods to address the challenges of a virtual surgery simulator. We conclude Part I by describing a system to facilitate the modeling of cracked and shattered objects in the context of visual effects and computer animation.

In Part II, we present a numerical method utilizing virtual degrees of freedom to efficiently solve elliptic partial differential equations (specifically: Poisson's equation with interfacial jump conditions; and linear elasticity in the nearly incompressible regime) on irregular domains within a regular background Cartesian grid. Our method enforces Dirichlet boundary conditions and interfacial jump conditions weakly, formulating our system as a constrained

minimization problem. In this context, we describe an algorithm to generate an associated discrete Lagrange multiplier space that allows one to derive an equivalent symmetric positive definite linear system. We provide a family of multigrid algorithms to solve this linear system with near optimal efficiency. Our method is second order accurate in L^∞ and possesses a feature set rarely found among the broad class of embedded methods for elliptic problems.

The dissertation of Jeffrey Lee Hellrung, Jr. is approved.

Demetri Terzopoulos

Stanley J. Osher

Christopher R. Anderson

Joseph M. Teran, Committee Chair

University of California, Los Angeles

2012

Table of Contents

List of Figures

List of Tables

Acknowledgments

I wish to thank the following for their support over the last several years as I've progressed through my doctoral degree here at UCLA:

- My family, especially my mom. I know it was difficult for her to resist the temptation to ask me about my degree progress every week, especially in the later years.
- My girlfriend, Sirian; and her mom, Lisa.
- The rest of my entering class who've remained in the department as long as I have. Or longer.

The research in Chapters 1, 4, and 5 was partially supported by UC Lab Fees Research / Department of Energy grant 09-LR-04-116741-BERA; Office of Naval Research grants N00014-03-1-0071 and N00014-10-1-0730; and National Science Foundation grant CCF-0830554. Additionally, the research in Chapter 1 was partially supported by National Science Foundation grant DMS-0914813; and the research in Chapters 4 and 5 by National Science Foundation grants DMS-0502315 and DMS-0652427.

Chapter 1 is a version (with moderate revisions) of "An XFEM method for modeling geometrically elaborate crack propagation in brittle materials" by Casey L. Richardson, Jan Hegemann, Eftychios Sifakis, Jeffrey Hellrung, and Joseph M. Teran (PI) in *International Journal for Numerical Methods in Engineering* (88(10):1042–1065, 9 December 2011) (DOI:10.1002/nme.3211).

Chapter 2 is a version (with minor revisions) of "Local Flaps: A Real-Time Finite Element Based Solution to the Plastic Surgery Defect Puzzle" by Eftychios Sifakis, Jeffrey Hellrung, Joseph Teran (PI), Aaron Oliker, and Court Cutting, M.D. in *Studies in Health Technology and Informatics* (142:313–318, 2009) (PMID:19377176).

Chapter 3 is a version (with minor revisions) of "Geometric fracture modeling in BOLT" by Jeffrey Hellrung, Andrew Selle, Arthur Shek, Eftychios Sifakis, and Joseph Teran (PI) in *SIGGRAPH 2009: Talks* (SIGGRAPH '09, pp. 7:1–7:1, New York, NY, USA, 2009) (DOI:10.1145/1597990.1597997). This research was supported by Walt Disney Animation Studios.

Chapter 4 is a version (with moderate revisions) of "A Second Order Virtual Node Method for Elliptic Problems with Interfaces and Irregular Domains in Three Dimensions" by Jeffrey Lee Hellrung, Jr., Luming Wang, Eftychios Sifakis, and Joseph M. Teran (PI) in *Journal of Computational Physics* (231(4):2015–2048, February 2012) (DOI:10.1016/j.jcp.2011.11.023). I wish to thank Jacob Bedrossian and James H. von Brecht for their helpful discussions; and Russell Howes and Alexey Stomakhin for submitting typographical errors and providing comments on the later drafts of the publication submission.

Chapter 5 is a version (with moderate revisions) of "A second-order virtual node algorithm for nearly incompressible linear elasticity in irregular domains" by Yongning Zhu,

Yuting Wang, Jeffrey Hellrung, Alejandro Cantarero, Eftychios Sifakis, and Joseph M. Teran (PI) (accepted for publication in *Journal of Computational Physics*, 2012).

Vita

2003, summer	Research Student for Associate Professor Jon T. Jacobsen of Harvey Mudd College (Claremont, CA)
2004, summer	Summer Intern at Auditude^{TM}, Inc. (Los Angeles, CA)
2004 - 2005	Clinic Project Manager serving Hewlett-Packard Company (Palo Alto, CA) via Harvey Mudd College Mathematics Clinic
2005, May	Received degree of Bachelor of Science in Mathematics (with High Distinction, Honors in Mathematics) from Harvey Mudd College
2005 - 2012	Teaching Fellow for the Department of Mathematics, University of California Los Angeles
2005, summer	Prof. NE MTS Level 1 at The Aerospace Corporation (El Segundo, CA)
2006, June	Received degree of Master of Arts in Mathematics from University of California Los Angeles
2006, summer	Prof. NE MTS Level 1 at The Aerospace Corporation
2007, summer	Prof. NE MTS Level 1 at The Aerospace Corporation
2008, summer	Graduate Associate at Walt Disney Animation Studios (Burbank, CA)
2010, summer	Teaching Assistant at Park City Mathematics Institute (Park City, UT)

Publications

M. Hecht, D. Buettner, **J. Hellrung**. "Risk assessment of real time digital control systems." *Proceedings of the RAMS '06. Annual Reliability and Maintainability Symposium, 2006*, pp. 409–415, 2006. DOI:10.1109/RAMS.2006.1677409

Eftychios Sifakis, **Jeffrey Hellrung**, Joseph Teran, Aaron Oliker, Court Cutting, M.D. "Local Flaps: A Real-Time Finite Element Based Solution to the Plastic Surgery Defect Puzzle." *Studies in Health Technology and Informatics*, 142:313–318, 2009. PMID:19377176

Jeffrey Hellrung, Andrew Selle, Arthur Shek, Eftychios Sifakis, Joseph Teran. "Geometric fracture modeling in BOLT." *SIGGRAPH 2009: Talks*, SIGGRAPH '09, pp. 7:1–7:1, New York, NY, USA, 2009. DOI:10.1145/1597990.1597997

Casey L. Richardson, Jan Hegemann, Eftychios Sifakis, **Jeffrey Hellrung**, Joseph M. Teran. "An XFEM method for modeling geometrically elaborate crack propagation in brittle materials." *International Journal for Numerical Methods in Engineering*, 88(10):1042–1065, December 2011. DOI:10.1002/nme.3211

Jeffrey Lee Hellrung, Jr., Luming Wang, Eftychios Sifakis, Joseph M. Teran. "A Second Order Virtual Node Method for Elliptic Problems with Interfaces and Irregular Domains in Three Dimensions." *Journal of Computational Physics*, 231(4):2015–2048, February 2012. DOI:10.1016/j.jcp.2011.11.023

Yongning Zhu, Yuting Wang, **Jeffrey Hellrung**, Alejandro Cantarero, Eftychios Sifakis, Joseph M. Teran. "A second-order virtual node algorithm for nearly incompressible linear elasticity in irregular domains." (accepted for publication in *Journal of Computational Physics*, 2012)

Awards

William Lowell Putnam Mathematical Competition - Top-500 Individual Placement (2001, 2004); Top-200 Individual Placement (2002, 2003); 11^{th} Team Placement (2004)

Courtney S. Coleman Prize in Mathematics (awarded by HMC; 2003)

ACM (Association for Computing Machinery) Programming Contest - 20^{th} (of 59) place (2003); 7^{th} (of 63) place (2004)

MCM (Mathematical Contest in Modeling) - Meritorious Winner (2004)

Microsoft Imagine Cup Algorithm Invitational - 18^{th} place internationally (2004)

Stavros Busenberg Prize in Applied Mathematics (awarded by HMC; 2004)

Robert Borrelli Clinic Prize for Most Outstanding Clinic Team (awarded by HMC; 2005)

Chancellor's Prize (awarded by UCLA; 2005 - 2006)

VIGRE Fellowship (awarded by UCLA Department of Mathematics; 2005 - 2009)

ICFP (International Conference in Functional Programming) Programming Contest - 80^{th} (of 215+) place (2010); 95^{th} (of 199) place (2011)

Google Games Santa Monica - 3^{rd} place (2011)

Introduction

The simulation of a variety of physical phenomena often requires addressing frequent, and sometimes quite drastic, topological changes. Fracture simulations and virtual surgery simulations require the dynamic introduction of one or more co-dimension one *crack surfaces* and user-defined surgical incisions, respectively, which may be completely new or may extend some crack or incision introduced at a previous time step. Multiphase fluid flow and phase change problems naturally have dynamic interfaces which divide the original domain into multiple irregularly shaped subdomains. Further, many shape optimization procedures continuously change the geometry – and, perhaps, even the topology – of the domain on which one must repeatedly solve some partial differential equation as a subproblem.

The common thread among all the above examples is the necessity to deal with complicated and irregular geometries, whether it be crack surfaces, interfaces, or domain boundaries. A natural approach to this complexity is to use *unstructured meshes* that conform to the irregular geometry of relevance [Bab70, BK96, WK99, CZ96, HZ01, LW04, Dry05, CGL09]. However, meshing complex geometries can prove difficult and, with frequent shape changes, time-consuming, especially in 3 dimensions. In the case of shape optimization for elastic materials [SW00, OS01, AJT04, DMJ06, CRW08, WW08], the task is further complicated when using the more elaborate element types seen in mixed finite element method formulations, which are typically necessary for stability in the nearly incompressible regime. Furthermore, many numerical methods, such as finite difference methods and geometric multigrid methods, do not naturally apply to unstructured meshes.

These concerns motivated the development of *embedded* (or *immersed*) methods, in which a structured mesh, such as a regular Cartesian grid, simply encompasses, rather than geometrically adheres to or conforms to, the irregular geometry. The irregular geometry is embedded within mesh elements: its location is tracked relative to the surrounding mesh. This avoids the complexities inherent in unstructured mesh generation while opening the door to the use of efficient solution techniques such as multigrid methods. Early research on embedded methods include works of Harlow and Welch [HW65], Peskin [Pes72], Hyman [Hym52], and Saul'ev [Sau63].

The remainder of this text is divided into two parts. Part I involves the application of the sophisticated mesh cutting algorithm of Sifakis et al. [SDF07] to crack propagation in 2 dimensions, virtual surgery, and object cracking and shattering for use in visual effects and computer animation. All of these applications involve the embedding of some open and/or non-manifold cut or crack surface within a regular background simplex mesh, with consequent topology change and duplication of degrees of freedom. We summarize the machinery enabling this embedding in the prelude to Part I. Part II discusses the solution of elliptic partial differential equations in an innovative embedded framework. We will specifically consider Poisson's equation with interfacial jump conditions and the equilibrium equations of linear elasticity. The feature set possessed by the described numerical methods to solve these partial differential equations has several advantages over, and compares quite favorably with, existing alternative methods.

Part I

Applications of Arbitrary Lagrangian Mesh Cutting

Introduction

Many physical simulations necessitate the modeling of fracture or crack surfaces, and in the most demanding of these simulations, these surfaces may have a highly complex non-manifold topology, e.g., with many branches and open "fronts". In a dynamics scenario, this may be further complicated by the frequent extension of existing crack fronts and the introduction of new failure surfaces. Applying a traditional finite element method in this context is challenging: one must either constantly regenerate the simulation mesh to account for the fracture geometry, which is computationally expensive and easily introduces ill-conditioned "sliver" elements; or one must artificially and often severely limit the potential paths and resolution of the failure surface to lie along element boundaries. Additionally, in either case, handling a high resolution fracture surface necessitates a correspondingly high resolution simulation volume, at least locally, which in turn increases the computational cost of solving the relevant discrete continuum mechanics equations. Ideally, one should be able to decouple the resolution of the fracture surface (which may be highly detailed for visual effects purposes, for example; see Chapter 3) from the resolution of the simulation mesh (which may be limited by available computational resources; see Chapter 2).

Given the above difficulties with traditional finite element methods, Belytschko, Black, Moës, Dolbow [BB99, MDB99] and others developed the *eXtended Finite Element Method* (XFEM), specifically in the context of modeling cracks, which avoids the need to remesh to capture crack geometry. The basic idea of the XFEM is to enrich the usual finite element spaces with additional degrees of freedom which incorporate the near tip asymptotic solutions and allow the displacements around the crack surface to be discontinuous. We hold off a complete introduction to and history of the XFEM until Chapter 1, but do mention one of the main challenges with utilizing the XFEM: automating the determination of material connectivity and subsequent enrichment of the finite element spaces. In the following chapters, we apply the mesh cutting algorithm of Sifakis et al. [SDF07] to resolve arbitrary Lagrangian cutting surfaces against the simulation volume mesh, to determine material connectivity, and to automatically duplicate mesh vertices to yield *virtual nodes* which effect the requisite enrichment necessary for separation.

We conclude this prelude to Part I with a brief summary of the mesh cutting algorithm from [SDF07], as the main subject of Part I is the various applications of this algorithm. Chapter 1 combines the XFEM with the mesh cutting algorithm to simulate propagating cracks in 2 dimensions; we also introduce a novel quadrature scheme to accurately and straightforwardly integrate the nonlinear and singular finite element basis functions. In Chapter 2, we consider the combination of this mesh cutting algorithm with nonlinear continuum mechanics in 3 dimensions to create a virtual surgery simulator. Lastly, in Chapter 3, we discuss a framework to systematically create cracked and shattered models for visual effects and computer animation.

Mesh Cutting Algorithm Overview

We now give a brief overview of the mesh cutting algorithm described by Sifakis et al.; see [SDF07] for more details, specifically in resolving the intra-simplex geometry, the discussion of which we exclude here. The essence of the algorithm may be described by considering the resolution of a segmented curve cutting surface against a triangulated area as the volumetric mesh, although the following overview applies equally well to higher dimensions (as seen in Chapters 2 and 3, for example). Ultimately, the algorithm produces a volumetric mesh geometrically coincident with the original (uncut) mesh with mesh elements along the cutting surface duplicated into topologically and materially disconnected counterparts; see Figure I.1 for an example overview of the entire procedure.

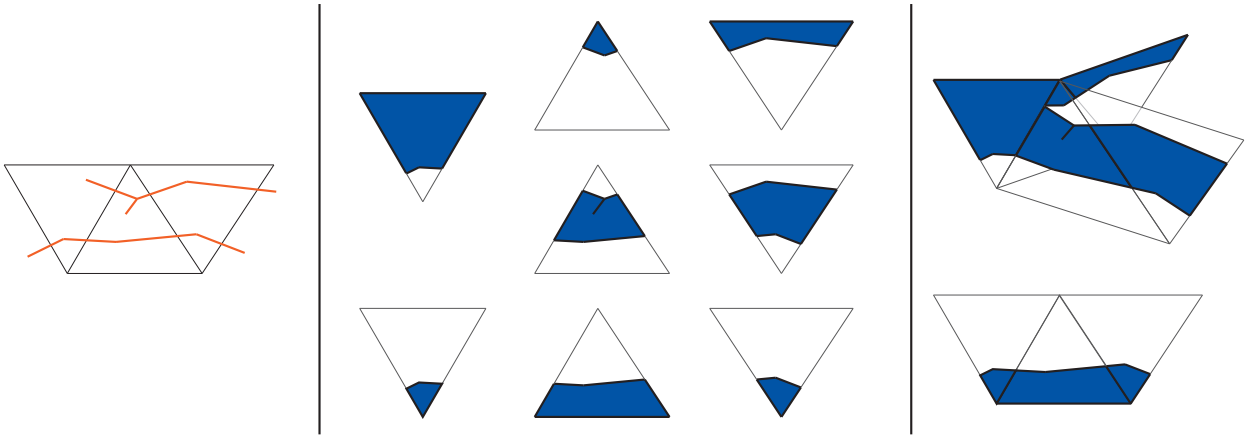


Figure I.1: This mesh is cut by two curves, one of which contains a branch (left). The cutting algorithm first treats each triangle individually, creating duplicates for each locally disjoint material region (center), and then uses the global mesh topology to join these duplicates on the proper degrees of freedom (right).

In the first phase, the algorithm processes each mesh triangle individually, identifying the disjoint *material components* the triangle is divided into by the cutting curve and describing each as a closed polygonal region (depicted blue in Figure I.2). For each such material region, a duplicate copy of the triangle is created and assigned said material region. For example, in Figure I.2, the cutting curve divides the triangle into two distinct material regions, inducing the creation of two duplicates of the original triangle with each duplicate possessing one of the material regions. In the duplicated triangles, we identify vertices within material regions (solid blue circles in Figure I.2) with the original triangle vertices. We also furnish the duplicate triangles with *virtual nodes* (hollow blue circles in Figure I.2). A less trivial example is given in Figure I.3.

After processing all triangles in the original mesh $\mathcal{T}$ individually, we obtain a duplicate mesh $\mathcal{T}'$ composed of duplicated triangles and vertices. The second phase of the algorithm proceeds to determine global material connectivity within this duplicate mesh (see Figure I.1). For notational convenience, let $C(T)$ denote the set of triangles duplicated from an

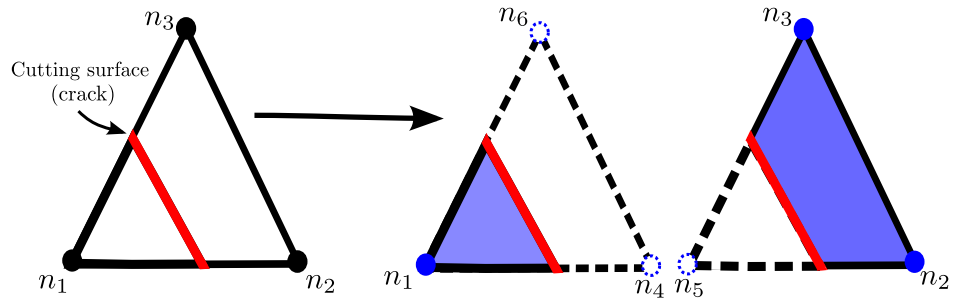


Figure I.2: Simple example of an original mesh triangle (left) duplicated with its disjoint *material regions* (blue) being distributed among the duplicates (middle, right). Likewise, the vertices in each duplicate are either identified with the vertices of the original triangle (*material nodes*, sold blue) or duplicate copies (*virtual nodes*, hollow blue), depending on whether they fall within a material region.

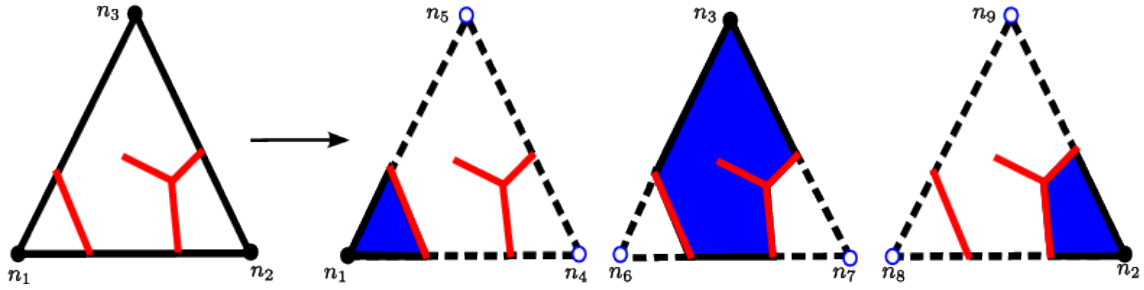


Figure I.3: A more complex example of the initial division of a triangle by a cutting curve. Since the original mesh triangle is divided into 3 disjoint material regions (left), the algorithm generates 3 duplicate triangles, each possessing a different material region (right). (Coloring is consistent with that used in Figure I.2.)

original triangle $T \in \mathcal{T}$, and let $P(T')$ denote the original "parent" triangle of a duplicate triangle $T' \in \mathcal{T}'$, so that $T' \in C(T)$ if and only if $T = P(T')$. Determining global material connectivity then proceeds as follows. Given a triangle $T' \in \mathcal{T}'$, let $T = P(T')$. Then for each $U' \in C(U)$ where U is face-adjacent to T, determine if U' shares *material connectivity* across the $T - U$ face with T'. If so, the vertices of the corresponding faces of T' and U' are identified as equivalent and collapsed, thus joining these duplicate triangles together.

Refer again to Figure I.1 for an example overview of the entire algorithm. The original mesh, at left, consists of three triangles. This mesh is cut by two segmented curves (red); the geometry typifies some of the subtleties in the algorithm, as the center triangle contains a branch, a tip, and is cut into multiple pieces. In the first phase of the algorithm, each triangle is processed in isolation and duplicated based on the disjoint material regions created by the cutting curves, as shown in the center of Figure I.1. In the second phase, these duplicate triangles are joined along faces where they share material connectivity, with the final mesh on the right of Figure I.1.

Notice that, in particular, if mesh vertices are identified with degrees of freedom, the latter mesh in Figure I.1 possesses the necessary richness in degrees of freedom to allow the upper crack to partially separate and the lower crack to separate entirely. This automatic generation of additional degrees of freedom is one of the primary motivations to combine this mesh cutting algorithm with an XFEM framework, and its fruits are demonstrated in the remainder of Part I.

CHAPTER 1

Crack Propagation in Two Dimensions

1.1 Background and Existing Methods

[1] Since our discretization is essentially an *eXtended Finite Element Method* (XFEM), we summarize the main idea and historical background of the XFEM; see [BGV09], [KX03], and [AH08] for more complete surveys. The idea is to enrich the usual finite element spaces with additional degrees of freedom, which incorporate the near-tip asymptotic solutions and allow the displacements to be discontinuous across the crack surface. The application of XFEM to cracks began with Belytschko and Black [BB99], where they applied the partition of unity methods (see, e.g.,[MB96]) to the problem of using finite elements with discontinuous basis functions. In [MDB99] Moes et al. used XFEM to create a technique for simulating crack propagation in two dimensions without remeshing the domain. Sukumar et al. [SMM00] began the extension to three dimensions. They used the two dimensional enrichment functions for planar cracks, and then further extended in [AB05].

Since its introduction, XFEM enrichment has been employed in a variety of settings to model fracture. Moes and Belytschko [Mo02] modelled cohesive fracture using special enrichments, and this was extended in [ZB03] and continues to be developed (see, e.g., [MP03, BGW04, APN07]). Work in other settings includes fracture with elastodynamics [BC04] and crack propagation in composite materials [HB09]. The XFEM has been combined naturally with the level set methods of Osher and Sethian [OS88, OF04] to track the moving discontinuity sets (for cracks see, e.g., [BMU01, MGB02, GMB02, Duf07, PCG07]; and for holes and inclusions see [SCM01]); Sukumar et al. [SBM08] coupled the XFEM with fast marching methods. Bordas et al. [BDL07] studied error estimates, and various techniques have increased the rate of convergence, such as cut off functions and geometric enrichment [CLR06, CLR08, SL09]. However, the XFEM approach still carries a couple technical challenges: assembling the stiffness matrix requires integration of singular or discontinuous functions; and implementing enrichment requires resolving material connectivity (often using a level set representation).

Quadrature for integration of the gradients of the XFEM basis functions is an active area of research because of the singularities and discontinuities present. As noted in [DMD00], the use of Gaussian quadrature or Monte Carlo integration is unstable: since the crack geometry within a given triangle is unknown a priori, quadrature points could be very close to singularities in the integrand. One approach to the problem (see, e.g., [SBC03]) is to perform a Delaunay triangulation on the cut triangle that respects the crack geometry and

[1] The content of this chapter is a version of [RHS11] with moderate revisions.

then use Gaussian quadrature on each of the resulting triangles. This triangulation does not produce additional degrees of freedom; it is only used for integration of the basis functions. Other methods, e.g., [BMM05, LPR05], map the near-tip enrichment functions to domains amenable to Gaussian quadrature, but also require meshing of the tip triangle. Another approach is to use higher order Gaussian quadrature [SBC00]. In [VGB09], Ventura et al. transformed the area integral required for assembly of the stiffness matrix into a more stable line integral. Park et al. [PPD09] also used a mapping technique to remove the singularity for tetrahedral elements (in three dimensions), while Areias and Belytschko [AB05] used a smoothing technique. For integrating the Heaviside functions, Ventura [Ven06] used a map to equivalent polynomials which were integrated using standard quadrature techniques; Holdych et al. [HNS08] used a similar technique where they introduced a dependence of the Gaussian quadrature weights on the position of the quadrature point within the triangle. Benvenuti et al. [BTV08] regularized the Heaviside function for integration with Gauss quadrature and proved that the solutions converge as the regularization parameter goes to zero. Mousavi and Sukumar [MS10] used a quadrature rule that avoids Delaunay triangulation and does not require splitting cut elements for Heaviside enrichment.

We introduce a simple method of integration (see §1.5) that combines naturally with the mesh cutting algorithm (see §1.4). Our scheme involves creating a finer mesh for integration purposes only. This is similar to the approach of Ji et al. [JCD02] and Dolbow [Dol99]; however, we resolve the crack surface inside the quadrature elements and use an approximation of the nonlinear basis functions for the purposes of quadrature. Mousavi et al. [MGS11] also use modified enrichment functions; however, they compute new enrichment functions by solving a partial differential equation, whereas we simply project the usual enrichment functions onto a simpler finite element space. Also in [MGS11], the authors solve for and integrate their enrichment functions on a mesh that is refined near the tip; since we also use such a mesh, it may be possible to incorporate their idea of computing the enrichment functions into our approach.

In order to allow cracks to open, XFEM needs to generate additional degrees of freedom, generally referred to as *enrichment*. In a region that has been unambiguously separated into two pieces (i.e., away from the crack tip), the enrichment is provided by a Heaviside function, defined to be $+1$ on one side of the crack surface and -1 on the other side. This is easy in the case of a single straight crack but more challenging as the crack geometry becomes complicated. Daux et al. [DMD00] handle the case of branched cracks by using separate enrichments for each crack, and then use another enrichment function to represent the junction itself. They then generalize this technique to cracks that have multiple branches; however, their method requires that the cracks have been hierarchically decomposed into a main crack and its branched components, and it still involves solving the problem of material connectivity. Budyn et al. [BZM04] and Zi et al. [ZSB04] extend [DMD00] to incorporate multiple cracks and to address the issue of intersecting cracks. Song and Belytschko [SB09a] introduced the cracking node method, which is based on XFEM and is designed to more easily handle complicated crack geometries.

The use of phantom, ghost, or virtual nodes (e.g., [MBF05]) to incorporate discontinuities has become increasingly popular. The methods of Hansbo and Hansbo [HH04], Song et al.

[SAB06], and Duan et al. [DSM09] (which are equivalent; see [AB06]) use a notion of ghost or phantom degrees of freedom to handle displacement discontinuities. Song and Belytschko also use a phantom node method in [SB09b], where they additionally use the product of multiple Heaviside functions to handle branched cracks. Dolbow and Harari [DH09] use phantom nodes in the context of embedded interface problems. We likewise use virtual nodes by leveraging the mesh cutting algorithm in [SDF07], making it possible to handle complex crack geometry (such as branching) systematically. Also, our method can create different finite element spaces than the methods presented in [HH04] or [SAB06] (see §1.5).

We present a method for simulating quasistatic crack propagation in 2 dimensions which combines the XFEM with a simple integration procedure and the geometrically flexible mesh cutting algorithm described in [SDF07]. To summarize, our approach

- is based on virtual nodes created by the mesh cutting algorithm that incorporates material connectivity;
- can handle complicated crack patterns (including multiple tips in the same element, branching, and tips within fully cut elements);
- can handle geometrically complex domains;
- does not require remeshing of the domain (which is entirely in the spirit of XFEM);
- employs a quadrature rule that again utilizes the mesh cutting algorithm, and whose degree of complexity is independent of the crack geometry.

1.2 Governing Equations

We assume quasistatic evolution, such that at each fixed point in time the material is in elastic equalibrium. Denoting the rest configuration by $\Omega \subset \mathbb{R}^2$ open and bounded, we thus consider the equations of elastic equilibrium given by

$$\begin{aligned} \nabla \cdot \boldsymbol{\sigma} + \mathbf{b} &= \mathbf{0} && \in \Omega \setminus \Gamma; \\ \mathbf{u} &= \mathbf{u}_0 && \in \partial\Omega_d; \\ \boldsymbol{\sigma} \cdot \hat{\mathbf{n}} &= \mathbf{g} && \in \partial\Omega_n; \\ \boldsymbol{\sigma} \cdot \hat{\mathbf{n}} &= \mathbf{0} && \in \Gamma^+; \\ \boldsymbol{\sigma} \cdot \hat{\mathbf{n}} &= \mathbf{0} && \in \Gamma^-; \end{aligned}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor; $\mathbf{b}$ is the body force per unit volume; $\mathbf{u}$ is the (unknown) displacement; $\mathbf{u}_0$ is the Dirichlet boundary condition (applied to a subset of the boundary $\partial\Omega_d$); $\mathbf{g}$ is the traction (Neumann) boundary condition (applied to $\partial\Omega_n$); $\hat{\mathbf{n}}$ denotes the unit outward-pointing normal; Γ is the crack surface; and Γ^+, Γ^- represent the two opposite orientations of the crack surface. In the present exposition, we consider the case of small

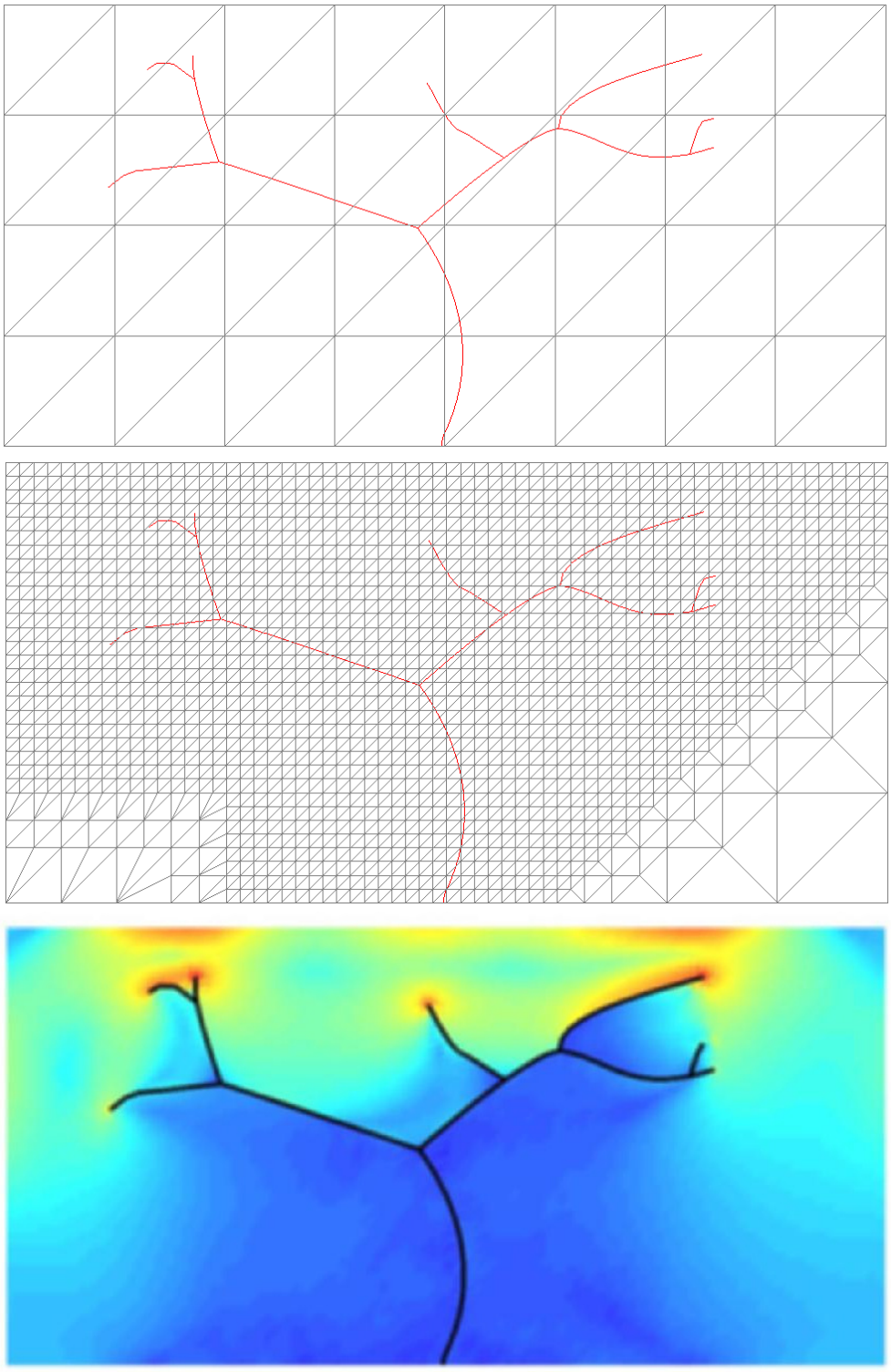


Figure 1.1: An example with a complex branching crack. (top) A crack surface cutting the simulation mesh; (center) the crack surface cutting the embedded quadrature mesh; (bottom) the computed stress field with uniform traction applied to the left and right edges.

strains and displacements, where linear elasticity is an accurate model of material behavior. Hence we use the Cauchy strain

$$\boldsymbol{\epsilon}(\mathbf{u}) := \nabla_S \mathbf{u} := \frac{1}{2}\left(\nabla \mathbf{u} + (\nabla \mathbf{u})^t\right)$$

and stress

$$\boldsymbol{\sigma} := \mathbf{C} : \boldsymbol{\epsilon}$$

where $\mathbf{C}$ is the Hooke tensor. Equivalently, one may consider minimizing over $\mathbf{u}$ the potential energy

$$\Psi[\mathbf{u}] := \frac{1}{2}\int_\Omega \boldsymbol{\epsilon}(\mathbf{u}) : \mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u}) d\mathbf{x} - \int_\Omega \mathbf{b}\cdot\mathbf{u} d\mathbf{x} - \int_{\partial\Omega_n} \mathbf{g}\cdot\mathbf{u} d\mathbf{S}(\mathbf{x}) \tag{1.1}$$

subject to $\mathbf{u} = \mathbf{u}_0$ on $\partial\Omega_d$.

1.3 Extended Finite Elements

Conceptually, the starting point of our work is the eXtended Finite Element Method (XFEM), which was originally motivated and studied in the context of fracture by Belytschko et al.; see, e.g., [MDB99]. In the original XFEM, a simulation with quasistatics evolution requires one to solve a discrete approximation to the equations of elastic equilibrium at each time step. The approximation subspace is formed by taking the usual C_0 conforming finite element space (in our case, on triangles) and *enriching* with additional degrees of freedom that allow cracks to open and increase the accuracy of the approximation near the crack tip. Thus, functions in an XFEM space $\mathcal{U}^h$ have the form

$$\mathbf{u}^h(\mathbf{x}) = \sum_i \mathbf{u}_i\phi_i(\mathbf{x}) + \sum_j \mathbf{b}_j\phi_j(\mathbf{x})H(\mathbf{x}) + \sum_k \phi_k(\mathbf{x})\sum_{\ell=1}^{4}\mathbf{c}_k^\ell F_\ell\left(r(\mathbf{x}),\theta(\mathbf{x})\right), \tag{1.2}$$

where $\{\phi_i\}$ are the usual nodal basis functions; $H(\mathbf{x})$ is the Heaviside function associated to the current crack geometry; $\{\mathbf{b}_j\}$ are enrichment degrees of freedom associated with crack separation away from the crack tip; $\{\mathbf{c}_k^\ell\}$ are enrichment degrees of freedom associated with near-tip displacement; and

$$\{F_\ell(r,\theta)\} := \left\{\sqrt{r}\sin\frac{\theta}{2}, \sqrt{r}\cos\frac{\theta}{2}, \sqrt{r}\sin\frac{\theta}{2}\sin\theta, \sqrt{4}\cos\frac{\theta}{2}\sin\theta\right\}$$

are the asymptotic near-tip enrichment functions (r and θ are the polar coordinates with respect to the crack tip). Notice that (1.2) expresses $\mathbf{u}^h$ as a linear combination of three types of basis functions: the usual nodal basis functions (which have support local to mesh vertices), Heaviside enrichment functions, and near-tip enrichment functions (which have support local to the crack tip). The sum over i in (1.2) is over all mesh vertices, while the sums over j and k are over those vertices whose corresponding conforming basis functions have support that intersects the crack surface (see [MDB99]). We use the term *one-ring* of a vertex to refer to the set of elements composing the support of the corresponding nodal basis function. Strictly speaking, (1.2) only accounts for a single crack tip, but it may be generalized to accomodate crack geometries with multiple tips.

1.4 Cutting of Cracked Domains

To simplify the exposition, we first focus on the discretization away from any crack tips (thus ignoring tip enrichment) where (1.2) takes the simpler form

$$\mathbf{u}^h(\mathbf{x}) = \sum_i \mathbf{u}_i \phi_i(\mathbf{x}) + \sum_{j \in J} \mathbf{b}_j \phi_j(\mathbf{x}) H(\mathbf{x}),$$

where J is the set of vertices whose one-ring intersects the crack surface. We begin by replacing these more traditional Heaviside-enriched degrees of freedom with more geometrically intuitive virtual nodes (see [MBF05]), also known as ghost or phantom nodes (see, e.g., [SAB06], [HH04], [DH09]). As discussed in the prelude to Part I, these virtual nodes are automatically created by the mesh cutting algorithm of Sifakis et al. [SDF07]. Figures 1.2 and 1.3 illustrate an example of the mesh cutting algorithm more closely related to our present application. In Figure 1.2, we introduce a crack which completely cuts one triangle and only partially cuts another. In Figure 1.3, we introduce the same crack into a locally refined mesh. In the present context, the unrefined mesh corresponds to the simulation mesh, while the refined mesh – after being cut as in the example – corresponds to the quadrature mesh (see §1.5).

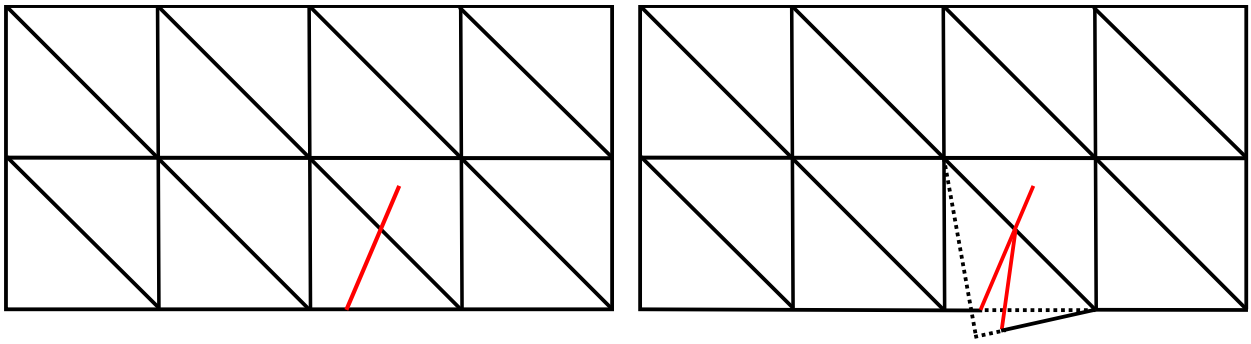


Figure 1.2: A crack is introduced into the mesh on the left, yielding the enriched mesh on the right, with duplicated triangles and virtual nodes, which we use as our simulation mesh.

Now, consider a mesh that has been cut, with the associated duplication of elements and introduction of virtual nodes. Corresponding to each virtual node, we create a nodal basis function that respects the crack geometry, i.e., we take into account that vertices and triangles may have been duplicated (see Figure 1.4 for a one-dimensional illustration). Let $\{\tilde{\phi}_i\}$ be the usual piecewise affine nodal basis "hat" functions on the simulation mesh, and let $\mathbf{x}_i$ be the vertex (which may be a virtual node) corresponding to a given $\tilde{\phi}_i$. Denote by Ω_i the collection of triangles in the one-ring of vertex $\mathbf{x}_i$. Given the preceding notation, we define a new *truncated* hat function ϕ_i via

$$\phi_i(\mathbf{x}) := \tilde{\phi}_i(\mathbf{x}) \sum_{T \in \omega_i} \chi_T^M(\mathbf{x}), \tag{1.3}$$

where χ_T^M is the characteristic function of the material region of triangle T.

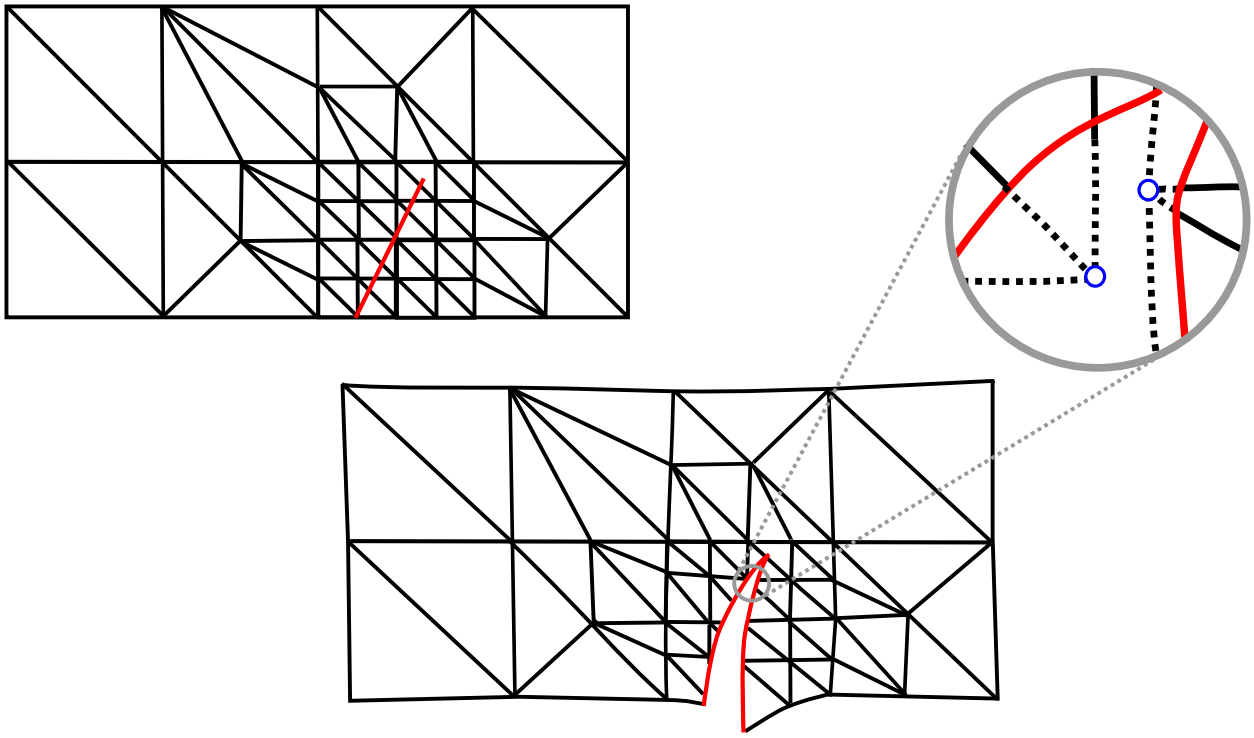


Figure 1.3: For quadrature purposes, we refine the uncut mesh in Figure 1.2(left) around the cut (upper left), then cut this refined mesh (bottom). A blow up of the region near the crack (upper right) shows the virtual nodes relative to the crack surface.

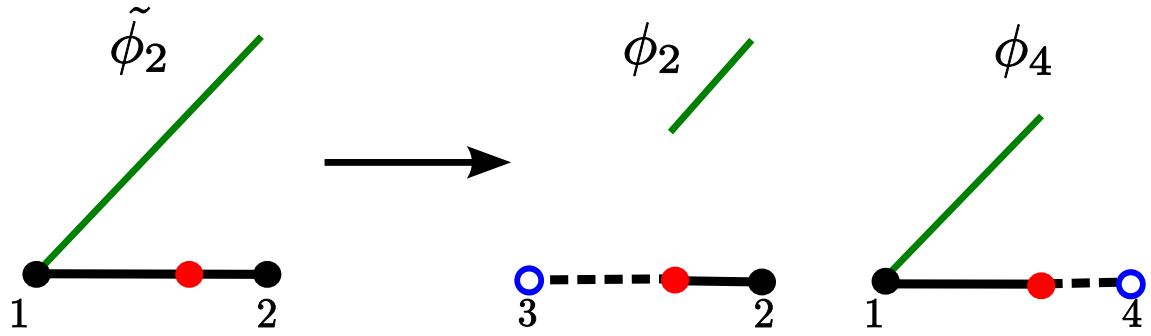


Figure 1.4: A truncated basis function in dimension 1. At left, a mesh element with vertices 1 and 2 is cut by the red dot, resulting in the two duplicate elements on the right. The original nodal basis "hat" function $\tilde{\phi}_2$ corresponding to vertex 2 (green) gets truncated into derived basis functions ϕ_2 and ϕ_4 according the material region within each duplicate element.

In many cases, this virtual node approach is equivalent to the more traditional use of Heaviside enrichment as well as the methods of Song et al. [SAB06] and Hansbo and Hansbo [HH04], i.e., they give equivalent finite element spaces. However, certain crack geometries do, in fact, yield different spaces. As an example, the configuration in Figure 1.5(left) is cut by a crack given by the red line. The resulting finite element spaces differ between the two methods. Heaviside enrichment and the methods of Song et al. [SAB06] and Hansbo and Hansbo [HH04] yield the degrees of freedom depicted in Figure 1.5(center), while the mesh cutting algorithm from [SDF07] yields the degrees of freedom depicted in Figure 1.5. Since the material region of the top triangle to the right of the crack surface is not materially connected to the bottom triangle, the bottom left virtual nodes are allowed to separate (see top-right of Figure 1.5). Generally speaking, the finite element spaces resulting from the mesh cutting algorithm are at least as rich as the spaces resulting from Heaviside enrichment, and in some situations it may in fact be strictly richer (i.e., have strictly larger dimension).

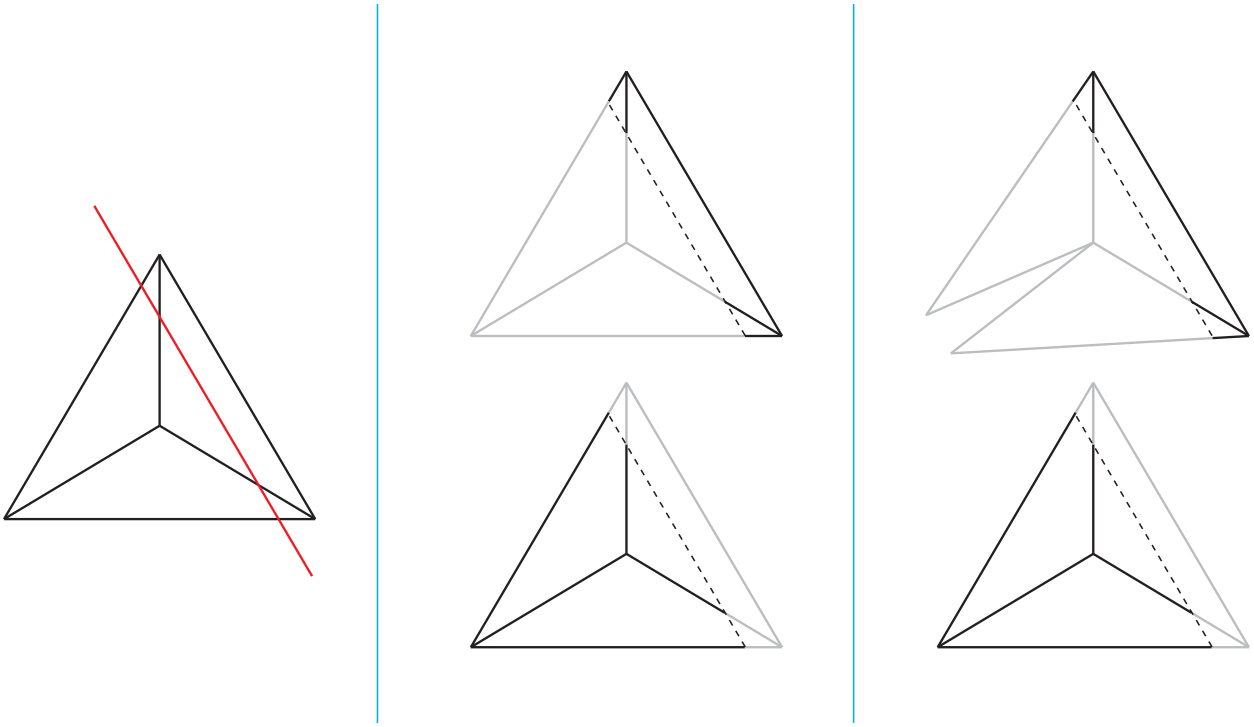


Figure 1.5: Given the uncut mesh and crack at left, XFEM using traditional Heaviside enrichment yields 8×2 degrees of freedom (center), while using virtual nodes as described in §1.4 yields 9×2 degrees of freedom (right).

Finally, we note that Sifakis et al. [SDF07], in their presentation of the mesh cutting algorithm, make the simplifying assumption that the cutting surface never coincides with a mesh vertex or aligns with an element face. Nevertheless, the virtual node algorithm is perfectly compatible with such degenerate cases, and this simplifying hypothesis is made only to ease certain implementation challenges related to the representation of the crack surface as an explicit simplex mesh. In fact, the mesh cutting algorithm can accommodate any cut configuration, including these degenerate cases, as long as the following two queries

can be algorithmically determined: (a) the number of disjoints fragments a given element is divided into; and (b) material connectivity of element fragments originating from face-adjacent elements.

For example, if the crack surface is instead represented implicitly as the zero isocontour of a level set function sampled at the vertices of the simulation mesh (or a refinement thereof), both of the above queries admit a straightforward algorithm determination, and hence the element duplication and joining algorithm described in [SDF07] may be applied, even when the zero isocontour coincides with a mesh vertex. In the present case, with the crack surface represented explicitly as a simplex mesh (segmented curve in 2 dimensions or triangulated surface in 3 dimensions), the robust determination of (a) and (b) is challenging given the limited precision of typical floating point arithmetic. Rather than complicate the implementation to address degenerate or near-degenerate scenarios, we found it preferable and practical to avoid these scenarios entirely by simply perturbing the crack surface negligibly. Naturally, if we use a level set representation, no such perturbation is necessary.

1.5 Integration

Our integration scheme utilizes a subordinate quadrature mesh in addition to the primary simulation mesh, the details of which we describe here. For simplicity, we assume a domain-conforming triangulation, creating a mesh corresponding to Figure 1.2(left). We then construct a pair of derived meshes:

- We apply the mesh cutting algorithm to resolve the crack surface against the original mesh, yielding the *simulation mesh*, corresponding to Figure 1.2(right). The simulation mesh contains the actual simulation degrees of freedom, include both virtual and crack tip enrichment degrees of freedom.

- We also locally refine the original mesh around the crack and resolve the crack against this refined mesh, yielding the *quadrature mesh*, corresponding to Figure 1.3(right). The quadrature mesh only possesses nodal degrees of freedom (i.e., no crack tip enrichment degrees of freedom), and these degrees of freedom are subordinate to the interpolated values from the simulation mesh. We only use the quadrature mesh to aid in the integrations involving the enrichment functions on the simulation mesh, and it does not add additional degrees of freedom to the system.

We solve the equilibrium equations of linear elasticity on the relatively coarse simulation mesh (with fewer degrees of freedom), but perform the requisite integrations on the relatively fine quadrature mesh by approximating the (generally nonlinear) basis functions over the simulation mesh with piecewise affine projections over the quadrature mesh. We then use a composite one-point quadrature rule over the quadrature mesh.

Our integration scheme is similar to another scheme presented in the XFEM literature (e.g., [SBC03]). The main idea is to compute a Delaunay triangulation respecting the crack geometry within each simulation triangle and assembling the stiffness matrix by applying

Gaussian quadrature over each Delaunay triangle. Hence, like our scheme, this employs a finer triangulation of the original mesh solely for integration purposes and does not add degrees of freedom to the simulation. However, our integration scheme uses the mesh cutting algorithm [SDF07] to resolve the crack geometry within the quadrature mesh (see, e.g., Figure 1.1(center)), and so, in contrast to Delaunay triangulation, the quadrature mesh will only approximately conform to the crack geometry. This has the advantage of decoupling the resolution of the quadrature mesh from the resolution of the crack surface. Further, this use of the mesh cutting algorithm naturally extends to higher dimensions, where Delaunay tessellations become significantly more challenging.

1.5.1 Construction of the Simulation Mesh and Quadrature Mesh

We now describe the construction of our key meshes in more detail, and we will use notation consistent with [Bra07]. We construct the simulation mesh as described in §1.4. Using the modified hat functions from (1.3), we define our (simulation) finite element space $\mathbf{V}^h$ as those vector-valued functions $\mathbf{u}^h$ of the form

$$\mathbf{u}^h(\mathbf{x}) = \sum_i \mathbf{u}_i \phi_i(\mathbf{x}) + \sum_k \phi_k(\mathbf{x}) \sum_{\ell=1}^{4} \mathbf{c}_k^\ell F_\ell \left(r(\mathbf{x}), \theta(\mathbf{x})\right) \tag{1.4}$$

where $\{F_\ell\}$ are the asymptotic near-tip enrichment functions given in §1.3. The difference between (1.4) and (1.2) is that (1.4) already incorporates the Heaviside enrichment (the sum over j in (1.2)) via the introduction of virtual nodes and the truncated basis functions. We let $F^h := \{\mathbf{u}_i, \mathbf{c}_k^\ell\}$ denote the degrees of freedom of $\mathbf{V}^h$ and identify F^h with $\mathbb{R}^N$, where N is the number of degrees of freedom in the simulation.

We now discuss the construction of the quadrature mesh. We begin by regularly refining triangles in the original mesh that intersect the crack surface. This refinement is progressively graded as the distance from the crack increases via red-green refinement [MBT03]. We resolve the crack surface against this refined mesh via the mesh cutting algorithm to yield the quadrature mesh; this ensures the quadrature mesh will respect the crack topology (see Figure 1.3). We then define a quadrature finite element space $\mathbf{V}_q^h$ to be the piecewise affine finite element space over this quadrature mesh, i.e., $\mathbf{u}_q^h \in \mathbf{V}_q^h$ takes the form

$$\mathbf{u}_q^h(\mathbf{x}) = \sum_i \mathbf{u}_i^q \phi_i^q(\mathbf{x}) \tag{1.5}$$

where $\{\phi_i^q\}$ are the associated truncated nodal basis functions (as in (1.3)). We let $F_q^h := \{\mathbf{u}_i^q\}$ denote the degrees of freedom of $\mathbf{V}_q^h$ and identify F_q^h with $\mathbb{R}^M$, where M is twice the number of vertices in the quadrature mesh. Since we do not employ the asymptotic near-tip enrichment functions in the quadrature finite element space (as we do for the simulation finite element space), all the degrees of freedom F_q^h can be identified with a vertex in the quadrature mesh and a coordinate direction.

We use the quadrature mesh and associated finite element space to approximate the integrals of the gradients of the $\mathbf{V}^h$-basis functions. To this end, we subordinate the quadrature

mesh to the simulation mesh via a fixed linear relationship between F_q^h and F^h (see [SSI07]). Consider $\mathbf{u}_i^q \in F_q^h$ and the position of its corresponding mesh node $\mathbf{x}_i^q$. We express $\mathbf{u}_i^q$ in terms of the simulation degrees of freedom F^h using (1.4):

$$\mathbf{u}_i^q = \sum_j \mathbf{u}_j \phi_j(\mathbf{x}_i^q) + \sum_k \phi_k(\mathbf{x}_i^q) \sum_{\ell=1}^{4} \mathbf{c}_k^\ell F_\ell \left(r(\mathbf{x}_i^q), \theta(\mathbf{x}_i^q) \right). \tag{1.6}$$

Note that the sume over j involves at most 3 nonzero terms as $\mathbf{x}_i^q$ can be in the support of at most three of the ϕ_j's. According to (1.6), the quadrature degrees of freedom $\mathbf{u}_i^q$ are functionally constrained to the simulation degrees of freedom $\{\mathbf{u}_i, \mathbf{c}_k^\ell\}$ and hence do not introduce any new degrees of freedom. This effectively defines a linear relationship between the quadrature and simulation degrees of freedom which we denote by the matrix W (see below). Binding the quadrature mesh to the simulation mesh in this fashion allows us to project the basis functions of $\mathbf{V}^h$ onto $\mathbf{V}_q^h$ and operate on these projections in the piecewise affine quadrature finite element space, where the integration is simpler.

Special care must be taken when computing the polar coordinates $(r(\mathbf{x}_i^q), \theta(\mathbf{x}_i^q))$ of $\mathbf{x}_i^q$ when it corresponds to a virtual node. As illustrated in Figure 1.6, for virtual nodes, we must reverse the orientation of the angle θ with respect to the crack, allowing it to take values outside the typical $[-\pi, +\pi]$ bounds. We effectively associate a virtual node with the opposite side of the crack surface, where the material region associate with the virtual node resides. This ensures that $\theta(\mathbf{x})$ is continuous throughout the triangle.

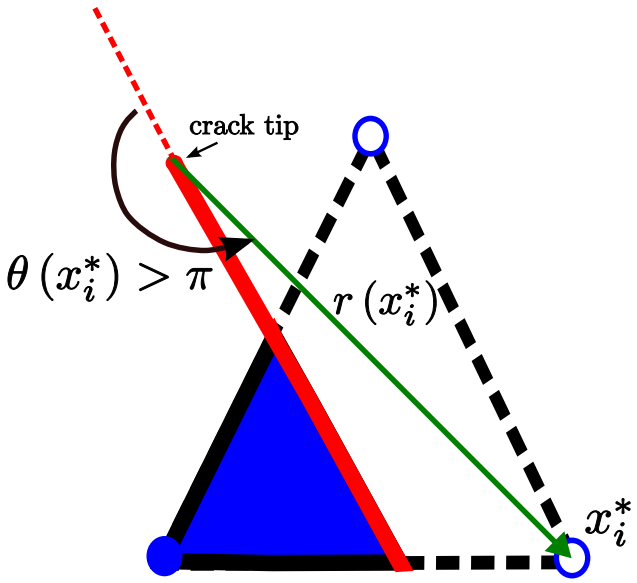


Figure 1.6: The polar angle of a virtual node in the quadrature mesh is measured across the crack surface to ensure continuity of θ throughout a quadrature triangle.

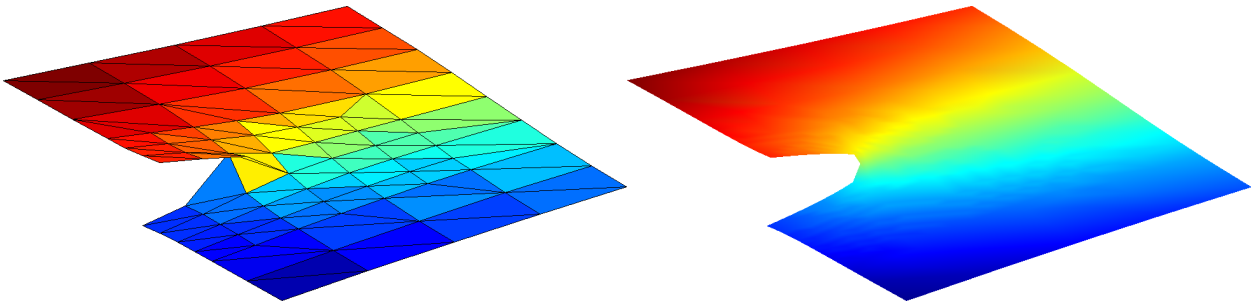
1.5.2 Integration Scheme

The relation (1.6) binds the quadrature degrees of freedom F_q^h to the simulation degrees of freedom F^h. Letting $\vec{u} \in F^h$ and $\vec{u}^q \in F_q^h$, we can encode the coefficients of (1.6) in a matrix W, such that $\vec{u}^q = W\vec{u}$. Let A and A^q denote the stiffness matrices associated with the energy (1.1) discretized over the finite element spaces $\mathbf{V}^h$ and $\mathbf{V}_q^h$, respectively; and let α denote the bilinear form associated with (1.1). Since $\mathbf{V}_q^h$ is a piecewise affine finite element space, there is a natural and standard procedure to assemble its stiffness matrix A^q. Our integration scheme then approximates the true stiffness matrix A via

$$\vec{u}^t A\vec{u} = \alpha(\mathbf{u}^h, \mathbf{u}^h) \approx \alpha(\mathbf{u}_q^h, \mathbf{u}_q^h) = (\vec{u}^q)^t A^q \vec{u}^q = \vec{u}^t W^t A^q W\vec{u}, \tag{1.7}$$

where $\mathbf{u}^h$ ($\mathbf{u}_q^h$) is the function in $\mathbf{V}^h$ ($\mathbf{V}_q^h$) corresponding to the vector $\vec{u} \in F^h$ ($\vec{u}^q \in F_q^h$). It follows that $A \approx W^t A^q W$.

Ultimately, our integration scheme utilizes an approximation (indeed, a projection) of the non-smooth basis functions (see Figure 1.7), with this approximation improving as one further refines the quadrature mesh. Note that we may end up sampling a singular basis function near the singularity. However, unlike integration schemes based on Gaussian quadrature or Monte Carlo methods, any function evaluations at these sampling points will be weighted by the area of the smaller quadrature triangle. Thus, no single sample, possibly located near the tip singularity, contributes disproportionally. Finally, we note that for simplicity and improved stability, we can further approximate the integrations implicit in (1.7) by treating cut quadrature triangles as if they were full of material, i.e., removing the characteristic function multiplications in (1.3) and assembling A^q over the usual nodal basis "hat" functions. This optional modification vanishes under refinement of the quadrature mesh, as the error caused by the additional basis function support goes to zero. In §1.7, our examples account for the material region in cut quadrature triangles. We additionally present some comparisons between treating quadrature triangles as completely full and respecting the actual material regions.



(a) (b)

Figure 1.7: Two samplings of the asymptotic near-tip enrichment function $F_1 = \sqrt{r}\sin\frac{\theta}{2}$. (a) shows a sampling at a low resolution, while (b) shows a sampling at a high resolution.

1.6 Crack Propagation

For a fixed state of the system (a given crack and boundary conditions and associated equilibrium displacement), engineers use several different criteria to determine the angle at which the crack should propagate. We follow Moës et al. [MDB99] in using the maximum circumferential stress criterion to compute the propagation direction and then move the crack by a small fixed increment. We choose this approach so that the results of using our integration technique can be compared against the test cases in [MDB99]. This method is fairly standard and the details are found in the references, so we only sketch it here.

The criterion involves computing the stress intensity factors at the crack tip, and then calculating the angle of maximal stress via

$$\theta_c := 2 \arctan \left(\frac{1}{4} \left(\frac{K_{\mathrm{I}}}{K_{\mathrm{II}}} \pm \sqrt{\left(\frac{K_{\mathrm{I}}}{K_{\mathrm{II}}} \right)^2 + 8} \right) \right).$$

We compute the stress intensity factors using the so-called interaction J-integral, which is defined for two possible states of the system, which we denote using superscripts 1 and 2:

$$I^{(1,2)} := \int_\Gamma \left(W^{(1,2)} \delta_{1j} - \left(\sigma_{ij}^{(1)} u_{i,1}^{(2)} + \sigma_{ij}^{(2)} u_{i,1}^{(1)} \right) \right) n_j d\mathbf{S} \tag{1.8}$$

where

$$W^{(1,2)} := \sigma_{ij}^{(1)} \epsilon_{ij}^{(2)}.$$

Choosing the two states to be the current state and a pure Mode I state in the above gives K_{I}:

$$K_{\mathrm{I}} := \frac{1}{2} E^* I^{(\text{current,Mode I})},$$

where

$$E^* = \begin{cases} \dfrac{E}{1-\nu^2}, & \text{plane strain} \\ E, & \text{plane stress} \end{cases},$$

and E is Young's modulus and ν is Poisson's ratio. K_{II} is found using a similar relation but with a pure Mode II state. As in [MDB99], we compute these interaction integrals by converting them into area intregrals via multiplication by a suitably smooth test function and applying integration by parts. We then compute the resulting area integral on the quadrature mesh described in §1.5. Having used finite elements to compute the displacement, the stresses and strains for the current state of the system are piecewise affine on the quadrature mesh. We then interpolate the displacements, strains, and stresses for the pure Mode I and pure Mode II solutions used to compute (1.8) using functions in the space $\mathbf{V}_q^h$. The required integrations are then simple to compute, since all the quantities in (1.8) are piecewise affine.

1.7 Numerical Examples and Experiments

We tested our approach with some examples from the literature ([BB99] and [MDB99]). We chose these examples because the exact stress intensity factors (or good approximations) can be calculated analytically for comparision; we also compare our results to the literature.

As discussed in Section 1.5, we computed our approximate solutions by integrating over just the material regions of cut quadrature triangles. For Example 1 below, we also compare this with integrating over the entire area of cut quadrature triangles.

As in [MDB99], all of our examples use a Young's modulus of $E := 10^5$ and Poisson's ratio of $\nu := 0.3$. For our propagation examples, we chose a fracture toughness of 1.

1.7.1 Example 1: Straight Crack with Pure Mode I Displacement

Example 1 involves a straight center crack in a rectangular body with a constant traction applied to part of the boundary of the body; see Figure 1.8 (as in [MDB99], we use $L := 16$, $W := 7$, $a := 3.5$, $\epsilon := 100$[kpsi] and $\nu := 0.3$).

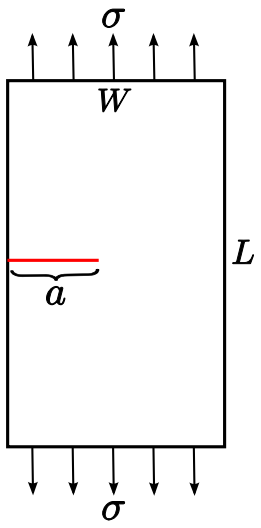


Figure 1.8: The setup for Example 1.

In this case, the exact Mode I stress intensity factor is given by

$$K_{\mathrm{I}} = C\sigma\sqrt{a\pi},$$

where C is a finite geometry correction factor:

$$C := 1.12 - 0.231\left(\frac{a}{W}\right) + 10.55\left(\frac{a}{W}\right)^2 - 21.72\left(\frac{a}{W}\right)^3 + 30.39\left(\frac{a}{W}\right)^4.$$

We normalize K_{I} through an appropriate choice of σ and compare our results over various combinations of granularity of the simulation mesh and refinement level of the quadrature

mesh. Table 1.1 contains the results of this study. The resolution of the simulation mesh varies over the columns while the refinement level of the quadrature mesh varies over the rows. Note that the numerical stress intensity factors improve as the simulation mesh is refined, as expected, but we also get good results by pairing a coarse simulation mesh with a relatively refined quadrature mesh (of course, only up to a limit that is determined by the simulation resolution). Also, increasing the refinement level of the quadrature mesh has roughly the same effect as increasing the resolution of the simulation mesh, and that accuracy improvement is acheived at a lower computational cost since refining the quadrature mesh does not add new degrees of freedom to the system. Figure 1.9 illustrates the results of three convergence tests for this example (using 1, 3, and 5 levels of quadrature mesh refinement). These plots clearly show first order convergence respect to the simulation mesh resolution. Finally, Figures 1.10 and 1.11 compare the use of respecting the actual material regions in the quadrature against treating the quadrature triangles as completely full. Both of these approaches give linear convergence, with slightly different constants. Further, the difference between those two methods goes to zero under refinement of the quadrature mesh (keeping the resolution of the simulation mesh fixed).

	Levels	64×32	128×64	256×128	512×256
K_{I}	0	7.25207	3.02688	1.68144	1.22950
	1	0.87748	0.93603	0.96710	0.98310
	2	0.90933	0.95357	0.97632	0.98783
	3	0.92324	0.96132	0.98039	0.98991
	4	0.93210	0.96621	0.98296	0.99122
	5	0.93858	0.96979	0.98482	0.99217
K_{II}	0	17.22220	5.78543	2.01135	0.70673
	1	−0.00252	−0.00117	−0.00068	−0.00037
	2	0.00003	0.000004	−0.00010	−0.00008
	3	−0.00210	−0.00091	−0.00052	−0.00028
	4	−0.00245	−0.00107	−0.00059	−0.00031
	5	−0.00267	−0.00117	−0.00063	−0.00034

Table 1.1: Results for Example 1. Theory gives $K_{\mathrm{I}} = 1$, $K_{\mathrm{II}} = 0$. "Levels" refers to the refinement level of the quadrature mesh.

1.7.2 Example 2: Straight Crack with Constant Shear Displacement

Example 2 uses the same geometric configuration as Example 1, but this time we apply a zero displacement Dirichlet boundary condition to one end of the domain and a constant shear (with respect to the crack frame) traction boundary condition to the other end; see Figure 1.12. The stress intensity factors are known (see [MDB99]): $K_{\mathrm{I}} = 34.0\,[\mathrm{psi}\sqrt{\mathrm{in}}]$ and $K_{\mathrm{II}} = 4.55[\mathrm{psi}\sqrt{\mathrm{in}}]$. As for Example 1, we vary both the resolution of the simulation mesh and the refinement level of the quadrature mesh, with the results summarized in Table 1.2 and convergence plots in Figure 1.13.

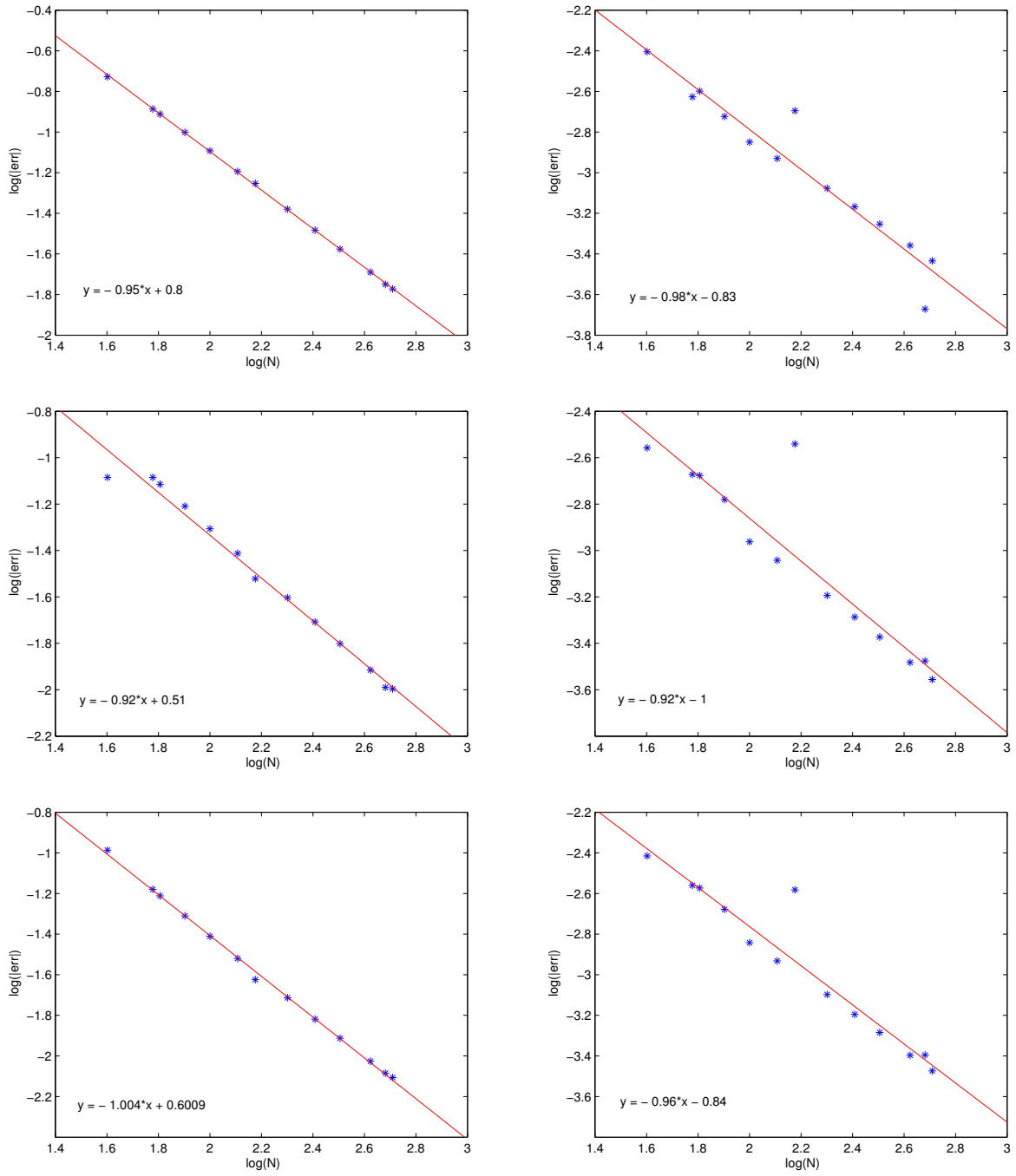


Figure 1.9: Log-log convergence plots of the errors in K_{I} (left) and K_{II} (right) for Example 1. Each row corresponds to a fixed refinement level of the quadrature mesh: 1 level (top), 3 levels (center), and 5 levels (bottom). All the linear regressions fit to the plots have slopes close to -1, incidating first order convergence.

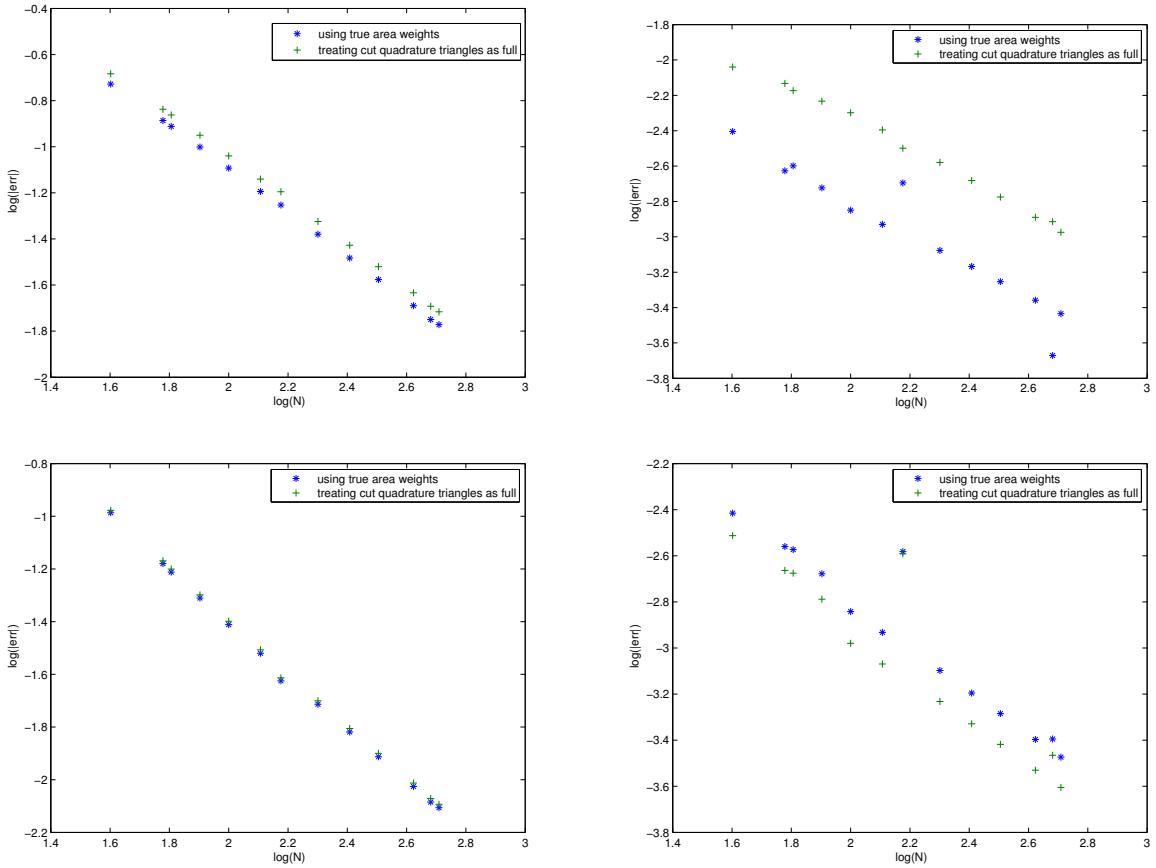


Figure 1.10: Log-log convergence plots of the errors in K_{I} (left) and K_{II} (right) for Example 1. Each row corresponds to a fixed refinement level of the quadrature mesh: 1 level (top) and 5 levels (bottom). Each plot compares respecting the material region within a cut quadrature triangle during integration against treating the quadrature triangle as completely full. The results indicate first order convergence regardless of the method.

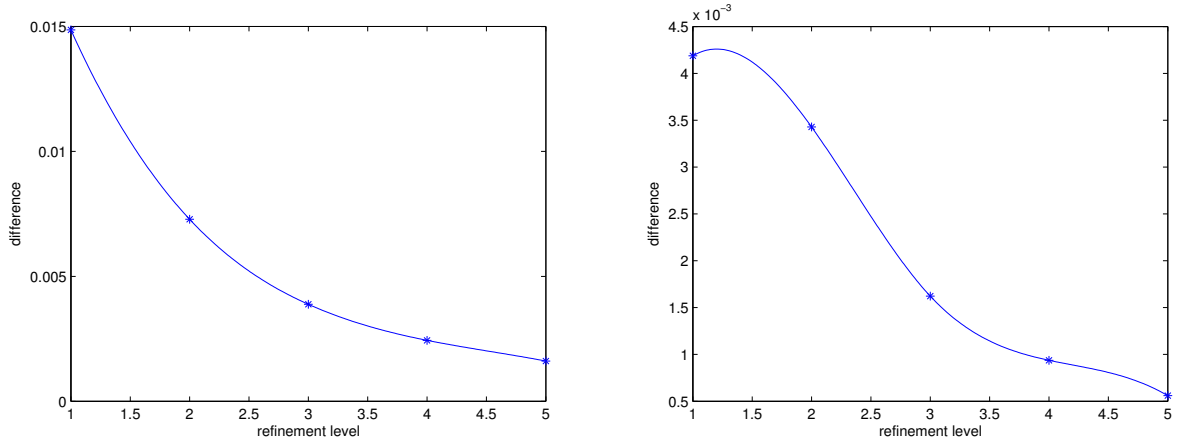


Figure 1.11: Plots illustrating the difference betwen respecting the material region within a cut quadrature triangle during integration against treating the quadrature triangle as completely full. We show the differences between K_{I} (left) and K_{II} (right) as we vary the quadrature mesh refinement level on a 64×32 resolution simulation mesh.

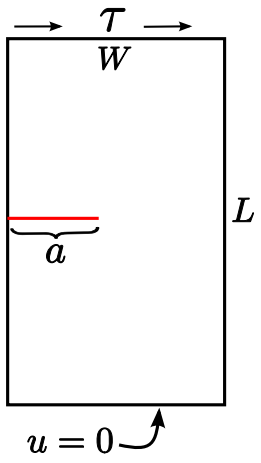


Figure 1.12: The setup for Example 2.

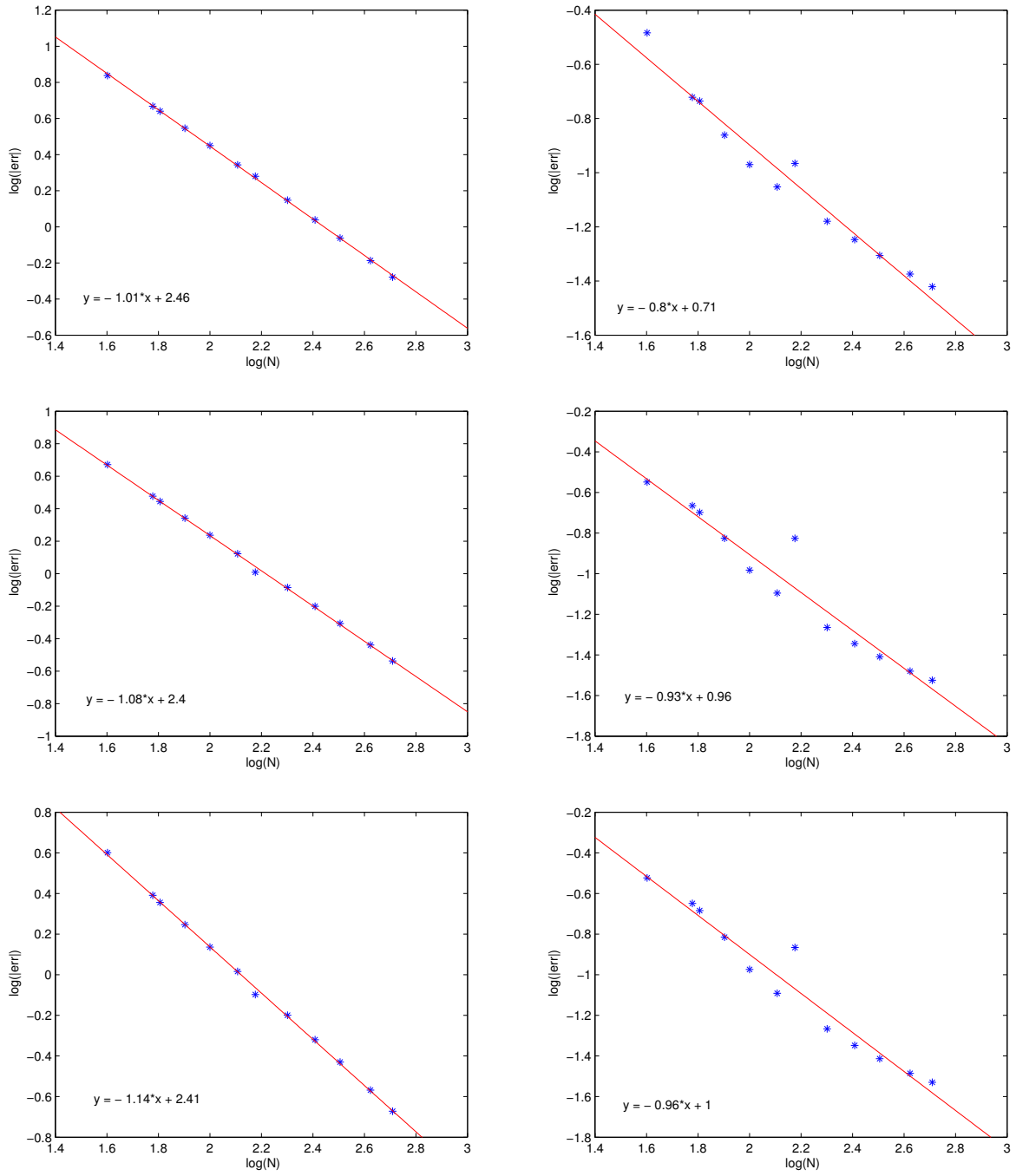


Figure 1.13: Log-log convergence plots of the errors in K_{I} (left) and K_{II} (right) for Example 2. Each row corresponds to a fixed refinement level of the quadrature mesh: 1 level (top), 3 levels (center), and 5 levels (bottom). All the linear regressions fit to the plots have slopes very close to -1, incidating first order convergence.

	Levels	64×32	128×64	256×128	512×256
K_{I}	0	27.53640	30.42170	32.13890	33.06540
	1	29.62800	31.79550	32.90860	33.47320
	2	30.72490	32.40350	33.22860	33.63730
	3	31.21530	32.67390	33.37000	33.70950
	4	31.51230	32.83990	33.45720	33.75420
	5	31.73140	32.96180	33.52120	33.78710
K_{II}	0	7.81139	5.55605	4.86779	4.64271
	1	4.36623	4.46139	4.49335	4.51208
	2	4.40807	4.49142	4.51380	4.52418
	3	4.34959	4.46972	4.50476	4.52010
	4	4.34610	4.46936	4.50504	4.52036
	5	4.34312	4.46894	4.50513	4.52048

Table 1.2: Results for Example 2. Theory gives $K_{\mathrm{I}} = 34.0$, $K_{\mathrm{II}} = 4.55$. "Levels" refers to the refinement level of the quadrature mesh. Note that most of the benefits of the quadrature mesh are realized after only 2 or 3 levels of refinement; beyond that the approximation error from the simulation finite element space dominates the integration error from utilizing the quadrature mesh.

1.7.3 Example 3: Angled Center Crack with Mixed Mode Displacement

We borrow the example in Section 4.3 of [MDB99] for Example 3. The domain is a square plate with an angled center crack which is subjected to a far field constant traction; see Figure 1.14. We use the parameters $W = 10$[in] and $a = 0.5$[in]. Since the crack size is small and far removed from the plate boundaries, the stress intensity factors may be approximated as if the domain were the entire plane, giving

$$K_{\mathrm{I}} = \sigma\sqrt{\pi a}\cos^2\beta, \tag{1.9a}$$

$$K_{\mathrm{II}} = \sigma\sqrt{\pi a}\sin\beta\cos\beta. \tag{1.9b}$$

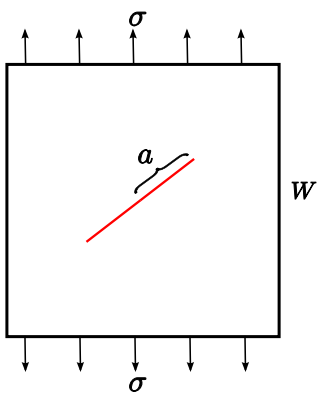


Figure 1.14: The setup for Example 3.

We compute the stress intensity factors as β ranges from 0 to $\pi/2$ in increments of $\pi/20$. Our simulation mesh has resolution 64×64 and we use a variety of refinement levels for the quadrature mesh. Figure 1.15 shows the relative error in the numerically computed stress intensity factors compared to the approximations (1.9). Our results are comparable to [MDB99].

1.7.4 Propagation Examples

Figures 1.16, 1.17, 1.19, and 1.21 show the results of applying our method to simulate the propagation of cracks (see §1.6). Each of these examples uses a Young's modulus of $E := 10^5$, Poisson's ratio of $\nu := 0.30$, and fracture toughness of 1. For each example, we compute a release rate via the stress intensity factors and propagate the crack by a fixed increment of 0.03 if the release rate exceeds the toughness [And05].

In Figure 1.16, we simulate a rectangular domain initialized with a straight crack with symmetric displacement Dirichlet boundary conditions applied to the left and right sides of the domain. The result is a crack propagating in a straight line that eventually divides the domain into disconnected halves. The colors used in Figure 1.16 represent the Frobenius norm of the stress, with the maximum norm in red and the minimum norm in blue (we use this color convention in the remaining figures as well).

Figure 1.17 shows the initial setup for the quasistatic propagation of a crack in a beam. As in [BB99], we use the beam dimensions $L := 11.82$ and $W := 3.94$, and we vary the initial perturbation angle θ among $1.43°$, $2.86°$, and $5.71°$. Figure 1.18 shows the results of the simulation for each of these three initial angles, where we have plotted the position of the crack tip at each time step: blue triangles correspond to $\theta = 1.43°$; orange squares correspond to $\theta = 2.86°$; and yellow triangles correspond to $\theta = 5.71°$. Our simulation mesh has resolution 64×20 and our quadrature mesh has 3 refinement levels. Our results are in good agreement with [BB99].

Figure 1.19 depicts a more complicated scenario. The initial configuration (Figure 1.19 (top left)) is a square (with side lengths 1) with two congruent holes (with radii 1/64). We initialize the simulation with two cracks, one emerging from each hole and both at $45°$ with respect to the horizontal such that the resulting geometry is rotationally symmetric (Figure 1.19(top right)). We subject this domain to constant traction boundary conditions on the left and right sides of the domain. Our simulation mesh has resolution 64×64 and our quadrature mesh has 2 levels of refinement. The bottom row of Figure 1.19 shows the simulation after 20 time steps (for clarity, we removed the disconnected material region between the two cracks).

For the example in Figure 1.19 and the next example, we use the mesh cutting algorithm from [SDF07] to construct an embedding mesh for the domain. We start with a triangulation of a square domain and use the mesh cutting algorithm to excise from it the two circular holes, each represented as a segmentd curve. We thus effectively treat the embedded boundary of the domain around the circular holes as crack surface that has fully disconnected the discs from the square. This illustrates a general technique to mesh a domain which has the

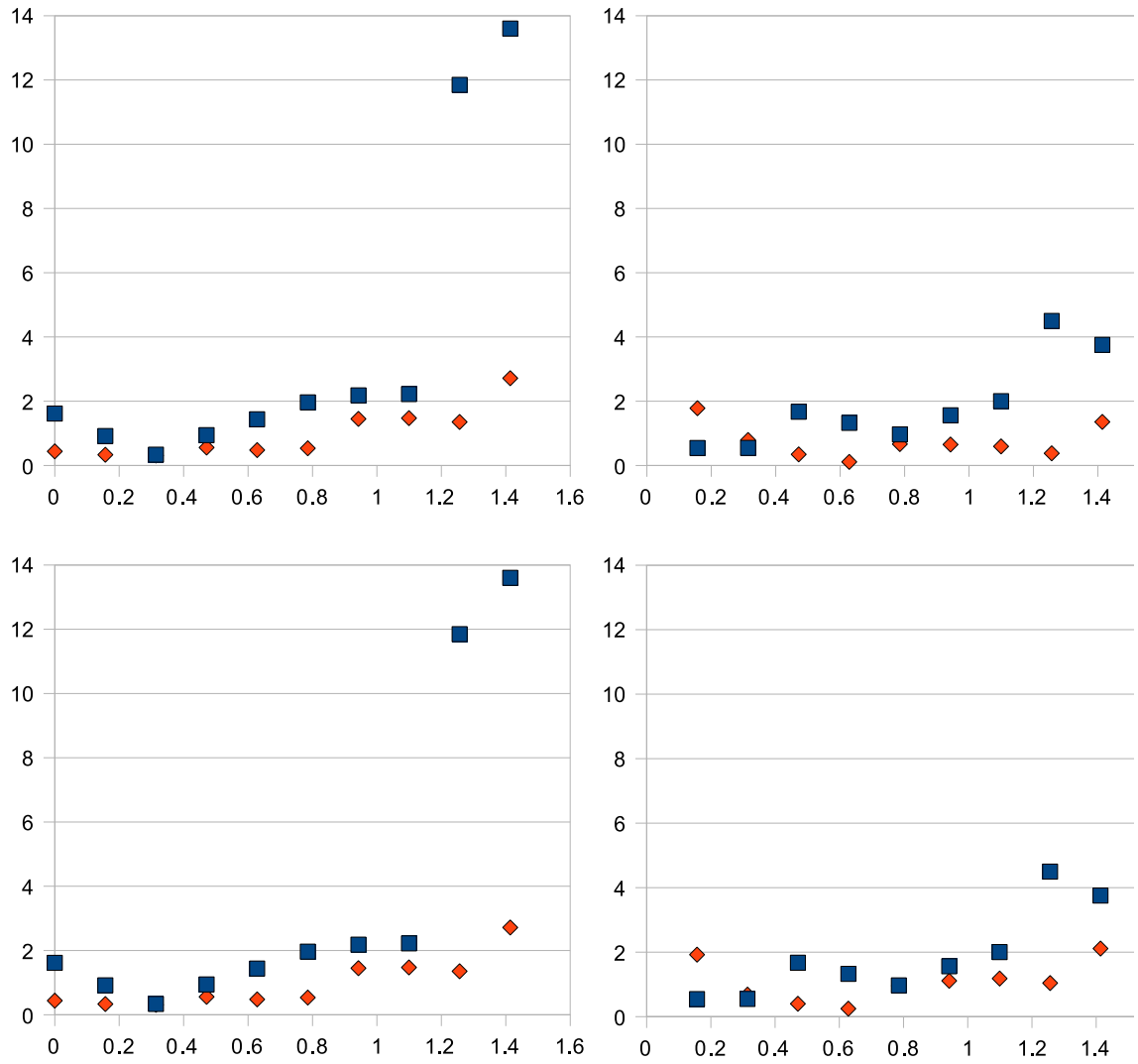


Figure 1.15: Results for Example 3. The top row compares the relative errors in K_{I} (left) and K_{II} (right) for 0 levels of quadrature mesh refinement (blue squares) and 5 levels of quadrature mesh refinement (orange diamonds). The bottom row compares 0 levels of refinement to just 1 level of refinement. Note that much of the accuracy improvement achieved with a very fine quadrature mesh (5 levels of refinement) is already present at only 1 level of refinement of the quadrature mesh.

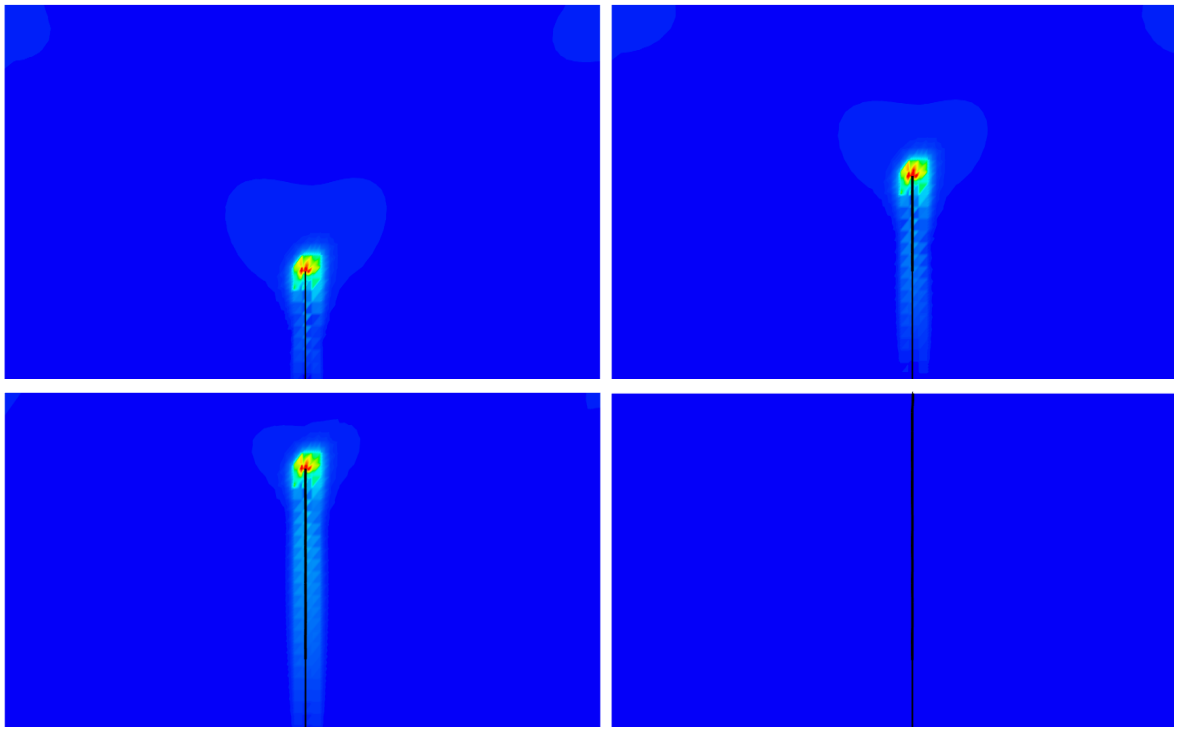


Figure 1.16: Simulation of a rectangular domain with symmetric boundary displacements; initial configuration (upper left), at 5 time steps (upper right), at 10 time steps (lower left), and at 15 time steps (lower right).

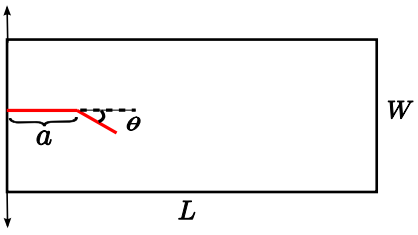


Figure 1.17: The setup for a crack propagation example involving a beam with a crack at various initial perturbation angles θ.

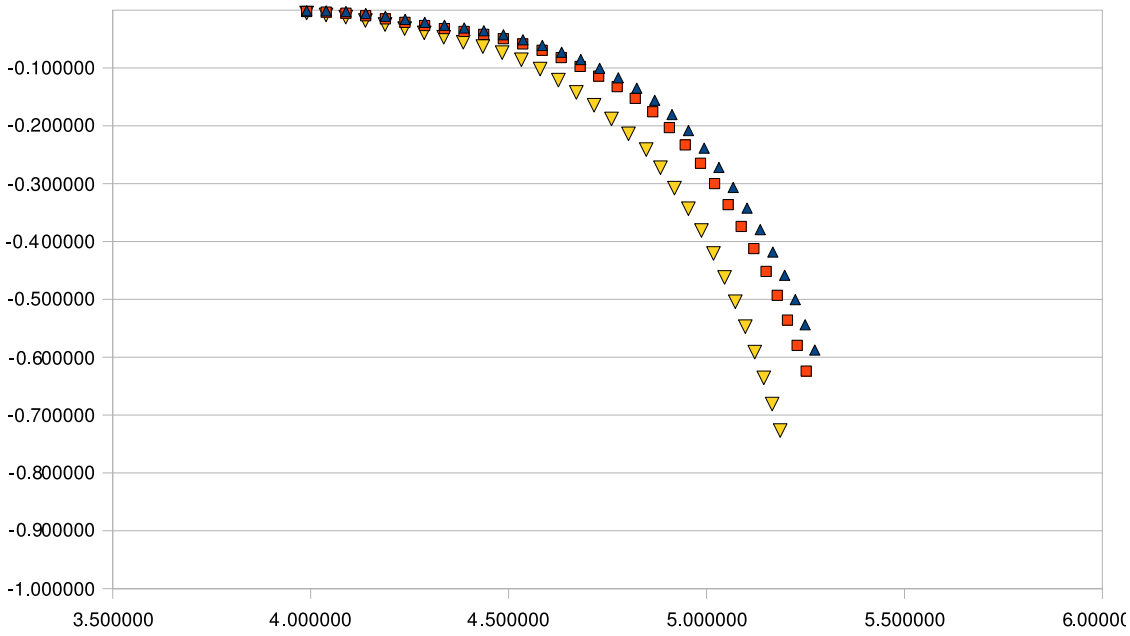


Figure 1.18: The results of crack propagation in the cantilever beam example. We plot the position of the crack tip at each time step: blue triangles correspond to $\theta = 2.86°$; and yellow triangles correspond to $\theta = 5.71°$.

advantage that one may control the simulation mesh resolution independent of the geometry of the domain boundary. Further, since the mesh cutting algorithm naturally handles a crack tip inside a fully cut triangle, we can automatically handle the crack tip reaching the (embedded) boundary of the domain.

Figure 1.21 shows an example with more complicated domain and crack geometry. As in the previous example, we mesh the domain by excising a segmented curve from a triangulated rectangle (with $L = 2$ and $W = 1$) with resolution 128×64 (see Figure 1.20). We then introduce initial triple-junction cracks into the domain and apply displacement Dirichlet conditions on the left and right. Specifically, we assign a fixed displacement of 0.001 to subset of the domain to the left of $x = -0.8$ or to the right of $x = +0.8$; these boundary conditions are visually evident in Figure 1.21, where the domain subsets on which we apply the fixed displacement have identically zero stress. We additionally apply traction boundary conditions at five other points along the boundary; these boundary conditions are also visually evident in Figure 1.21 by the high stress they induce. Our quadrature mesh has 2 levels of refinement. We simulate the propagation of the cracks over 20 time steps. Note that as the cracks evolve they may join with other cracks, which we accomplish by procedurally merging cracks whose paths intersect. However, crack tips cannot branch (creating new junctions) with the propagation methd described in §1.6.

1.8 Discussion and Conclusion

We presented an XFEM-based method for simulating crack propagation. This method employs the mesh cutting algorithm of Sifakis et al. [SDF07] to automatically generate the extra degrees of freedom traditionally associated with XFEM Heaviside enrichment; these degrees of freedom enable the two sides of the crack surface to separate. We additionally described an integration scheme based on a subordinate and independently refined quadrature mesh to accurately evaluate integrals involving the nonlinear (and sometimes singular) basis functions. This in turn yields accurately computed stress intensity factors, which we use to propagate the crack. The generality of the mesh cutting algorithm allows us to accommodate complex crack and domain geometry, as shown in the examples. We also illustrated the accuracy of our method, and it compares favorably with the results from the literature.

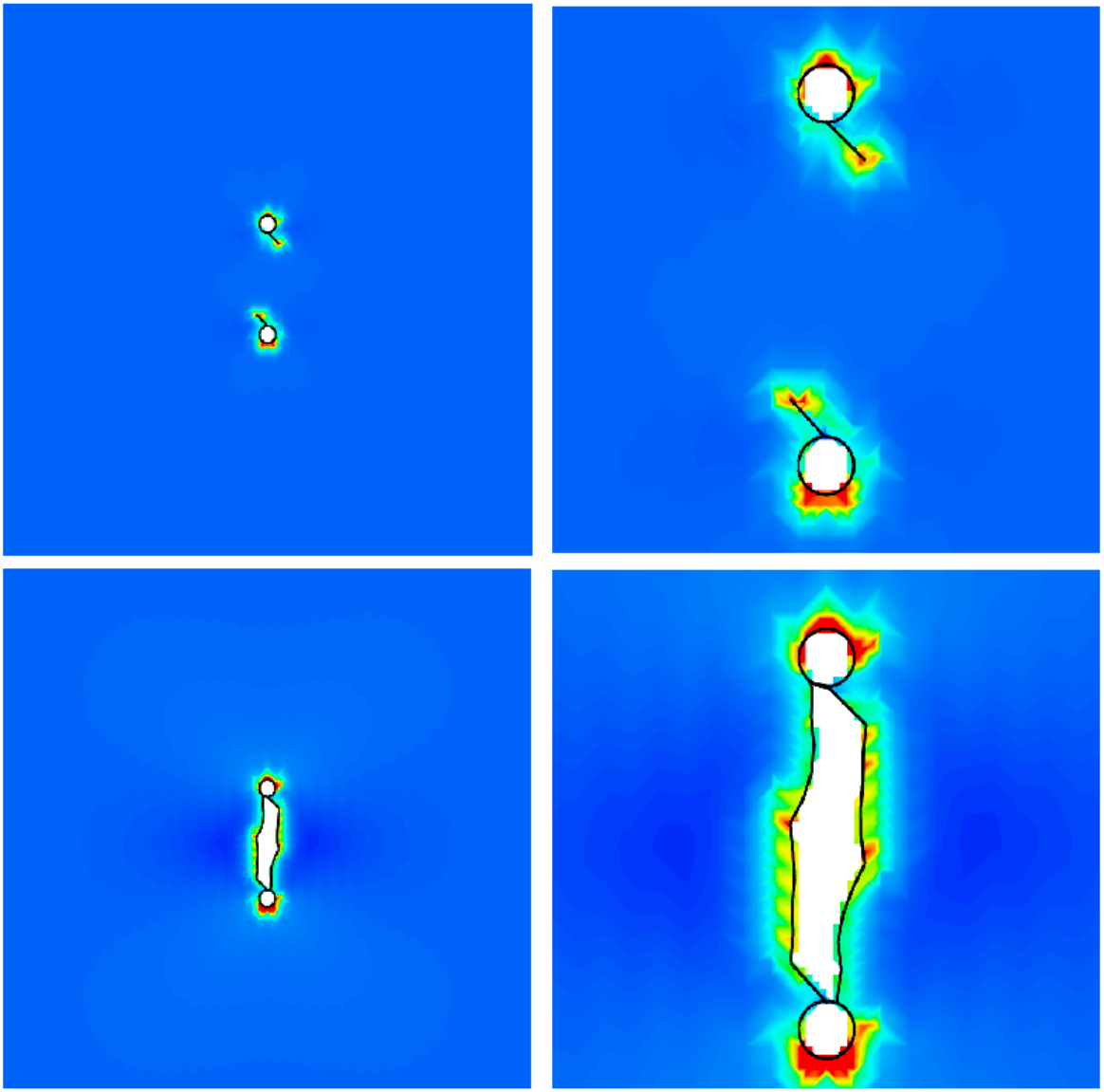


Figure 1.19: Crack propagation in a square domain with holes; initial configuration (top row) and simulation after 20 time steps (bottom row).

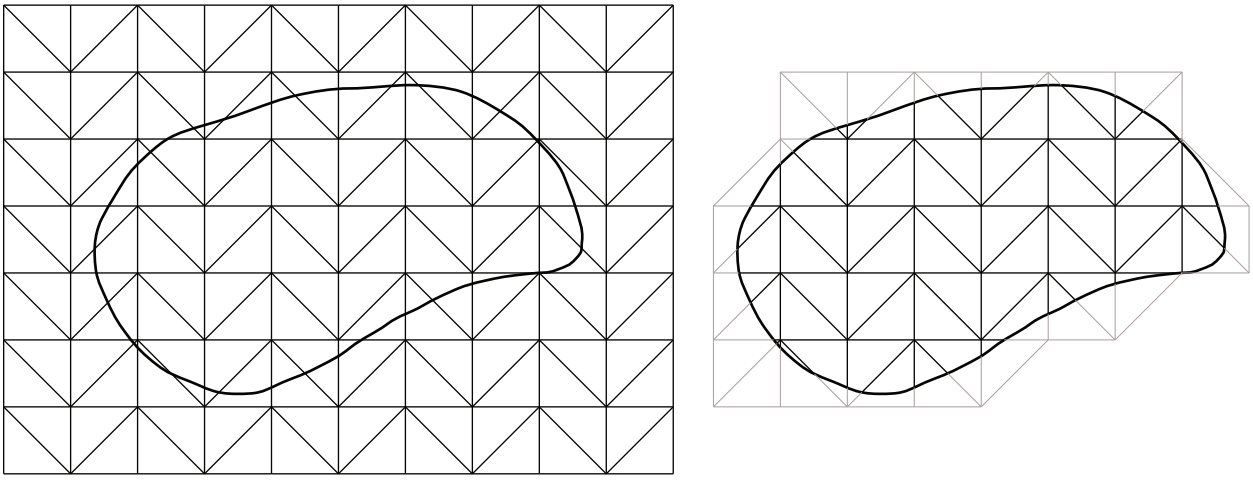


Figure 1.20: An example application of the mesh cutting algorithm to mesh a domain. A triangulated rectangle overlays the domain (left), and the mesh cutting algorithm resolves the domain boundary against the background mesh. This yields two disconnected meshes, an interior one (right) and an exterior one, which we discard.

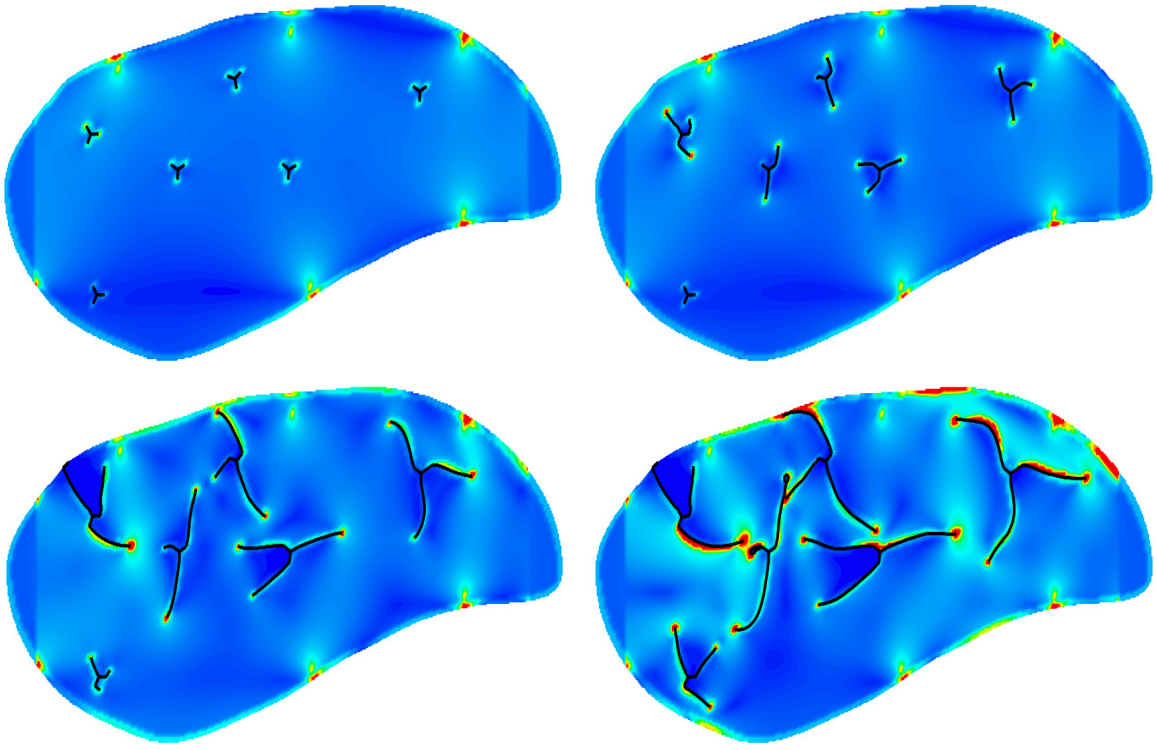


Figure 1.21: Crack propagation simulation with complex geometry; initial configuration (top left), at 5 time steps (top right), at 10 time steps (bottom left), and at 25 time steps (bottom right).

CHAPTER 2

Virtual Surgery

[1] The core principle of plastic surgery practice is the alteration of the geometry and topology of the skin. For a patient diagnosed with malignant melanoma, the plastic surgeon in many cases has to resect the tumor and the surrounding area. The extent of the skin that needs to be removed depends on the size and shape of the tumor. Furthermore, the removal of a large amount of tissue may have a dramatic affect on the patient's recovery and post-operative quality of life. The excision of tissue and subsequent defect closure constitute a "puzzle" the surgeon has to solve. In the past it has been typical for a surgeon to require many years to master the craft of skin flap design. The only documentation on how to solve this difficult three-dimensional problem would typically be limited to collections of two-dimensional illustrations. Currently a plastic surgeon can only practice this skill on a live patient in an operating room. Doing this as a laptop simulation [PLR95] has long been a dream; yet only recently have computer hardware advances promised sufficient computational capacity to accommodate the accuracy and real-time performance requirements of a tool usable in actual clinical practice.

The methodology illustrated in this paper is focused on the computer-aided simulation of *open surgery*. In contrast, closed surgery, exemplified by endoscopic procedures (e.g. laparoscopy) typically shifts the computational burden to the three-dimensional visualization and navigation of the endoscopic tools in the internal anatomy, while simulation of deformable tissues typically excludes topological manipulation, or where any topological change is limited and localized (e.g. local excision or cauterizaion). In contrast, open surgery is predominantly centered around the alteration of both the *geometry* and *topology* of the surgical subject. A simplistic example can be seen in a tracheostomy where an incision is performed in the neck, cutting through skin, cartilage and muscle down to the wind pipe, and the resulting flap is stitched to the surface. More sophisticated open surgery, such as a cleft lip and palate repair, may involve a highly elaborate sequence of incisions, tissue transposition and suturing. The *essence* of open surgery is thus the topological change involved.

Despite the elaborate nature of certain open surgery procedures, the large majority of such repairs can be constructed from a small number of building blocks or fundamental operations. In our virtual surgical simulation environment we formalize the application of these operations via three basic tools:

- The Incision Tool is used to specify the surface swept by the virtual scalpel, and define the topological change intended by the surgeon.

[1] The content of this chapter is a version of [SHT09] with minor revisions.

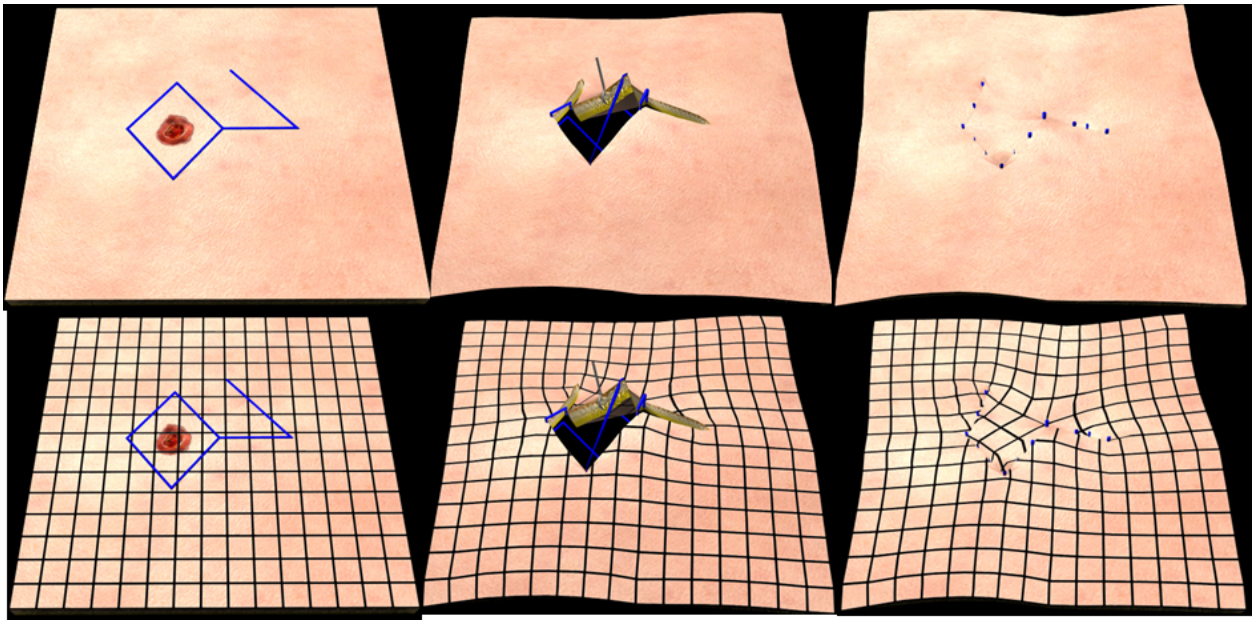


Figure 2.1: Simulation of a malignant melanoma removal and closure of the resulting defect with a rhomboid flap procedure. The gridded texture demonstrates the post-procedure topology and geometry of the tissue.

- The Retraction Tool is used to grasp and manipulate the skin after or in between performing incisions.
- The Suture Tool is used to stitch parts of the geometry together, once they have been placed adjacent to one another.

The successive application of these fundamental tools is recorded in a replayable script file, such that the sequence of actions may be repeated even if the underlying skin geometry or the discrete simulation mesh are modified. This allows an operation strategy to be efficiently sketched out at interactive simulation rates, and then replayed with a much higher simulation resolution and more accurate material property models to obtain more accurate measurements of tissue deformation and stress.

2.1 Technical Background

A usable, practical and beneficial open surgery simulator needs to meet certain important requirements in order to address the needs of a clinical setting. First, the computational performance must enable real-time interaction when authoring a certain surgical strategy. The material models used must be accurate and representative of the (complex, typically highly nonlinear) constitutive properties of the biological tissues involved. Furthermore, reconstructive surgery typically entails substantial tissue deformation, requiring the numerical and algorithmic robustness of any simulation techniques used. Finally, all these requirements

need to be reconciled with the need for a high level of visual detail, both in terms of texture and geometrical detail, to reflect the geometric and visual complexity of the subject tissues and aid in the reproduction of the process in the operating room by providing discernible surface landmarks for the various surgical operations.

2.1.1 Real-Time Simulation

A virtual simulation environment will have a vastly reduced potential for being used in actual practice if it does not offer the ability for a clinician to cut and manipulate the skin in real-time. In a finite-element discretization of a volumetric object representing a tissue flap, certain algorithmic and numerical factors may severely compromise the real-time performance of such a system. First, the resolution of the simulation mesh alone needs to be limited enough to allow for real-time simulation; although discretizations with several hundreds of thousands or millions of tetrahedral elements would be desired (and may actually be feasible in the light of emerging massively parallel computing platforms), commodity computer hardware dictates stricter limits for simulations tractable with the computational resources of a mainstream laptop, for example. Instead of compromising the visual detail contained in our model for the sake of a coarser discretization, we employ an embedded scheme (as in [SDF07, SSI07]) to allow a coarser simulation mesh to serve as a "framework" for a higher resolution geometry. The surface resolution may be substantially higher than that of the embedding grid, and certain aspects of simulation (such as collision processing) may be handled on the high-resolution embedded geometry if so desired. Additionally, this eliminates the risk of ill-conditioned simulation elements necessitated to resolve intricate geometrical features of the tissue surface. More important, this practice circumvents the need for remeshing of the tissue geometry in order to resolve the topological change incurred by incisions. At all times, the simulation grid is maintained at a regular, lower degree-of-freedom lattice (with a topology adhering as closely as possible to the topology of the continuous tissue volume).

2.1.2 Accuracy and Nonlinear Deformation

Although linear material models and mass-spring networks have been widely used for interactive simulation of deformable solids, providing a virtual surgery simulation system with the ability to make reliable predictions about the behavior of real tissue necessitates the adoption of much more accurate, nonlinear, anisotropic constitutive material models. This requirement will be essential in making the virtual surgery framework presented here capable of reliable predictions of the surgical outcome, establishing a virtual system as a dependable platform for surgical planning. The constitutive models that accurately convey the material properties of human flesh have to account for the inhomogeneity of materials (e.g. anatomical parts consisting of passive fatty tissue, active musculature, tendons, ligaments and connective tissue). Furthermore, the materials involved are nonlinear and near-incompressible, in sharp contrast with linear material approximations which may be employed in applications where deformation is limited to the small strain regime, or where physical accuracy is not

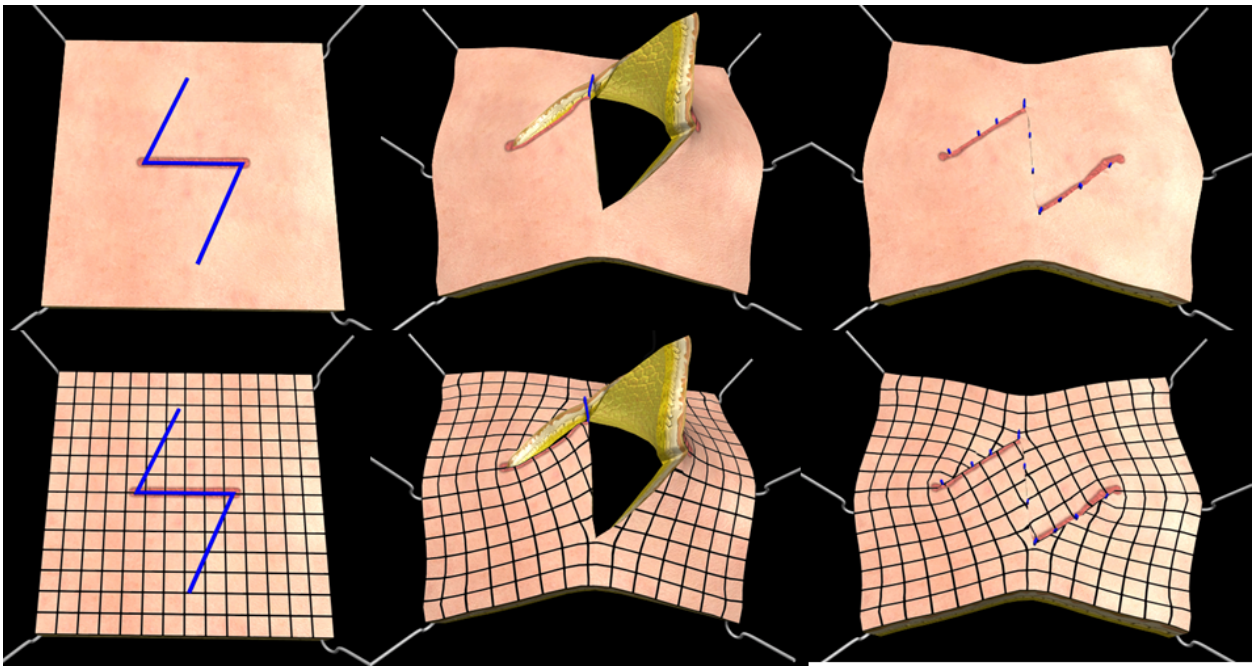


Figure 2.2: Simulation of a z-plasty procedure for the elongation of a scar contracture.

essential. Our system supports arbitrary nonlinear, inhomogeneous and anisotropic material properties, which may be defined on the basis of every distinct simulation element in the underlying embedding mesh. This also highlights the ability of the system to facilitate a two-pass simulation process, where a lower resolution embedding mesh (where the nonlinearity and inhomogeneity may manifest themselves in a limited capacity) can be used for crafting the surgical approach interactively, and a subsequent pass where the same sequence of actions is repeated offline on a highly refined embedding mesh, which is able to resolve the intricacies of the nonlinear deformation.

2.1.3 Robustness Under Large Deformation

The manipulation of flesh during plastic surgery operations is by no means limited to small geometric change; in fact, large strain deformation is quite typical of the configurations involved in the closure of the tissue flaps created. Therefore, especially given the necessity for nonlinear constitutive models and the relatively under-resolved nature of the simulation (owing to the embedding approach), the simulation methods employed have to address and survive extreme geometric configurations, such as element inversion or collapse. We employ the Invertible Finite Element method of [ITF04] which allows such simulations to continue and gracefully recover when transitioning through such extreme geometric configurations, while still supporting the full gamut of nonlinear constitutive models.

2.2 Tools and Methods

We have created a "local flaps" simulator that will allow surgeons to practice their closing designs in a three-dimensional environment with real-time interaction. This environment uses the PhysBAM physics simulation library to allow the user to make incisions, move tissue flaps, and create virtual sutures to simulate closing of a skin defect, all in a scientifically accurate framework. The simulator consists of several simple surgical tools:

- The Incision Tool. This tool creates a triangulated surface which represents the area swept by the scalpel during an incision. This incision surface is generated by sketching a curve (either a sequence of straight line cuts, or a spline curve) on the surface of the tissue, while controlling the angle which the scalpel forms with the tissue being cut. The result of this operation is a discrete triangulated surface representation of the incision being made. The algorithm of [SDF07] is subsequently used to determine the topology resulting from this manipulation of the tissue geometry, generating the necessary degrees of freedom to enable the opening of the tissue at the location of the incision.

- The Retraction Tool. Once the topology of the incision has been resolved, the action of grasping and manipulating the tissue flap is performed using a simulated hook-and-handle mechanism. By direct selection, a point on the surface of the tissue is defined as the anchor point of a retraction site (i.e. a hook point). At the same time, a "handle" point is defined by offsetting the hook location a certain short distance off the surface of the tissue. This handle can be arbitrarily positioned in 3D space, giving rise to deformation of the simulated tissue. The target position of the handle is communicated to the deformable tissue via a spring force that aims to bring the hook at the specified 3D location.

- The Suture Tool. Once the retraction tool has been used to deform the tissue shape into its target location, this tool emulates the process of suturing two adjacent surfaces together. The suture can be either a point-to-point connection, or a curved path connecting two sides of tissue. The geometry on either side of the suture are brought together by the simulated action of a zero rest-length spring joining the parts of the tissue connected by the suture.

The successive application of these fundamental tools is recorded in a replayable script file, such that the sequence of actions may be repeated even if the underlying skin geometry or discrete simulation mesh are modified; this allows an operation strategy to be efficiently sketched out in interactive simulation rates, and then replayed with a much higher resolution simulation mesh to obtain more accurate measurements of tissue deformation and stress. Using this simple combination of surgical tools, the surgeon is able to practice existing procedures for closing the defect. A surgeon may also invent a new pattern altogether and assess its efficacy based on such physically quantifiable metrics as post procedure tension in the tissue and suture. The elasticity of the tissue is simulated using the finite element

method defined on a volumetric tetrahedral representation of the tissue. An implicit time stepping scheme is used to obtain a frame rate adequate for interactivity. Tissues involved typically undergo large deformation and the algorithms of [ITF04] and [TSI05] are used to guarantee robust performance in this challenging setting. Tissue incisions are represented using the novel tetrahedral cutting approach of [SDF07] and sutures are modeled with the highly flexible embedding framework of [SSI07].

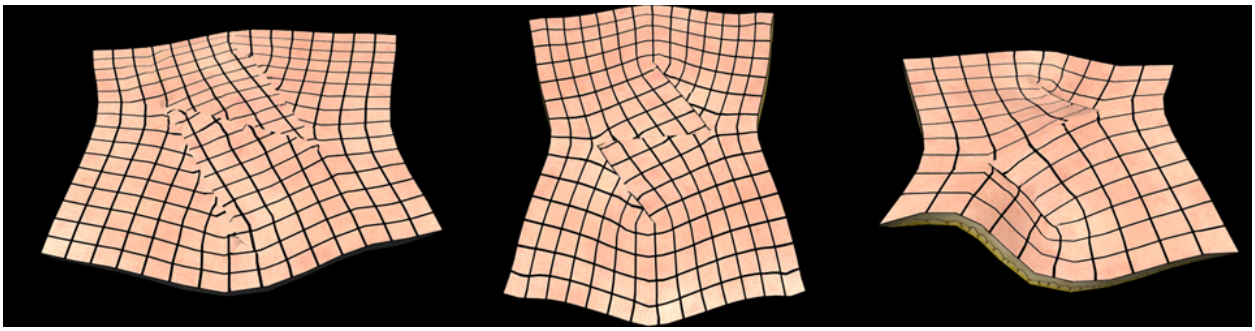


Figure 2.3: Comparison of the simulated results of Z-plasty procedures with incision angles at 45, 60 and 90 degrees respectively from left to write. Simulation confirms the conventional wisdom that 60 degrees is the optimal incision angle. Also, the 90 degree incision reproduces the so-called "dog ear" effect.

2.3 Results

Figure 2.1 depicts the results of a local flaps simulation of a rhomboid flap procedure for removing a malignant melanoma and repairing the skin near the excision region. This is a very common procedure and the simulated tissue configuration matches the conventional wisdom very closely. Different stages in a z-plasty procedure (typically used for elongating scar contractures) shown in Figures 2.2 and 2.3, show the effects of varying the angles of the z-incision. In practice, the optimal angle of incision is 60 degrees. Our simulated results also suggest that 60 degrees is the optimal angle as lower angles fail to produce sufficient elongation and large angles produce non-planar equilibrium configurations (or the so-called "dog ear" effect).

2.4 Conclusions

The local flaps environment is an effective tool that the plastic surgeon can utilize to leverage physically accurate simulation in an interactive environment to improve many aspects of his cognitive surgical practice. The uses of the environment fit into two basic categories. The first is related to training in existing procedures. These scenarios demand realtime interaction but not necessarily the highest degree of physical accuracy. The second is related to design of new procedures. Such applications require a higher degree of physical accuracy. Our finite

element based approach allows for this and has the ability for future incorporation of subject specific constitutive models that can be used to study repair of unique injuries (e.g. those arising on the battlefield).

CHAPTER 3

Geometric Fracture Modeling in Computer Animation

Figure 3.1: Left, middle: Rhino's ball is riddled with cracks as a metal gate crushes it down. Right: A roadway is torn up by Bolt's "superbark".

3.1 Introduction

[1] Modeling the geometry of solid materials cracking and shattering into elaborately shaped pieces is a painstaking task, which is often impractical to tune by hand when a large number of fragments are produced. In Walt Disney's animated feature film *Bolt*, cracking and shattering objects were prominent visual elements in a number of action sequences. We designed a system to facilitate the modeling of cracked and shattered objects, enabling the automatic generation of a large number of fragments while retaining the flexibility to artistically control the density and complexity of the crack formation, or even manually controlling the shape of the resulting pieces where necessary. Our method resolves every fragment *exactly* into a separate triangulated surface mesh, producing pieces that line up perfectly even upon close inspection, and allows straightforward transfer of texture and look properties from the un-fractured model.

3.2 Crack Geometry Generation

The input to our system consists of a closed triangulated surface defining the (uncut) solid object to be fractured and one or more additional triangulated surfaces defining the geometry of the cracks. The geometry of this "crack surface" is not constrained by the shape of the material object itself; cracks are free to extend outside the material into the empty space, and can have non-manifold shapes, topological junctions, or even intersect themselves. Leveraging this flexibility, we can automatically create a fracture surface which partitions

[1] The content of this chapter is a version of [HSS09] with minor revisions.

the ambient space into any given number of regions by simply introducing the same number of seed points and computing the 3D Voronoi regions of those seed locations. The fracture surface is then defined as the union of all boundaries between Voronoi cells (see Figure 3.2, top right). In practice, we approximate these regions by laying down a background, procedurally generated tetrahedral mesh, and computing the Voronoi cells via a flood-fill on the tetrahedral mesh, starting from the elements containing the seed points.

The shape of the fracture surface can be controlled by specifying the number of seed points, or even by volumetric painting of seed point densities, to control which regions will shatter into more, smaller fragments. We also control the smoothness of the boundaries between Voronoi regions by jittering the background tetrahedral mesh prior to the flood-fill, and selectively smoothing the boundary surface where smoother cracks are desired. Finally, in certain situations (e.g. the cracks of the hamster ball in Figure 3.1), more specific artist control of the fragment geometry is desired. For these cases, we extruded sets of artist-drawn 3D curves in the direction normal to the object surface to create a fracture surface that cuts through the material and reflects the crack design intended by the artists.

3.3 Automatic Fragment Mesh Generation

We adapt the cutting algorithm of [SDF07] to automatically generate triangulated meshes for the material fragments defined by the object and fracture geometries of the previous section. In particular, their method begins with a tetrahedral volumetric representation of the material to be fractured, as opposed to the triangulated boundary geometry we assumed as input to our system. We handle this representation discrepancy by first generating a tetrahedral mesh that fully covers the object to be fractured . The triangulated surface of the uncut object itself is used as the first cut in an application of the algorithm of [SDF07], effectively sectioning the background tetrahedral volume into the "material" and "void" regions. The fracture surface is then applied as the second cut, resulting in the separation of the material volume into separate fragments. The cutting algorithm computes the triangulated boundary of every volumetric fragment in a way that every triangle of a fragment is contained inside a triangle either of the uncut object, or of the fracture surface (Figure 3.2, bottom). As a result, texture and look properties can be remapped simply by embedding each resulting triangle barycentrically into either the material or the fracture surface respectively, and looking up the properties of the embedding triangle. Finally, although our framework is currently used for geometric modeling, we aim to employ it in conjunction with simulation in the future, to model time-dependent crack propagation.

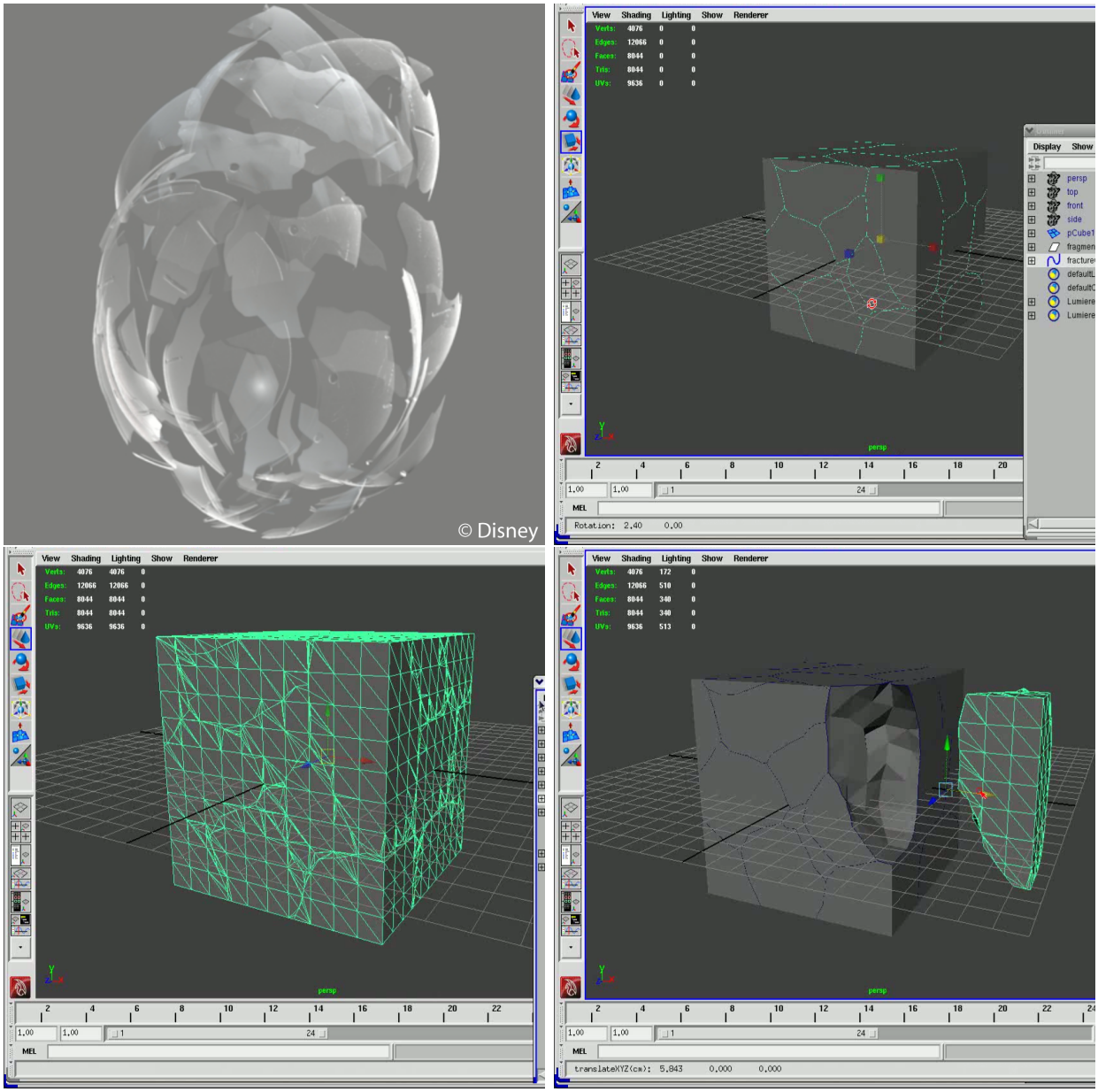


Figure 3.2: Top left: Simulation of the shattered fragments of Rhino's ball. Top right: Fracture surfaces defined as the boundaries of Voronoi regions in 3D. Bottom: The fragments are fully resolved as independent surface meshes, and can be separately manipulated.

Part II

Virtual Node Methods for Elliptic Problems

Introduction

We now turn toward the solution of elliptic partial differential equations on arbitrary, irregular domains with a discretization based on a regular Cartesian grid. We will specifically consider the following problems in Chapters 4 and 5, respectively.

- Poisson's equation with interfacial jump conditions:

$$-\nabla \cdot (\beta \nabla u) = f, \quad \in \Omega \setminus \Gamma; \tag{II.1a}$$
$$[u] = a, \quad \in \Gamma; \tag{II.1b}$$
$$[\beta \nabla u \cdot \hat{\mathbf{n}}] = b, \quad \in \Gamma; \tag{II.1c}$$
$$u = p, \quad \in \partial\Omega_d; \tag{II.1d}$$
$$\beta \nabla u \cdot \hat{\mathbf{n}} = q, \quad \in \partial\Omega_n; \tag{II.1e}$$

  where

  - the domain $\Omega \subset \mathbb{R}^d$ is open ($d = 2$ or 3, typically);
  - the interface Γ is a co-dimension one closed curve ($d = 2$) or surface ($d = 3$) that divides Ω into an interior domain Ω^- and an exterior domain Ω^+, such that $\Omega = \Omega^- \sqcup \Omega^+ \sqcup \Gamma$;
  - u (unknown), β (known), and f (known) are scalar functions which can exhibit discontinuities across Γ but otherwise are assumed to have smooth restrictions u^σ, β^σ, f^σ to Ω^σ, $\sigma \in \{+, -\}$;
  - $\hat{\mathbf{n}} \equiv \hat{\mathbf{n}}(\mathbf{x})$ denotes the outward unit normal, both to Ω^- for $\mathbf{x} \in \Gamma$ and to Ω for $\mathbf{x} \in \partial\Omega$; and
  - $[v](\mathbf{x}) := v^+(\mathbf{x}) - v^-(\mathbf{x}) := \lim_{\epsilon \to 0^+} v(\mathbf{x} + \epsilon \hat{\mathbf{n}}(\mathbf{x})) - \lim_{\epsilon \to 0^+} v(\mathbf{x} - \epsilon \hat{\mathbf{n}}(\mathbf{x}))$ denotes the *jump* of the quantity v across the interface Γ.

  We note that the relevant physics generally determine the jumps in the solution (II.1b) and in the flux (II.1c), as well as the boundary conditions (II.1d), (II.1e) on $\partial\Omega$.

- The equilibrium equations of linear elasticity:

$$-\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}) = \mathbf{f}, \quad \in \Omega; \tag{II.2a}$$
$$\mathbf{u} = \mathbf{u}_0, \quad \in \partial\Omega_d; \tag{II.2b}$$
$$\boldsymbol{\sigma}(\mathbf{u}) \cdot \hat{\mathbf{n}} = \mathbf{g}, \quad \in \partial\Omega_n; \tag{II.2c}$$

  where

  - the domain $\Omega \subset \mathbb{R}^d$ is open (we consider $d = 2$ primarily);
  - $\mathbf{u}$ is the (unknown) material displacement map;
  - $\boldsymbol{\sigma}$ is the Cauchy stress tensor;

– $\mathbf{f}$ is the external force per unit volume;
– $\mathbf{u}_0$ is the prescribed Dirichlet boundary displacements over $\partial\Omega_d \subset \partial\Omega$; and
– $\mathbf{g}$ is the prescribed external surface traction over $\partial\Omega_n \subset \partial\Omega$.

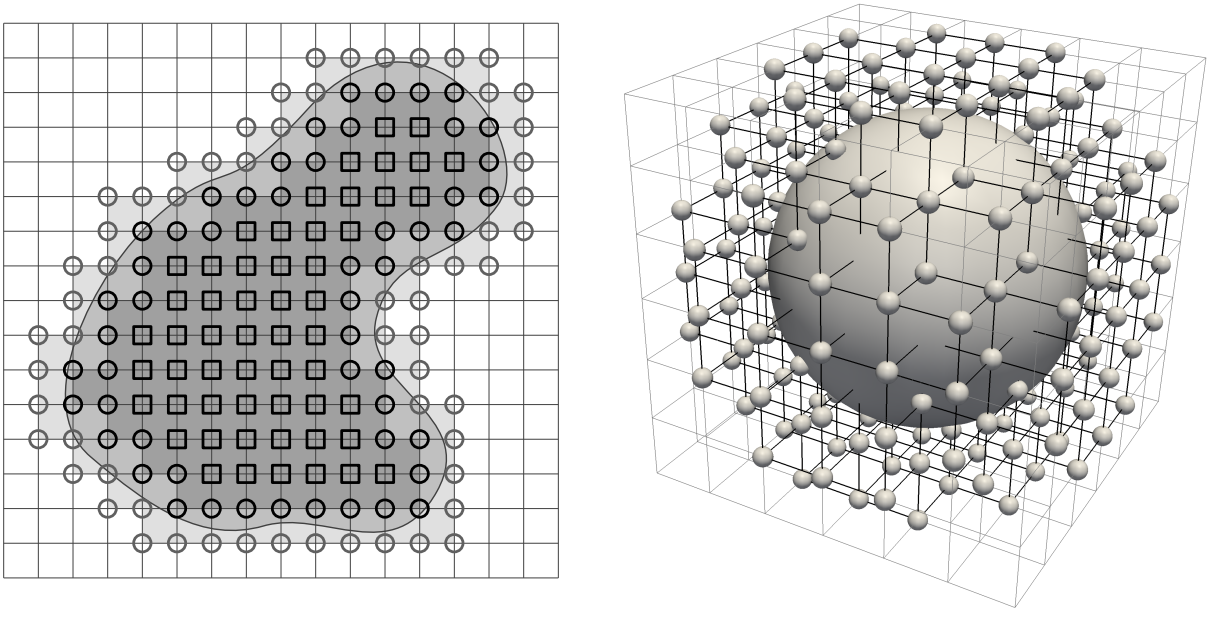
In linear elasticity, the stress $\boldsymbol{\sigma}(\mathbf{u})$ is linearly dependent on the Cauchy strain $\boldsymbol{\epsilon}(\mathbf{u})$ via

$$\begin{aligned}\boldsymbol{\epsilon}(\mathbf{u}) &:= \frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^t\right), \\ \boldsymbol{\sigma}(\mathbf{u}) &:= 2\mu\boldsymbol{\epsilon}(\mathbf{u}) + \lambda\left(\operatorname{tr}\boldsymbol{\epsilon}(\mathbf{u})\right)\mathbf{I} \\ &= \mu\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^t\right) + \lambda\left(\nabla\cdot\mathbf{u}\right)\mathbf{I}.\end{aligned}$$

Therefore, (II.2) is equivalently formulated as

$$-\left(\mu\Delta\mathbf{I} + (\lambda+\mu)\nabla\nabla^t\right)\mathbf{u} = \mathbf{f} \quad \in \Omega \tag{II.3a}$$
$$\mathbf{u} = \mathbf{u}_0 \quad \in \partial\Omega_d \tag{II.3b}$$
$$\mu\left(\mathbf{u}\cdot\hat{\mathbf{n}} + \nabla\left(\mathbf{u}\cdot\hat{\mathbf{n}}\right)\right) + \lambda\left(\nabla\cdot\mathbf{u}\right)\hat{\mathbf{n}} = \mathbf{g} \quad \in \partial\Omega_n. \tag{II.3c}$$

In all the above, $\partial\Omega = \partial\Omega_d \sqcup \partial\Omega_n$, and we assume Γ (if applicable) and $\partial\Omega$ to be sufficiently smooth. See Figure II.3 for an example of an embedding of a domain Ω within a background Cartesian grid in 2 and 3 dimensions.



(a) Embedding in 2 dimensions

(b) Embedding in 3 dimensions

Figure II.3: Example domain embeddings in (a) 2 dimensions and (b) 3 dimensions. In a typical domain embedding, only grid vertices on grid cells which intersect the domain (in (a), shaded) are considered degrees of freedom.

Although the use of embedded methods avoids the complexities inherent to unstructured mesh generation, as discussed in the overall Introduction, they do introduce difficulties of their own, though typically of a different and less computationally intensive variety. One prominent such difficulty is the enforcement of algebraic surface conditions (such as boundary conditions or interfacial jump conditions) on embedded features such as $\partial\Omega$ and Γ, since the degrees of freedom now do not lie on said embedded features. Our methods employ a Lagrange multiplier space in a relatively straightforward way to weakly enforce Dirichlet boundary conditions and interfacial jump conditions. We mention some alternative techniques in the background sections in the subsequent chapters, and for a good review, see Lew et al. [LB08].

A further challenge is the retention of higher order accuracy in L^∞. The difficulty typically lies in determining the numerical stencils near embedded features that retain the accuracy achieved in the regions of the domain sufficiently distanced from embedded features. Many present methods address these problems at the cost of implementation complexity and sometimes require significant effort to adapt to general applications. But while achieving higher order accuracy in the discretization may be nontrivial, to reiterate from the overall Introduction, the ability to use a regular Cartesian grid greatly facilitates the implementation of efficient solution techniques. Specifically for high resolution discretizations, direct methods to solve the discrete linear systems become too slow and memory intensive. Geometric multigrid methods and domain decomposition approaches have been shown to provide very favorable performance in this setting. However, their application to embedded discretizations is not entirely straightforward. Ultimately, special attention must be paid near embedded features for both discretization accuracy and efficient numerical linear algebra, as we describe in the subsequent chapters.

The methods presented in the following chapters build on the work of Bedrossian et al. [BBZ10], which introduced a second order virtual node method for solving the interfacial Poisson problem (II.1) in 2 dimensions. The discretization presented in [BBZ10] is easy to implement and yields a symmetric positive definite sparse linear system for both interface problems and boundary value problems on irregular domains. In summary, this virtual node method employs a uniform Cartesian grid with duplicated Cartesian bilinear elements along the interface. These duplicated elements introduce additional *virtual* nodes or degrees of freedom to accurately capture the lack of regularity in the solution. The method is variational to define stencils symmetrically, and a different discretization is used depending on proximity to embedded features, allowing for the retention of the standard 5-point finite difference stencil away from embedded boundaries and interfaces. Langrange multipliers are used to weakly enforce embedded Dirichlet conditions (II.1d) and embedded interfacial jump conditions (II.1b), and the choice of Lagrange multiplier space admits a symmetric positive definite discretization. In the special case when β is smooth, a discontinuity removal technique allows the use of the standard 5-point Poisson stencil even across the embedded interface.

In Chapter 4, we improve many aspects of [BBZ10] and provide key modifications necessary to extend the method to 3 dimensions. Within the context of discretizing the Dirichlet conditions (II.1d) and interfacial jump conditions (II.1b), we present a novel and flexible al-

gorithm to define the discrete Lagrange multiplier space. This algorithm gives more control on the conditioning of the resulting linear system and specifically addresses the conditioning issues (see Appendix C) we found in the straightforward extension of [BBZ10] to 3 dimensions. We also give an expanded treatment of the discontinuity removal technique, detailing an algorithm for the construction of a scalar function satisfying the interfacial jump conditions (II.1b, II.1c).

In Chapter 5, we apply some of the same basic ideas from Chapter 4 to numerically solve the equilibrium equations of linear elasticity in the nearly incompressible regime. We stabilize the method by introducing pressure as an additional unknown in a mixed variational formulation. Our discretization of this variational formulation is then based on a MAC-type staggering of x and y-component displacements with pressure degrees of freedom at cell centers. Again, we achieve second order accuracy in L^∞ while retaining a symmetric positive definite linear system.

Further, within both chapters, we present a family of geometric multigrid algorithms to solve the resulting linear systems with near-optimal multigrid efficiency independent of grid resolution. The variational nature of the methods naturally enables symmetric numerical stencils along embedded features, and together with our choice of Lagrange multiplier space, this admits an efficient means for smoothing boundary and interface equations. Indeed, this shows the suitability of efficient parallel implementations, as indicated, e.g., by the similarities to the first order method to solve (II.3) in [ZST10].

In summary, our methods for solving elliptic problems such as Poisson's equation (II.1) and the equilibrium equations of linear elasticity (II.3) have a distinguishing feature set rarely found simultaneously in the large class of embedded methods:

- second-order accuracy in L^∞;
- addresses both Neumann (II.1e) / (II.3c) and Dirichlet (II.1d) / (II.3b) boundary conditions as well as the interfacial jump conditions (II.1b, II.1c);
- relatively simple to implement with a nice progression of complexity from the discretization of Neumann boundary conditions to Dirichlet boundary conditions to interfacial jump conditions;
- yields a discrete linear system which is symmetric and positive definite, facilitating the application of a wide variety of black-box solvers or, as we show, efficient geometric multigrid methods with nearly optimal convergence rates.

Furthermore, our method for solving (II.3) retains the above properties even in the nearly incompressible regime.

CHAPTER 4

Poisson with Interfacial Jump Conditions

4.1 Background and Existing methods

[1] To review, this chapter addresses the solution of Poisson's equation with interfacial jump conditions, repeated here for convenience:

$$-\nabla \cdot (\beta \nabla u) = f, \quad \in \Omega \setminus \Gamma; \tag{4.1a}$$
$$[u] = a, \quad \in \Gamma; \tag{4.1b}$$
$$[\beta \nabla u \cdot \hat{\mathbf{n}}] = b, \quad \in \Gamma; \tag{4.1c}$$
$$u = p, \quad \in \partial\Omega_d; \tag{4.1d}$$
$$\beta \nabla u \cdot \hat{\mathbf{n}} = q, \quad \in \partial\Omega_n; \tag{4.1e}$$

where we wish to solve for the unknown scalar function u.

The *Immersed Interfaced Method* (IIM) is perhaps the most popular finite difference method for approximating (4.1) to second order accuracy. LeVeque and Li first proposed the IIM for approximating elliptic interface problems in [LL94] and the term now applies to a widely researched and extensively applied class of finite difference methods [LL97, LL01, LL03, LKP06, XW06, TLL08, XW08]. See [LI06] and the references therein for a complete exposition of the method and its numerous applications, and [BL06] for justification of the general IIM approach. Using generalized Taylor expansions, the original IIM adaptively modifies the stencil to obtain $\mathcal{O}(h)$ truncation error along the interface. For smooth β, this reduces to the standard 5-point or 7-point finite difference stencil, but otherwise results in an asymmetric discretization that follows from locally solving constrained optimization problems that enforce a discrete maximum principle [LI01]. The IIM also generally requires the evaluation of higher order jump conditions and surface derivatives along the interface. This can lead to difficulty in implementation, especially in 3 dimensions [DIL03, LI06, XW06, XW08]. Chen and Strain described a new approach to the IIM, called the *Piecewise-polynomial Interface Method* (PIM), in [CS08] that does not require the derivation of additional jump conditions and accurately treats complex interfaces. Various other attempts have been made [Li98a, WB00, Ber04, AL02, AC04, AC05, LI06] to improve the efficiency and reduce the complexity of the IIM.

Extrapolation-based finite difference schemes such as [LFK00, GFC02, ZZF06, GF05, JM05, CS07] introduce fictitious points along coordinate axes and use the known jump conditions to determine their values. The *Ghost Fluid Method* (GFM), such as that presented

[1] The content of this chapter is a version of [HWS12] with moderate revisions.

by Liu et al. in [LFK00], exemplifies such methods. For 2− and 3−dimensional interface problems, the GFM neglects the tangential flux terms $[\beta \nabla u \cdot \hat{\tau}]$ when determining fictitious values, yielding a symmetric positive definite system and a resulting method which is first order accurate [LFK00, LS03]. However, the GFM is capable of achieving up to fourth order accuracy for irregular domain problems [GFC02, GF05]. The GFM is similar to our approach in spirit. We also incorporate similar ideas from the *Virtual Node Algorithm* (VNA) [MBF05, BHT07, SDF07]. Various other approaches attain higher order accuracy by accounting for the tangential flux in other ways, often sacrificing simplicity and symmetry of discretization in the process. For instance, the *Coupling Interface Method* (CIM) [CS07] extends the GFM to higher order by using a second order extension at most grid points but reverting to a first order method at grid points where the second order extension cannot be applied. The method couples jump conditions in different directions to express the tangential derivatives, and the use of one-sided differences results in an asymmetric discretization. Similarly, the *Matched Interface and Boundary* (MIB) *method* [ZZF06] uses higher order extrapolations of the solution matched with higher order one-sided discretizations of the jump conditions to determine the values at fictitious points. The MIB method accounts for nonzero $[\beta \nabla u \cdot \hat{\tau}]$ by differentiating the given jump conditions using one-sided interpolations. This widens the stencil in several directions that depend on the local geometry, and results in an asymmetric discretization. The work of [ZW06] extended the MIB method to handle high curvature geometry, the work of [YW07] provides a 3-dimensional version, and more recent progress is given in [ZW09]. Kejia Pan et al. in [PTH10] derived symmetric finite difference formulas (in 1 and 2 dimensions) within the MIB framework. In [HL05, HWW10] Hou et al. also use techniques seemingly inspired by the analysis of the original GFM approach done in [LFK00, LS03]. They develop a second order variational GFM by altering finite element interpolating functions to capture the jump conditions in the solution. Their approach is remarkably robust to non-smooth interface geometry (especially [HWW10]), but results in an asymmetric discretization in the general case. The recent works of [NMG09a, PGR10] treated the cases of Robin and Neumann boundary conditions by altering the 5-point stencil along the boundary using a finite volume-like approach. This results in a symmetric positive definite system.

Ideas similar to the extrapolation-based finite difference schemes have also seen extensive use in the FEM community, for instance in fictitious domain methods [GPP94a, ABC97, PP09, JM10], the *Discontinuous Galerkin* (DG) *method* [LB08, GLJ09], the *eXtended Finite Element Method* (XFEM) [MDB99, DMD00, BMU01, MCC03, JD04, FB06, MBT06, GR07, BG09] [JSC06][2], and other non-conforming finite element methods [YMB90, HH02, LLW03, HH04, SAB06, MDH07, DF08, KPB08, DH09, HD10, KWC10, WX10]. Fictitious domain methods handle embedded features by including every element that intersects the feature into the discretization. This naturally introduces "virtual nodes" or "ghost nodes" into the resulting discretization. The XFEM enriches the standard finite element basis with additional discontinuous basis functions, thereby introducing new degrees of freedom. These basis functions exist only at the nodes of elements that intersect the embedded interface and usually are the standard basis elements multiplied by a generalized Heaviside function. The

[2]See [BCL08] for corrections to IIM convergence estimates.

methods of [HH02, HH04, SAB06, BHT07, DH09] introduce a related virtual node concept to provide the additional degrees of freedom required to represent the discontinuities. The most straightforward implementation of this virtual node concept [HH04, SAB06, DH09] yields a representation equivalent to the standard Heaviside enrichment of the XFEM. However, this approach generalizes to the slightly richer representations of [MBF05, SDF07] that attain more geometric detail, particularly when dealing with coarse grids and non-smooth interfaces. Moreover, virtual node representations are considered more geometrically intuitive and easier to incorporate into existing FEM code [SAB06, DH09] than traditional Heaviside enrichment.

The solution spaces of these FEM approaches generally do not satisfy the embedded boundary or interface conditions. Thus, these methods impose linear constraints with either penalty methods or Lagrange multipliers to enforce the conditions in some weak sense. For example, see [GPP94a, MDH07, DH09, PP09, BG09] and the references therein. When using Lagrange multipliers, the Ladyzhenskaya-Babuška-Brezzi inf-sup conditions place stringent limitations on the types of constraints that will retain optimal convergence rates of the approximation spaces [Bab73, Pit79, CB93, MBT06, MDH07, LB08]. Such inf-sup restrictions generally limit the strength of the Lagrange multiplier space relative to the solution approximation space. For certain elements, designing the proper approximation spaces is a non-trivial task [JD04, MBT06]. Moreover, the use of Lagrange multipliers requires the solution of an indefinite saddle point system that can potentially introduce significant cost. Applying stabilization through a consistent penalty method, such as Nitsche's method, presents an alternative approach [HH04, MDH07, DF08, DH09, HD10, WX10]. However, these can have adverse effects on conditioning and require the determination of the stabilization parameters. Instead of using Lagrange multipliers or stabilization, the methods of [LLW03, HL05, FB06, Li98b, KPB08, HWW10, KWC10] alter the basis functions to either satisfy the constraints directly, or simplify the process of doing so. In this regard, such methods represent the finite element analogues of the IIM.

The method of [JC98] offers a finite volume approach to embedded domain problems. Like some fictitious domain methods, XFEM, and our virtual node method, this method uses partially empty cells along the boundary. However, the one-sided quadratic interpolations used to compute the fluxes along the boundary yield an asymmetric system. See [SBC06] and [CCG10] for a more recent 3-dimensional version applied to Poisson's equation and the heat equation. In [OK06], Oevermann and Klein proposed a second order finite volume method for interface problems, and simplified and extended their method to 3-dimensions in [OSK09]. In an approach similar to ours, any Cartesian cell that intersects the interface yields a distinct multilinear representation of the solution. The jump conditions are then built into the difference stencil by locally solving constrained overdetermined systems. An asymptotic technique resolves the problem of vanishing cell volumes, though it requires specific treatment for each possible cell geometry. The resulting system is asymmetric for the general case of $[\beta] \neq 0$.

When $[\beta] \neq 0$ the majority of these second order methods do not retain a symmetric positive definite system. While the FEM approaches that use stabilization do retain a symmetric positive definite system [DH09], generally the finite element methods that use Lagrange multipliers, such as [DMD00], result in a symmetric indefinite system. Although

we use Lagrange multipliers, we present a simple method of reducing the indefinite system to a symmetric positive definite system using a null space method. On the other hand, when the coefficient β is smooth across the interface, methods such as the original IIM achieve second order accuracy by only altering the right-hand side of the system. For this case, we present a method that uses the virtual node framework that also retains the original left-hand side.

Several of the above works employ multigrid methods to solve the resulting linear systems. Black-box multigrid solvers, either of a purely algebraic variety [DIL03, OK06, CS07, OSK09] or of a more geometric variety [LI01], are often efficient alternatives to, or may be combined with, Krylov solvers [CS08, MST10]. However, less general multigrid algorithms specially tuned to the particular discretization method may outperform a black-box multigrid solver; see, for example, [AC05, MST10]. Some methods lend themselves to using relatively straightforward extensions of standard geometric multigrid techniques, including both mortar finite element methods [WK99, LW04] and embedded methods [ABC97, JC98, SBC06, CCG10], usually with special attention being paid near irregular features. Many of the works describing IIM-based discretizations [AL02, AC04, AC05, CS08] utilize a multigrid solver with a grid hierarchy defined geometrically but incorporate algebraic techniques in the remaining components (coarse-grid operators and grid transfer operators). In [WL04] Wan and Liu discuss the transfer operators near embedded features in a geometric multigrid method for irregular domain discretizations in general. In contrast to the multigrid approaches in many of the preceding works on embedded discretizations, our multigrid algorithms define the grid hierarchy, coarse-grid operators, and grid transfer operators geometrically, hence allow for efficient implementations that have lower memory requirements and increased parallelizability.

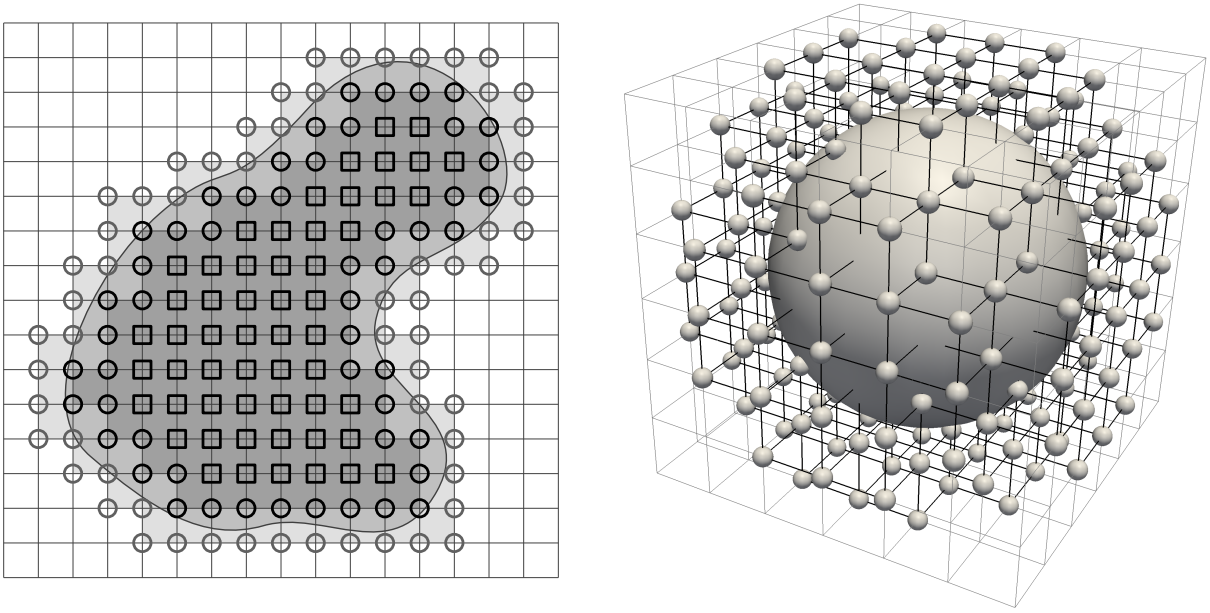
4.2 Discretization

Our numerical discretizations for domain and interface problems make use of an embedding of the domain boundary and/or interface within a uniform Cartesian grid. We thus first outline this embedding procedure and the associated notation. We subsequently describe our Neumann discretization, and we will then see how an alteration of our treatment of the boundary conditions in Neumann problems yields our discretization for Dirichlet problems. Finally, we will show how a natural combination of our Neumann and Dirichlet discretizations allows us to deal with interfacial discontinuities.

4.2.1 Domain and Interface Embedding and Integration

Let us first consider the treatment of Ω for domain problems. We embed the domain Ω into a non-conforming, uniform Cartesian grid $\mathcal{G}^h$ with grid-spacing $(\Delta x, \Delta y, \Delta z)$. (Note that to simplify the convergence analysis, our numerical examples assume $\Delta x = \Delta y = \Delta z =: h$.) We include all Cartesian grid cells c_i that intersect Ω in the discretization, and refer to this set $\mathcal{C}^h = \left\{c_i \in \mathcal{G}^h : c_i \cap \Omega \neq \emptyset\right\}$ as the *computational domain* (see Figure 4.1). Also,

we define the set of all cells that intersect the boundary as $\mathcal{C}^h_{\partial\Omega} = \left\{c_i \in \mathcal{C}^h : c_i \cap \partial\Omega \neq \emptyset\right\}$ and refer to these as *boundary (grid) cells*. Note that a boundary cell may be regarded as partially empty, since only a portion of the cell lies within Ω. We refer to this region of a boundary cell c_i that lies in the domain Ω, $c_i \cap \Omega$, as the *material region* of the cell, and use the terms *material node* and *virtual node* to describe the Cartesian grid vertices lying inside and outside Ω, respectively. We refer to the set of grid vertices spanned by the computation domain as $\mathcal{N}^h$, and specifically the material nodes as $\mathcal{N}^h_m$ and the virtual nodes as $\mathcal{N}^h_v$. See Figure 4.1 for a diagram labeling the grid vertices along a typical boundary. For domain problems, we identify each grid vertex in the computational domain, material or virtual, as a degree of freedom.



(a) Embedding in 2 dimensions

(b) Embedding in 3 dimensions

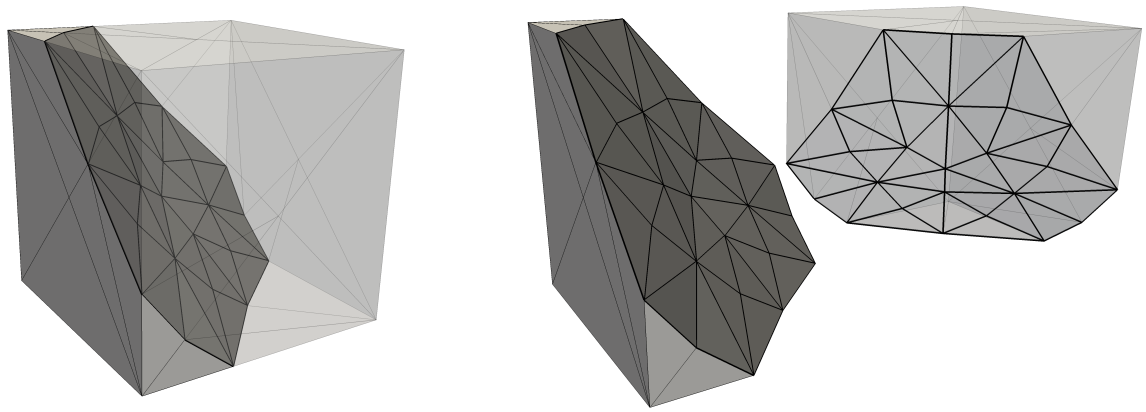
Figure 4.1: Example embeddings for domain problems. Subfigure (a) shows an example in 2 dimensions to clearly depict the various classes of grid cells and vertices: shaded grid cells comprise the computational domain ($\mathcal{C}^h$), with lighter-shaded grid cells on the boundary ($\mathcal{C}^h_{\partial\Omega}$); grid vertices surrounded by gray circles represent virtual degrees of freedom ($\mathcal{N}^h_v$); grid vertices surrounded by black circles represent material degrees of freedom ($\mathcal{N}^h_m$) incident to a boundary grid cell; and grid vertices surrounded by squares represent material degrees of freedom ($\mathcal{N}^h_m$) incident only to non-boundary grid cells. Subfigure (b) shows an example in 3 dimensions.

In the course of the discretization, for each boundary cell $c_i \in \mathcal{C}^h_{\partial\Omega}$, we will need to evaluate integrals over the following domains:

- the material volume within a cell, $c_i \cap \Omega$;
- the boundary of the material volume within a cell, $\partial(c_i \cap \Omega)$; and

- the boundary of Ω within a cell, $c_i \cap \partial\Omega$ (which is contained within $\partial(c_i \cap \Omega)$).

In all cases, the integrand is polynomial (or locally approximated by a polynomial). We evaluate these integrals using polyhedral representations $\mathcal{P}^{c_i}$ and $\mathcal{P}^{c_i}_{\partial\Omega}$ approximating $\partial(c_i \cap \Omega)$ and $c_i \cap \partial\Omega$, respectively. We use the term *polyhedral representation* to convey an analogous meaning as polygonalizing a curve in 2 dimensions, but we essentially regard $\mathcal{P}^{c_i}$ and $\mathcal{P}^{c_i}_{\partial\Omega}$ simply as collections of polygons. For implementation purposes, to maximize data structure reuse, it is convenient for $\mathcal{P}^{c_i}_{\partial\Omega} \subset \mathcal{P}^{c_i}$, i.e., all polygons in $\mathcal{P}^{c_i}_{\partial\Omega}$ are also members of $\mathcal{P}^{c_i}$. See Figure 4.2.



(a) Boundary grid cell with a polyhedralization of a portion of $\partial\Omega$ embedded inside

(b) The two halves of the boundary grid cell after division along $\partial\Omega$

Figure 4.2: A grid cell c_i with an example boundary dividing it. The left half of the cell in (b) corresponds to $c_i \cap \Omega$, the material region of the cell. (b) shows the polyhedralization $\mathcal{P}^{c_i}$ of the material region of the cell, where the shaded triangles highlight $\mathcal{P}^{c_i}_{\partial\Omega} \subset \mathcal{P}^{c_i}$, the polyhedralization just of the portion of $\partial\Omega$ passing through c_i.

We employ the divergence theorem to transform volume integrals over $c_i \cap \Omega$ into surface integrals over $\partial(c_i \cap \Omega)$ (cf. [MG07]). Such transformations are non-unique, but constructing a simple one is straightforward given the polynomial nature of the integrand. For example,

$$\begin{aligned}\int_{c_i \cap \Omega} x^p y^q z^r d\mathbf{x} &= \int_{c_i \cap \Omega} \frac{1}{p+1} \nabla \cdot \left(x^{p+1} y^q z^r, 0, 0\right) d\mathbf{x} \\ &= \int_{\partial(c_i \cap \Omega)} \frac{1}{p+1} \left(x^{p+1} y^q z^r, 0, 0\right) \cdot \hat{\mathbf{n}}(\mathbf{x}) d\mathbf{S}(\mathbf{x}).\end{aligned}$$

We decompose surface integrals over $\partial(c_i \cap \Omega)$ and $c_i \cap \partial\Omega$ into a sum of integrals over the component polygons of $\mathcal{P}^{c_i}$ and $\mathcal{P}^{c_i}_{\partial\Omega}$, respectively. For example, given a vector-valued function $\mathbf{h}(\mathbf{x})$,

$$\int_{\partial(c_i \cap \Omega)} \mathbf{h}(\mathbf{x}) \cdot \hat{\mathbf{n}}(\mathbf{x}) d\mathbf{S}(\mathbf{x}) = \sum_{g \in \mathcal{P}^{c_i}} \int_g \mathbf{h}(\mathbf{x}) \cdot \hat{\mathbf{n}}_g d\mathbf{S}(\mathbf{x}).$$

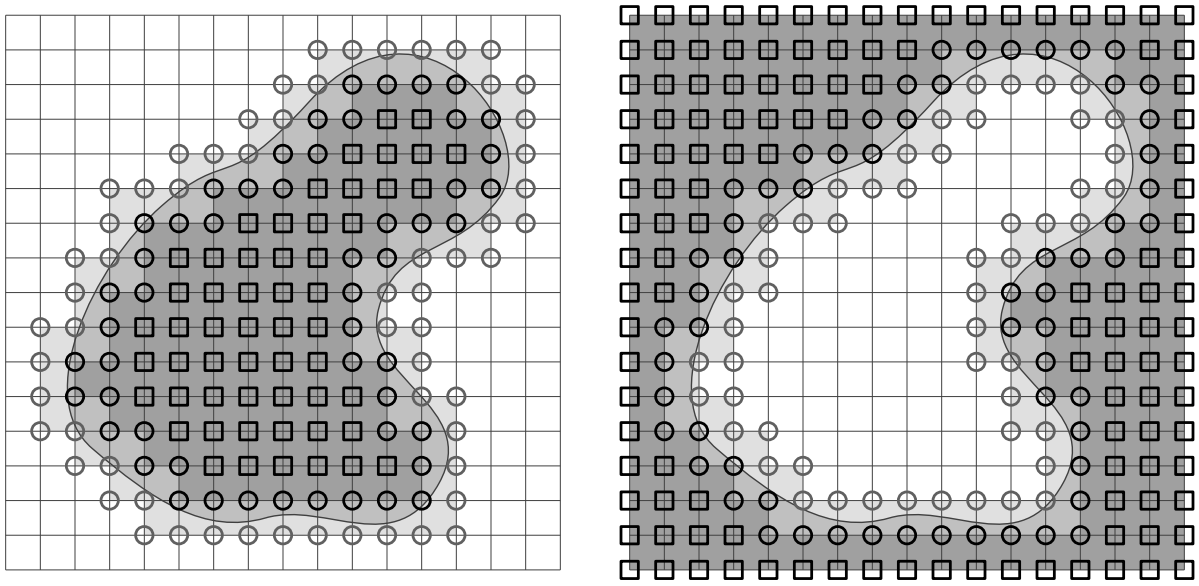
Note that over each polygon $g \in \mathcal{P}^{c_i}$, the unit normal $\hat{\mathbf{n}}_g$ is constant, hence $\mathbf{h}(\mathbf{x}) \cdot \hat{\mathbf{n}}_g$ restricted to g is a polynomial in $\mathbf{x}$ (assuming that the components of $\mathbf{h}$ are polynomials to begin with). To evaluate these polygon-local surface integrals, one could make a change of variables into a localized coordinate system and again apply the divergence theorem. However, the polynomial integrand may have degree as high as 5, and this change of variables requires a computationally intensive expansion of a composition of the integrand with the coordinate transformation. We found it simpler to triangulate each polygon and use a Gaussian quadrature rule over each component triangle. As the polygons in our implementation are limited to triangles and convex quadrilaterals (see §4.2.1.1 below), such a triangulation is trivial. To maximize efficiency while ensuring the quadrature is exact, we use a quadrature rule of order equal to the degree of the polynomial integrand. For specific quadrature rules up to order 5, we refer the reader to Appendix A.

For interface problems, we embed the interface Γ into $\mathcal{G}^h$ in a completely analogous way as for $\partial\Omega$ in domain problems. We likewise use the notation $\mathcal{C}^h_\Gamma = \left\{c_i \in \mathcal{G}^h : c_i \cap \Gamma \neq \emptyset\right\}$ and the term *interfacial (grid) cells* to refer to the set of cells through which the interface passes. As we will see in §4.2.4, our interface discretization is based on a domain discretization, as described above, in each of Ω^- and Ω^+. This naturally introduces an *interior computational domain* $\mathcal{C}^{h,-}$ and *exterior computational domain* $\mathcal{C}^{h,+}$, where $\mathcal{C}^{h,\sigma} = \left\{c_i \in \mathcal{G}^h : c_i \cap \Omega^\sigma \neq \emptyset\right\}$. Note that $\mathcal{C}^{h,-}$ and $\mathcal{C}^{h,+}$ are disjoint save for $\mathcal{C}^h_\Gamma$, where each Cartesian grid cell and the associated degrees of freedom, material and virtual, are duplicated. See Figure 4.3.

We will often speak generically about both our interface discretization and our domain (with Neumann and/or Dirichlet boundary conditions) discretizations. Due to the similarities in the embedding of $\partial\Omega$ (for domain problems) and of Γ (for interface problems) into the background grid $\mathcal{G}^h$, and to avoid cluttering the exposition with too many "boundary/interface" terms, we will occasionally simply use the term *embedded feature* or *embedded geometry* to refer both to the embedded boundary $\partial\Omega$ in domain problems and to the embedded interface Γ in interface problems.

4.2.1.1 Embedded feature polyhedralization

We define all of the domains Ω and interfaces Γ in the numerical examples in §4.4 analytically and implicitly as the zero isocontour of a level set function. This Eulerian representation ensures that we can always resolve embedded features to a resolution comparable to the background grid $\mathcal{G}^h$. Note that the embedding procedure and integration techniques described above require a Lagrangian-like polyhedral representation of the embedding within each boundary or interfacial grid cell. Thus, one must create some polyhedral approximation, per such grid cell, of the implicitly defined embedded geometry. Since it is relatively easy to divide a tetrahedron along a plane approximating the level set surface given the level set function values at the tetrahedron's vertices, we symmetrically partition each boundary or interfacial grid cell into 24 congruent tetrahedra and accordingly divide each tetrahedron. The union of these dividing surfaces (triangles and quadrilaterals) within each tetrahedron compose the polyhedral representation of the embedded geometry. In 2 dimensions, the analogous procedure would be to partition each square grid cell into 4 triangles and divide



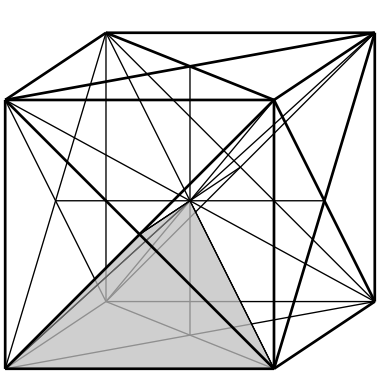
(a) Embedding for Ω^-

(b) Embedding for Ω^+

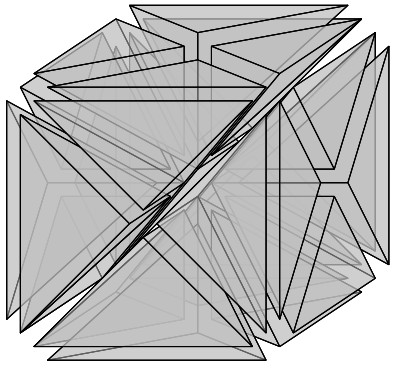
Figure 4.3: An example interface embedding in 2 dimensions, showing the separate domain embeddings for Ω^- and Ω^+. Grid cells and grid vertices are labelled as in Figure 4.1: shaded grid cells comprise the interior (Ω^-, (a)) and exterior (Ω^+, (b)) computational domains, with the lighter-shaded grid cells on the interface; grid vertices surrounded by gray circles represent virtual degrees of freedom; grid vertices surrounded by black circles represent material degrees of freedom incident to an interfacial grid cell; and grid vertices surrounded by squares represent material degrees of freedom incident only to non-interfacial grid cells. Notice how all interfacial grid cells and circled grid vertices are effectively duplicated between the grids embedding the interior and exterior domains. Also note that each grid vertex on an interfacial grid cell is duplicated into precisely one material degree of freedom and one virtual degree of freedom.

each triangle by a line according to the level set function values at the triangle's vertices. This polyhedralization procedure is similar to that described in [MG07]. See Figure 4.4.

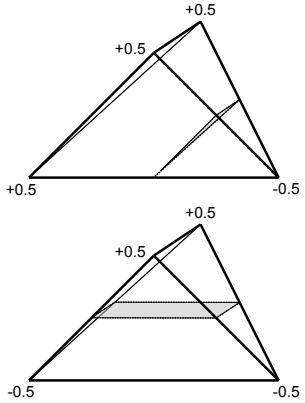
The procedure described above may produce a *sliver* polyhedron (a polyhedron with large aspect ratio) when dividing a given tetrahedron; likewise, the polygonal representation of the embedded surface may contain some sliver polygons. We note that the aspect ratio of such primitives has no *direct* bearing on the conditioning of the discretization. The quantities of actual relevance to conditioning are the measures of the material volume and the embedded surface within a boundary or interfacial grid cell. Unlike the conditioning issues associated with sliver elements in a conforming mesh, however, our method allows conditioning issues caused by vanishing material volume measures within a grid cell to be addressed via Jacobi preconditioning, as we discuss at the end of §4.2.2. Further, our constraint aggregation method described in §4.2.3.2 fully alleviates any conditioning issues caused by vanishing



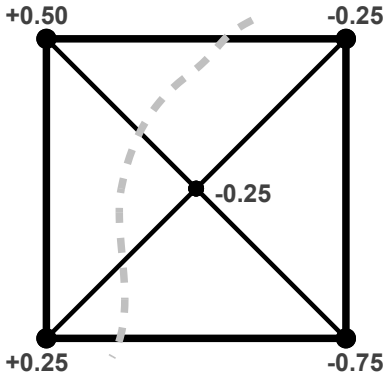
(a) Grid cell partitioned into 24 congruent tetrahedrons (wireframe)



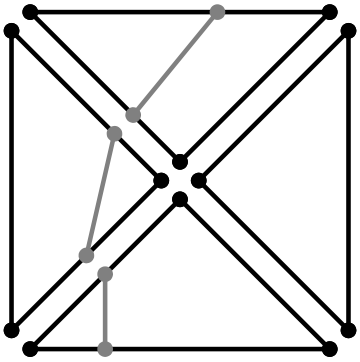
(b) Grid cell partitioned into 24 congruent tetrahedrons (separated)



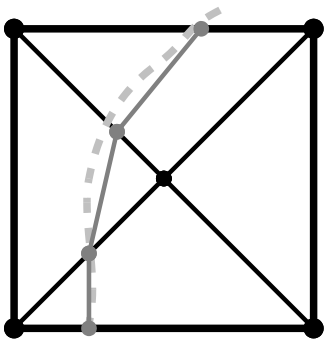
(c) Typical divisions of a tetrahedron given the level set function values at its vertices



(d) Grid cell (dimension 2) partitioned into 4 congruent triangles,with a typical level set curve (lightgray; $\partial\Omega$ or Γ) and level set function values at the vertices of the triangles.



(e) The division of each triangle according to the level set function values at its vertices.



(f) The polygonal representation (gray) of the level set curve is the union of the dividing segments within each triangle.

Figure 4.4: We approximate an embedded domain boundary or embedded interface implicitly defined by a level set function with a polyhedral representation computed by partitioning each boundary or interfacial grid cell into 24 congruent tetrahedra, as in (a) and (b); and subsequently dividing each tetrahedron according to the level set function values at its vertices, e.g., as in (c). The union of the dividing triangles and quadrilaterals within each divided tetrahedron compose the polyhedral representation of the embedded boundary or embedded interface. In 2 dimensions, the analogous procedure would be to partition each square grid cell into 4 triangles, as in (d), and divide each triangle according to the level set values at its vertices, as in (e). The union of the dividing segments within each triangle compose the polygonal representation of the embedded boundary or interface, as in (f).

embedded surface measures within a grid cell (which are only relevant within the context of discretizing the Dirichlet boundary conditions (4.1d) and the value jump interface conditions (4.1b)). See also [LB08] for a more detailed discussion on the advantages, with respect to conditioning, of using embedded domain methods over conforming mesh methods such as locally boundary-fitting remeshing schemes.

4.2.2 Embedded Neumann

Our discretization of Neumann problems is a generalization of the 2-dimensional method given by Bedrossian et al. [BBZ10], and is similar to some XFEM approaches, e.g., [DMD00], as well as the early work of Almgren et al. in [ABC97]. We discretize the Neumann problem,

$$\begin{aligned} -\nabla \cdot (\beta(\mathbf{x})\nabla u(\mathbf{x})) &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega; \\ \beta(\mathbf{x})\nabla u(\mathbf{x}) \cdot \hat{\mathbf{n}} &= q(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega; \end{aligned} \tag{4.2}$$

using the energy minimization form of (4.2):

$$\text{over all } u \in H^1(\Omega), \text{ minimize}$$

$$E(u) := e(u) - (f,u)_\Omega - (q,u)_{\partial\Omega} := \int_\Omega \frac{1}{2}\nabla u \cdot \beta \nabla u d\mathbf{x} - \int_\Omega f u d\mathbf{x} - \int_{\partial\Omega} q u d\mathbf{S}(\mathbf{x}). \tag{4.3}$$

We choose to discretize the energy minimization problem because this straightforwardly yields a symmetric system; it naturally incorporates the Neumann boundary conditions into the right-hand side of the system; and it provides the necessary setting to ensure accuracy of the discretization near the boundary. We define the solution space $V^h \subset H^1(\Omega)$ as the space of continuous functions that are trilinear over the material region of each cell $c_k \in \mathcal{C}^h$. For $u^h \in V^h$, we write $u^h(\mathbf{x}) = \sum_{i=1}^n u_i N_i(\mathbf{x})$ for $\vec{u} = (u_1, \ldots, u_n)^t \in \mathbb{R}^n$. Here $N_i(\mathbf{x})$ is the standard piecewise trilinear interpolation basis function associated with grid vertex i; and n denotes the number of degrees of freedom in the discretization, equal to the number of grid vertices that compose the cells of $\mathcal{C}^h$.

Using the above representation of $u^h \in V^h$, we define a discrete energy $E^h(u^h)$ approximating $E(u^h)$. Although we could discretize the energy directly from the piecewise trilinear representation of u^h, this would result in a 27-point stencil everywhere, even away from the boundary. To retain the standard second order 7-point stencil away from the boundary we use different discretizations of the energy over $\mathcal{C}^h \setminus \mathcal{C}^h_{\partial\Omega}$ and over $\mathcal{C}^h_{\partial\Omega}$,

$$E^h(u^h) := \sum_{c_k \in \mathcal{C}^h \setminus \mathcal{C}^h_{\partial\Omega}} e^{c_k}(u^h) + \sum_{c_k \in \mathcal{C}^h_{\partial\Omega}} \tilde{e}^{c_k}(u^h) - \sum_{c_k \in \mathcal{C}^h} (f, u^h)^{c_k}_\Omega - \sum_{c_k \in \mathcal{C}^h_{\partial\Omega}} (q, u^h)^{c_k}_{\partial\Omega}, \tag{4.4}$$

where the superscripts denote restriction to cell c_k. For cells $c_k \in \mathcal{C}^h \setminus \mathcal{C}^h_{\partial\Omega}$ that *do not* intersect the boundary, we define $e^{c_k}(u^h)$ as

$$e^{c_k}(u^h) := \frac{1}{2}\overline{\beta}\Delta x \Delta y \Delta z \left((D_x u^h)^2 + (D_y u^h)^2 + (D_z u^h)^2 \right).$$

Here $\overline{\beta}$ denotes a cell average of β; and $(D_x u^h)^2$ denotes the average of the squared finite difference approximations of $\partial_x u^h$ over the 4 x-oriented edges in the cell:

$$(D_x u^h)^2 := \frac{1}{4} \sum_{s,t \in \{0,1\}} \left(\frac{u_{i+1,j+s,k+t} - u_{i,j+s,k+t}}{\Delta x} \right)^2,$$

where $\{u_{p,q,r}\}$ denote the degrees of freedom at the 8 corners of the cell. $(D_y u^h)^2$ and $(D_z u^h)^2$ likewise denote approximations to $(\partial_y u^h)^2$ and $(\partial_z u^h)^2$, respectively. On the other hand, for cells $c_k \in \mathcal{C}^h_{\partial\Omega}$ that *do* intersect the boundary, we use the Cartesian trilinear representation of u^h to define $\tilde{e}^{c_k}(u^h)$. If we let $\mathcal{N}^h_{c_k}$ denote the indices of the 8 vertices at the corners of the cell c_k, and let $\left\{N_i : i \in \mathcal{N}^h_{c_k}\right\}$ denote the corresponding trilinear basis functions, then this yields the discretization

$$\tilde{e}^{c_k}(u^h) := \frac{1}{2} \sum_{i,j \in \mathcal{N}^h_{c_k}} \left(\overline{\beta} \int_{c_k \cap \Omega} \nabla N_i \cdot \nabla N_j d\mathbf{x} \right) u_i u_j. \tag{4.5}$$

Note that $\nabla N_i \cdot \nabla N_j$ is a 4^{th}-degree polynomial, hence we can evaluate these integrals as described in §4.2.1. Like the integrals, the cell average of β, $\overline{\beta}$, is computed only over the material region of the cell, $c_k \cap \Omega$.

We discretize the remaining forms cell-wise, as:

$$\begin{aligned}
(f, u^h)^{c_k}_{\Omega} &:= \sum_{i \in \mathcal{N}^h_{c_k}} \left(\overline{f} \int_{c_k \cap \Omega} N_i d\mathbf{x} \right) u_i; \\
(q, u^h)^{c_k}_{\partial\Omega} &:= \sum_{i \in \mathcal{N}^h_{c_k}} \left(\overline{q} \int_{c_k \cap \partial\Omega} N_i d\mathbf{S}(\mathbf{x}) \right) u_i.
\end{aligned}$$

Similar to $\overline{\beta}$, $\overline{f}$ is the average source over $c_k \cap \Omega$, and $\overline{q}$ is the average normal flux over $c_k \cap \partial\Omega$. Again, all integrals above have polynomial integrands, hence we can evaluate these integrals as described in §4.2.1. See Appendix B for details on how we computed $\overline{\beta}$, $\overline{f}$, and $\overline{q}$ for the numerical examples in §4.4.

Lastly, we minimize the discrete energy (4.4) by solving the linear system

$$\begin{aligned}
A\vec{u} &= \vec{f}, \\
A_{ij} &:= \frac{\partial^2}{\partial u_i \partial u_j} E^h(u^h), \\
f_i &:= \frac{\partial}{\partial u_i} \left((f, u^h)_{\Omega} + (q, u^h)_{\partial\Omega} \right)
\end{aligned} \tag{4.6}$$

for the vector $\vec{u}$. We use the standard FEM term *stiffness matrix* to refer to the matrix A, and it is clear from the derivation that A is symmetric and positive semi-definite. Indeed, its null space is spanned by the vector $\vec{u} = (1, 1, \ldots, 1)^t$ corresponding to $u^h \equiv 1$.

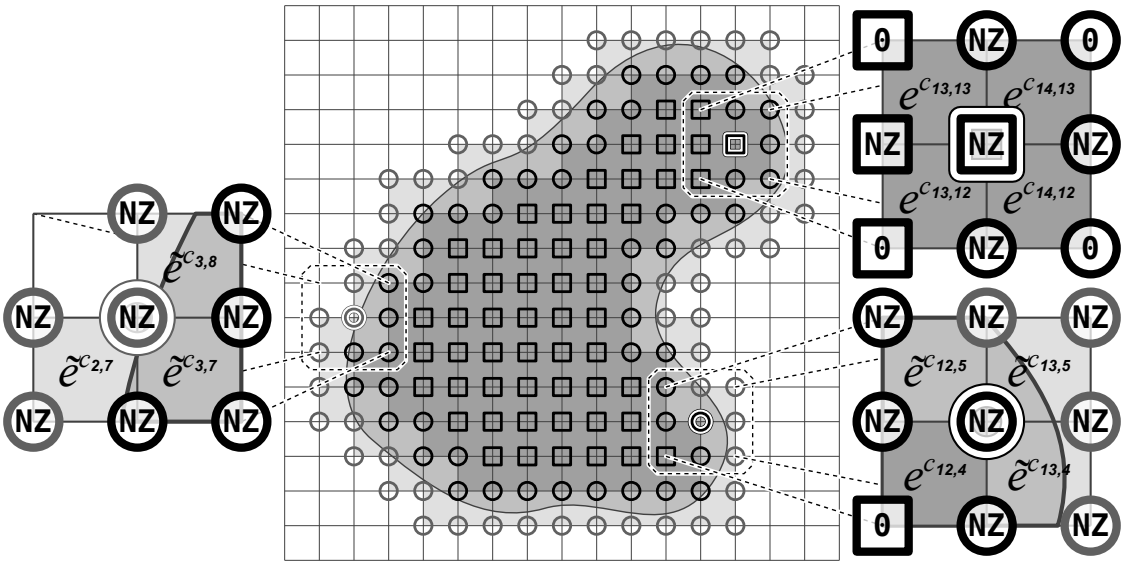


Figure 4.5: Illustration in 2 dimensions of the stiffness matrix (A) stencils for various grid vertices. The stencil for a degree of freedom indicates where the nonzero (NZ) entries are of the row (or column) in A corresponding to the degree of freedom. Squared grid vertices have the standard finite difference Poisson stencil (a 5-point stencil in 2 dimensions; a 7-point stencil in 3 dimensions), which naturally arises through the use of e^{c_k} to discretize the energy (4.3). Circled grid vertices (both black and gray) will generally have a denser stencil (up to a 9-point stencil in 2 dimensions; up to a 27-point stencil in 3 dimensions), due to the use of $\tilde{e}^{c_k}$.

With this approach, our definition of the energy (4.5) results in a slightly denser stencil near the boundary, as all 8 degrees of freedom in a cell couple together if $\partial\Omega$ passes through that cell. See Figure 4.5 for a graphical depiction of the stencil definitions and the sparsity pattern of the stiffness matrix.

The symmetric system (4.6) readily lends itself to black-box solvers such as (preconditioned) conjugate gradient. However, conditioning of the stiffness matrix may deteriorate when a cell has a very small material volume measure, as we first mentioned in §4.2.1.1. This arises from the increasing irrelevance of virtual nodes far from the boundary (see, for example, the $(4, 12)$ grid vertex in Figure 4.5). The respective row and column in A and the corresponding entry in $\vec{f}$ all approach zero simultaneously. We found that simple Jacobi preconditioning (and, in extreme cases, outright elimination of degrees of freedom; see §4.4 for explanation) mitigates these conditioning issues as in [BBZ10]. Note however that our multigrid solver described in §4.3 naturally suffers no such adverse effects from A's conditioning.

4.2.3 Embedded Dirichlet

Following the progression in [BBZ10], we extend our Neumann discretization to solve Dirichlet problems,

$$\begin{aligned} -\nabla \cdot (\beta(\mathbf{x})\nabla u(\mathbf{x})) &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ u(\mathbf{x}) &= p(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \end{aligned} \tag{4.7}$$

within our virtual node framework. We will show how a further extension will naturally yield a discretization for interface problems, resulting in a method that encapsulates all types of boundary conditions in a unified framework.

For Dirichlet boundary conditions, we use the constrained minimization problem:

$$\text{over all } u \in H^1(\Omega), \text{ minimize}$$

$$E(u) := e(u) - (f, u)_\Omega \quad \text{such that} \tag{4.8}$$

$$(u, \mu)_{\partial\Omega} = (p, \mu)_{\partial\Omega} \quad \forall \mu \in H^{-1/2}(\partial\Omega). \tag{4.9}$$

where $e(\cdot)$, $(\cdot,\cdot)_\Omega$, and $(\cdot,\cdot)_{\partial\Omega}$ are as in (4.3).

We discretize the energy (4.8) exactly as for the Neumann case, so the only difference comes in discretizing the constraints (4.9). We proceed by selecting a finite-dimensional subspace (the discrete Lagrange multiplier space) $\Lambda^h \subset H^{-1/2}(\partial\Omega)$, and enforce (4.9) for all $\mu^h \in \Lambda^h$. Not all plausible choices of Λ^h will yield an acceptably accurate approximation, as, in general, (Λ^h, V^h) must satisfy an inf-sup stability criterion to retain the optimal convergence rates of the approximation spaces [Pit79]. One possible choice for Λ^h, which we shall refer to as Λ_1^h and is used in, for instance, [JSC06] and [MDH07], defines μ^h as piecewise constant over each Cartesian grid cell c_i intersecting the boundary $\partial\Omega$ (see Figure 4.6). In other words, we define $\mu^h \in \Lambda_1^h$ as

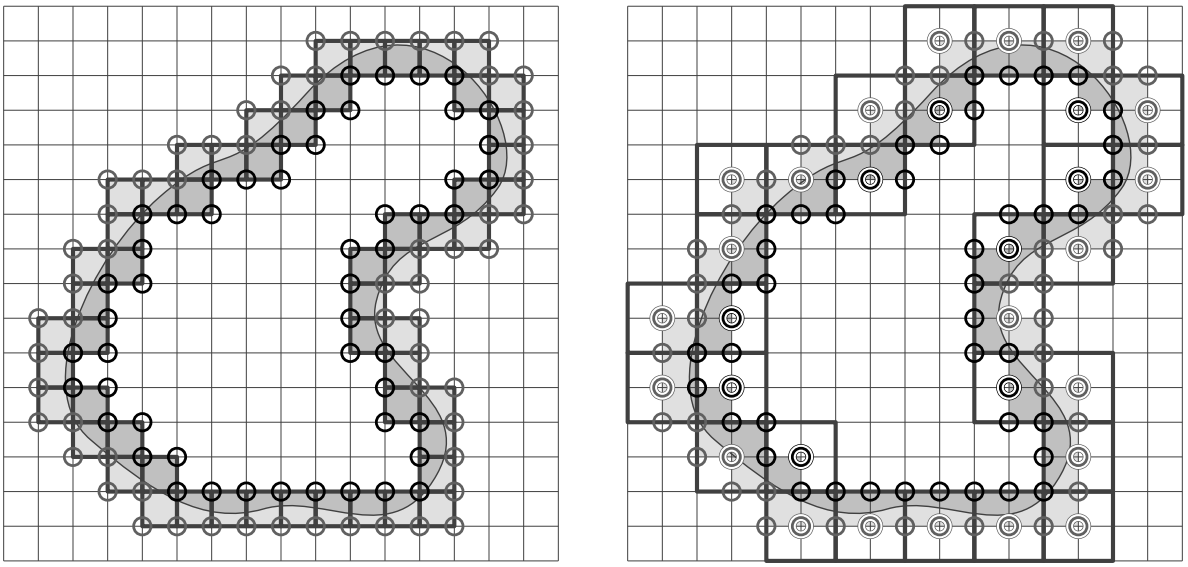
$$\mu^h(\mathbf{x}) := \sum_{c_i \in \mathcal{C}_{\partial\Omega}^h} \mu_i \chi_{c_i \cap \partial\Omega}(\mathbf{x}),$$

where the characteristic functions $\chi_{c_i\cap\partial\Omega}$ are given by

$$\chi_{c_i\cap\partial\Omega}(\mathbf{x}) := \begin{cases} 1, & \mathbf{x} \in c_i \cap \partial\Omega \\ 0, & \mathbf{x} \notin c_i \cap \partial\Omega \end{cases}. \tag{4.10}$$

With this choice of discrete Langrange multiplier space, satisfying (4.9) for all $\mu^h \in \Lambda^h = \Lambda_1^h$ yields a system of sparse linear constraints $B\vec{u} = \vec{p}$ on the coefficient vector $\vec{u}$ of the approximate solution u^h. Each row of the matrix B corresponds to a cell $c_i \in \mathcal{C}_{\partial\Omega}^h$ and enforces the condition

$$\int_{c_i\cap\partial\Omega} u^h(\mathbf{x}) d\mathbf{S}(\mathbf{x}) = \int_{c_i\cap\partial\Omega} p(\mathbf{x}) d\mathbf{S}(\mathbf{x}). \tag{4.11}$$



(a) Schematic of functions in Λ_1^h, with single-wide constraints C_1

(b) Schematic of functions in Λ_2^h, with double-wide constraints C_2

Figure 4.6: Schematics of two discretizations Λ^h of the Lagrange multiplier space $H^{-1/2}(\partial\Omega)$ in 2 dimensions used in (4.9). (a) shows a schematic of functions in Λ_1^h, which are piecewise constant over $\mathcal{C}_{\partial\Omega}^h \cap \partial\Omega$. (b) shows a schematic of functions in Λ_2^h, which are piecewise constant over $\mathcal{C}_{\partial\Omega}^{2h} \cap \partial\Omega$ (using the doubly-coarse grid $\mathcal{G}^{2h}$). Note that the center grid vertex (highlighted) in each doubly-coarse boundary grid cell is an independent degree of freedom with respect to C_2, the constraints induced by Λ_2^h. That is, the center grid vertex in a doubly-coarse boundary grid cell participates only in the constraint corresponding to that cell.

Therefore, if $\mathcal{C}_{\partial\Omega}^h = \{c_1, \ldots, c_m\}$ and $\vec{u} \in \mathbb{R}^n$, then $\vec{p} \in \mathbb{R}^m$, $B \in \mathbb{R}^{m \times n}$, and

$$B_{ij} := \int_{c_i \cap \partial\Omega} N_j(\mathbf{x}) d\mathbf{S}(\mathbf{x}) \tag{4.12}$$

for each Cartesian trilinear basis function $N_j(\mathbf{x})$. Since only 8 of these basis functions are supported over a given $c_i \cap \partial\Omega$, each row of B contains precisely 8 nonzero entries. The corresponding entry in $\vec{p}$ is

$$p_i := \int_{c_i \cap \partial\Omega} p(\mathbf{x}) d\mathbf{S}(\mathbf{x}). \tag{4.13}$$

As before, we evaluate these integrals as described in §4.2.1 (using a suitable polynomial approximation for $p(\mathbf{x})$ in each grid cell or a suitable quadrature rule to evaluate (4.13)). Discretizing (4.8, 4.9) thus gives rise to the quadratic program:

$$\text{minimize over } \vec{u} \in \mathbb{R}^n$$

$$E^h(u^h) := e(u^h) - (f, u^h)_\Omega := \frac{1}{2}\vec{u^t} A\vec{u} - \vec{f}^t\vec{u} \tag{4.14}$$

$$\text{subject to } B\vec{u} = \vec{p}.$$

The matrix A is exactly as for the embedded Neumann case described in §4.2.2, as is the vector $\vec{f}$ excepting the contribution of the Neumann constraint q (see (4.6)). This minimization problem may equivalently be expressed as a saddle point system, introducing a Lagrange multiplier $\vec{\lambda}$:

$$\begin{pmatrix} A & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ \vec{\lambda} \end{pmatrix} = \begin{pmatrix} \vec{f} \\ \vec{p} \end{pmatrix}. \tag{4.15}$$

4.2.3.1 Null space method and fundamental basis of constraint system

As is done in [BBZ10], we solve (4.14) / (4.15) using a *null space method*, which efficiently transforms our problem into a symmetric positive definite linear system. This affords us a wide variety of solution techniques, including black-box solvers such as (preconditioned) conjugate gradient; and a large class of preconditioners, such as incomplete Cholesky (which we use for many of the numerical examples in §4.4). This derived symmetric positive definite system also readily lends itself as a basis for a multigrid smoother such as Gauss-Seidel (as presented in §4.3). For these reasons, our null space approach has significant advantages over alternative approaches such as Schur's complement reduction, direct methods applied to the saddle point system (4.15), stationary methods such as Uzawa's method, penalty methods, or Krylov methods applied to (4.15). Those aforementioned approaches which are iterative typically require solving a linear system at each iteration and/or have slow convergence properties. Direct methods tend to be too computationally expensive and memory intensive when applied to large systems. Preconditioning saddle point systems such as (4.15) directly is much less well-developed are of research than preconditioning symmetric positive definite systems; hence, applying a Krylov method to (4.15) is less appealing than applying a Krylov method to an equivalent symmetric positive definite system. For a more complete survey of the advantages and disadvantages of these and other approaches, see [BGL05].

The null space method requires the construction of a matrix Z whose columns span the null space of B and a vector $\vec{c} \in \mathbb{R}^n$ satisfying the discretized constraints (i.e., $B\vec{c} = \vec{p}$). Our solution $\vec{u}$ to (4.14) or (4.15) may then be expressed as $\vec{u} = \vec{c} + Z\vec{v}$ for some $\vec{v}$, and substituting this expression for u into (4.15) (and eliminating $\vec{\lambda}$ via left multiplication by Z^t) yields the system $Z^t AZ\vec{v} = Z^t(\vec{f} - A\vec{c})$ for $\vec{v}$. As noted in §4.2.2, the null space of A is spanned by the vector $(1, 1, \ldots, 1)^t \in \mathbb{R}^n$, and the entries of B are all non-negative, so $\ker(A) \cap \ker(B) = \{\vec{0}\}$. Therefore, $Z^t AZ$ is non-singular and, specifically, symmetric positive definite. We have thus transformed (4.14)/(4.15) into a symmetric positive definite system for $\vec{v}$. We obtain $\vec{u}$ by setting $\vec{u} = \vec{c} + Z\vec{v}$.

We now address the determination of Z. Obtaining Z through a QR factorization or a SVD is likely to be computationally expensive and, moreover, produce a dense Z. A *fundamental basis* presents an alternative to numerical factorization [BGL05]. The matrix

B is full rank if and only if an ordering of the degrees of freedom exists so that B may be expressed as $B = (B_m|B_{n-m})$ for some $m \times m$ non-singular matrix B_m. Any such ordering gives the corresponding fundamental basis

$$Z = \begin{pmatrix} -B_m^{-1}B_{n-m} \\ I_{n-m} \end{pmatrix}. \tag{4.16}$$

Clearly, $BZ = 0$ and the vector $\vec{c} = \begin{pmatrix} B_m^{-1}\vec{p} \\ 0 \end{pmatrix}$ satisfies $B\vec{c} = \vec{p}$. Therefore, if we can solve systems of the form

$$B_m\vec{x} = \vec{d} \tag{4.17}$$

efficiently, we can store the factors B_m, B_{n-m}, and A sparsely and compute the action of Z^tAZ readily (e.g., for use in conjugate gradient). Note that, regardless of the choice of B_m, the symmetric positive definite stencil defined by Z^tAZ coincides with the standard 7-point stencil for all degrees of freedom sufficiently distanced from the boundary.

4.2.3.2 Aggregation of single-wide constraints

Unfortunately, as discussed in [BBZ10], the choice of Λ_1^h (the space of functions that are piecewise constant over each boundary grid cell) as the discrete Lagrange multiplier space approximating $H^{-1/2}(\partial\Omega)$ makes it difficult (if not impossible) to determine an ordering of the degrees of freedom that gives a well-conditioned and easily invertible B_m. Bedrossian et al. [BBZ10] give an ordering of the degrees of freedom and of the constraints that yields an upper-triangular B_m; however, although the resulting system (4.17) can theoretically be efficiently solved by back-substitution, in practice such a solution procedure introduces prohibitively large numerical errors for anything but the smallest grids.

As in [BBZ10], we remedy this by using an alternative approximation to $H^{-1/2}(\partial\Omega)$ that induces a different set of linear constraints. To motivate our approach, suppose we define a set of m linear constraints (other than those induced by Λ_1^h) such that each constraint contains an *independent* degree of freedom, a degree of freedom which participates only in that one constraint. Observe, then, that ordering these m independent degrees of freedom first, in matching order with their associated constraints, yields a *diagonal* B_m, which is trivial to invert. As the constraints induced by Λ_1^h generally have an insufficient number of independent degrees of freedom, we thus aim to manufacture an alternative discrete Lagrange multiplier space such that the induced set of constraints admits such a set of independent degrees of freedom, and hence gives a diagonal B_m. For example, Bedrossian et al. [BBZ10] uses $\Lambda_1^{2h} =: \Lambda_2^h$ (the set of scalar piecewise constant functions over the cells of the doubly-coarse grid $\mathcal{G}^{2h}$; see Figure 4.6) as an approximation to $H^{-1/2}(\partial\Omega)$, leading to what may be described as *double-wide constraints*. Each double-wide constraint encompasses a 2×2 (in 2 dimensions) or $2 \times 2 \times 2$ (in 3 dimensions) block of cells. The center vertex in such a block of cells always participates only in the associated constraint, hence these center vertices correspond to independent degrees of freedom. Double-wide constraints are acceptable for problems in 2 dimensions, as investigated by Bedrossian et al. [BBZ10]; however, the

structural rigidity of Λ_2^h presents conditioning issues in 3 dimensions (see Appendix C for a specific example). One of our major contributions is a more general, flexible approach toward constructing constraints which gives greater control on conditioning, and for which the double-wide constraints induced by Λ_2^h will be a special case.

The key idea is that rather than first defining the set of constraints and then selecting an independent degree of freedom from each constraint, we will first select the set of independent degrees of freedom and then subsequently build a single constraint equation around each independent degree of freedom. To this end, let C_1 denote the set of *single-wide constraints* (4.11) induced by Λ_1^h, as described above; and let G denote the adjacency graph induced by C_1, as depicted in Figure 4.7(a). That is, two degrees of freedom are *adjacent* in G if they simultaneously participate in some single-wide constraint; or, in more geometric terms, two grid vertices are adjacent in G if they share a common incident boundary grid cell. Choose $m_a < m$ degrees of freedom which constitute an *independent set* $\mathcal{I}$ with respect to G. In other words, no two degrees of freedom in $\mathcal{I}$ will simultaneously participate in the same single-wide constraint. An example of such an independent set is given in 4.7(b). Now associate each of the m single-wide constraints in C_1 to one of these independent degrees of freedom in $\mathcal{I}$, with the provision that, if a constraint contains an independent degree of freedom, it must be associated with said independent degree of freedom. (This latter requirement is conflict-free, as any single-wide constraint in C_1 will contain at most one independent degree of freedom, by construction.) Thus, for those single-wide constraints containing an independent degree of freedom, this association is precisely determined. However, some single-wide constraints may contain no independent degree of freedom, so some additional heuristic must be used to determine this association. See figures 4.7(c) and 4.7(d) for an example association of each single-wide constraint to an independent degree of freedom.

Let $\mathcal{I} = \{d_1, \ldots, d_{m_a}\}$ denote the independent set of degrees of freedom; and let $C_{d_i} \subset C_1$ denote the set of single-wide constraints associated with independent degree of freedom d_i, such that $\bigsqcup_i C_{d_i} = C_1$. We then form the following m_a *aggregate constraint* equations:

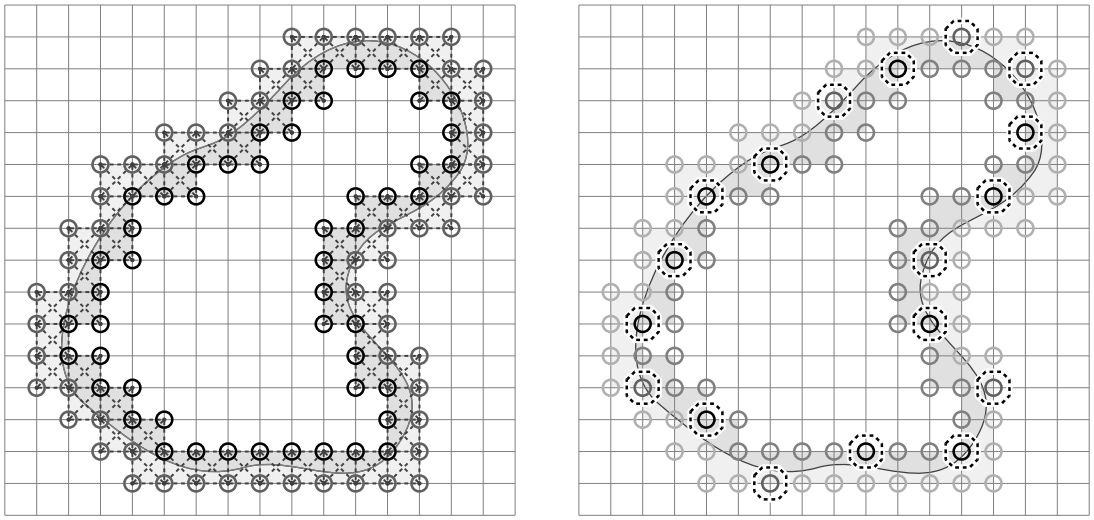
$$\sum_{c_k \in C_{d_i}} \int_{c_k} u^h(\mathbf{x}) d\mathbf{S}(\mathbf{x}) = \sum_{c_k \in C_{d_i}} \int_{c_k} p(\mathbf{x}) d\mathbf{S}(\mathbf{x}) \tag{4.18}$$

where $c_k \in C_{d_i}$ denotes that cell c_k corresponds to a single-wide constraint associated with independent degree of freedom d_i. Effectively, the single-wide constraint equations in C_1 associated to a given independent degree of freedom are summed into a single aggregate constraint equation. Likewise, the corresponding discrete Lagrange multiplier space Λ_a^h is spanned by sums of the basis functions $\chi_{c_k \cap \partial\Omega}$ of Λ_1^h from (4.10):

$$\mu^h(\mathbf{x}) := \sum_{d_i \in \mathcal{I}} \mu_i \sum_{c_k \in C_{d_i}} \chi_{c_k \cap \partial\Omega}.$$

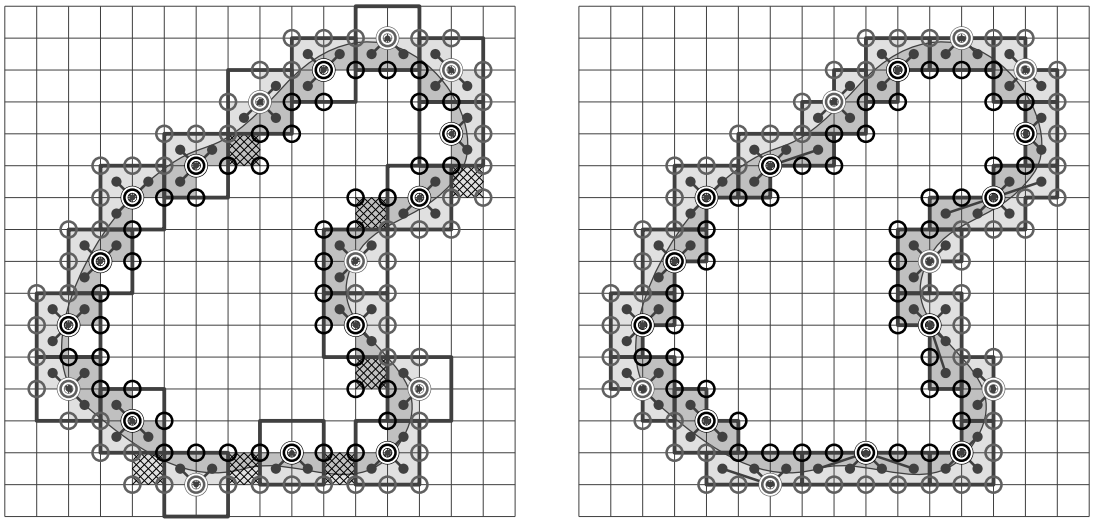
Now let B and $\vec{p}$ denote the matrix and right-hand side of the system of aggregate constraints (4.18):

$$B_{ij} := \sum_{c_k \in C_{d_i}} \int_{c_k \cap \partial\Omega} N_j(\mathbf{x}) d\mathbf{S}(\mathbf{x}), \quad p_i := \sum_{c_k \in C_{d_i}} \int_{c_k \cap \partial\Omega} p(\mathbf{x}) d\mathbf{S}(\mathbf{x}). \tag{4.19}$$



(a) Adjacency graph G induced by the set of single-wide constraints C_1. Boundary grid vertices are adjacent with respect to G if they share a common incident boundary grid cell.

(b) Example selection of an independent set $\mathcal{I}$ of degrees of freedom. No two independent boundary grid vertices share a common incident boundary grid cell; equivalently, no two independent degrees of freedom simultaneously participate in the same single-wide constraint.



(c) Associating single-wide constraints to a participating independent degree of freedom. Some constraints (identified by cross-hatching) contain no independent degree of freedom; one must resort to some additional heuristic to associate these constraints.

(d) Associating the remaining single-wide constraints to a nearby independent degree of freedom (using some implementation-defined heuristic) and the final set of aggregate constraints.

Figure 4.7: Illustrated progression of the constraint aggregation described in §4.2.3.2.

Clearly, by construction, this set of aggregate constraints admits an ordering of the degrees of freedom to give a diagonal B_{m_a}: just order the independent degrees of freedom first.

In summary, the above procedure aggregates the single-wide constraints C_1 to yield an alternative set of constraints C_a which admits an ordering of the degrees of freedom to give a diagonal B_{m_a}. We have thus far described this constraint aggregation in very general terms, and there indeed remains a great deal of flexibility, particularly in how one chooses the set of independent degrees of freedom. For example, selecting all degrees of freedom which exist in the doubly-coarse grid $\mathcal{G}^{2h}$ as independent degrees of freedom leads to the double-wide constraints C_2 mentioned earlier. For simplicity, in the following discussion, we consider only strategies which select independent degrees of freedom one at a time and greedily, noting that alternative approaches could very well yield equal or superior results. Such a constraint aggregation implementation may be described by the following parameters.

- One should decide how the degrees of freedom should be ordered or prioritized for consideration for inclusion in the independent set.
- We need some condition on which to terminate the further selection of independent degrees of freedom.
- Once we have selected the set of independent degrees of freedom, we must associate an independent degree of freedom to each otherwise unassociated single-wide constraint (a constraint containing no independent degree of freedom).

For purposes of selecting independent degrees of freedom, we found that weighting degrees of freedom by the sum of the their coefficients across all single-wide constraints (i.e., the weight of the j^{th} degree of freedom is $\sum_i B_{ij}$) gives good results. Thus, in each iteration, we select, for inclusion in the independent set, the degree of freedom with the largest weight, taking care to exclude degrees of freedom adjacent to previously selected independent degrees of freedom. The motivation for using $\sum_i B_{ij}$ as the weight for the j^{th} degree of freedom is an attempt to maximize the diagonal entries in B_{m_a} and ultimately improve the conditioning of the Z^tAZ system. An alternative weighting that seemed to give acceptable results was $\max_i B_{ij}$. We found that additionally limiting the independent degrees of freedom to *only virtual* degrees of freedom resulted in a vastly more efficient boundary smoother in our multigrid algorithm; see §4.3.

Now, given a degree of freedom weighting scheme like above, one may "freeze" the independent set once all remaining eligible degrees of freedom (those not adjacent to previously selected independent degrees of freedom) have a weight below some threshold. Alternatively, one may freeze the independent set once all the subsequently induced aggregate constraints (given the current set of independent degrees of freedom and some grid-cell-to-independent-degree-of-freedom association heuristic) satisfy some geometric bound. For example, one may terminate the further selection of independent degrees of freedom once the current set of independent degrees of freedom induces a set of aggregate constraints which each lie within a $4 \times 4 \times 4$ block of grid cells centered on the corresponding independent degree of freedom.

Finally, to minimize the geometric extent of the aggregate constraints, we associate an otherwise unassociated single-wide constraint to the geometrically closest independent degree of freedom, breaking ties by preferring higher-weighted degrees of freedom.

Algorithm 4.1 outlines an example implementation of the constraint aggregation algorithm described above. We followed this specific implementation of the constraint aggregation algorithm for the numerical examples given in §4.4.3. In this implementation, we select an independent set of virtual degrees of freedom prioritized by the sum of their associated coefficients over all single-wide constraints; and we terminate the further selection of independent degrees of freedom once all boundary grid cells are within some $4 \times 4 \times 4$ block of grid cells centered on an independent degree of freedom (Figure 4.8 explains this termination condition graphically). Together with the rule associating single-wide constraints to the geometrically closest independent degree of freedom, this termination condition ensures that all aggregate constraints fit within a $4 \times 4 \times 4$ block of grid cells centered on an independent degree of freedom, thus limiting the geometric extent of an aggregate constraint.

Algorithm 4.1 Constraint aggregration algorithm for embedded Dirichlet discretizations.

1: Reorder the degrees of freedom such that virtual degrees of freedom (VDOFs) are enumerated first and $w_1 > w_2 > \cdots$, where $w_j = \sum_i B_{ij}$ for VDOF j and B_{ij} is as in (4.12).
2: let $\mathcal{I} \leftarrow \emptyset$ {$\mathcal{I}$ denotes the set of independent degrees of freedom (IDOFs)}
3: {only iterate over VDOFs}
4: **for** $j = 1, 2, \ldots$ **do**
5: {Use an acceleration structure (e.g., an explicit set or bit set data structure) to make the following query efficient.}
6: **if** VDOF j is adjacent to some IDOF in $\mathcal{I}$ **then**
7: **continue**
8: **end if**
9: $\mathcal{I} \leftarrow \mathcal{I} \sqcup \{j\}$ {add VDOF j to the set of IDOFs}
10: {Use an acceleration structure (e.g., an associative array data structure) to make the following query efficient.}
11: **if** each boundary grid cell is within some $4 \times 4 \times 4$ block of grid cells centered on an IDOF in $\mathcal{I}$ (see Figure 4.8) **then**
12: **break**
13: **end if**
14: **end for**
15: Associate each boundary grid cell to the geometrically closest IDOF in $\mathcal{I}$, breaking ties by preferring IDOFs with higher weights (w_j). Let C_j denote the set of boundary grid cells associated to IDOF j.
16: **for all** $j \in \mathcal{I}$ **do**
17: Sum the single-wide constraint equations associated with the boundary grid cells in C_j to form a new aggregate constraint equation.
18: **end for**

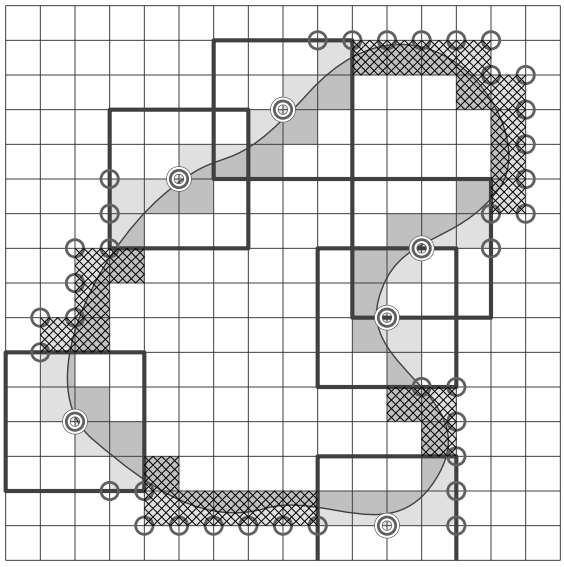


Figure 4.8: A graphical representation (in 2 dimensions) of a plausible state of Algorithm 4.1 after the selection of 6 independent degrees of freedom (highlighted). Some degrees of freedom have been removed to indicate their ineligibility as subsequently selected independent degrees of freedom: material degrees of freedom, by definition of Algorithm 4.1, are never selected as independent degrees of freedom (this vastly improved the performance of our boundary smoother in our multigrid algorithm; see §4.3); and those virtual degrees of freedom adjacent to one of the 6 previously selected independent degrees can not now be selected as independent degrees of freedom, simply by the definition of independence. Further, we distinguish between *covered* boundary grid cells, which lie within some 4×4 block of cells (shown as the dark gray outlined squares) around an independent degree of freedom; and the remaining *uncovered* boundary grid cells (denoted by cross-hatching). Once all boundary grid cells are covered, Algorithm 4.1 terminates further selection of independent degrees of freedom.

We conclude this section with some remarks regarding the discrete Lagrange multiplier space Λ^h. Generally speaking, using a richer discrete Lagrange multiplier space (one that better approximates $H^{-1/2}(\partial\Omega)$) results in a smaller error in the approximate solution u^h. Within the context of single-wide constraint aggregation, roughly speaking, one can increase the richness of Λ_a^h (the discrete Lagrange multiplier space associated with the aggregate constraints) by choosing more independent degrees of freedom. In some sense, then, the discrete Lagrange multiplier space Λ_2^h associated with the double-wide constraints represents the richest possible discrete Lagrange multiplier space one may obtain within this constraint aggregation framework, as its set of independent degrees of freedom is maximal. However, as shown in Appendix C, use of double-wide constraints leads to a relatively poorly conditioned $Z^t AZ$ system in 3 dimensions, and this behavior is characteristic of selecting too many independent degrees of freedom, some of which may be poorly supported and lead to poor

conditioning. We feel that our criterion in Algorithm 4.1 to terminate further selection of independent degrees of freedom strikes a balance between maintaining second order accuracy and ensuring reasonable conditioning in the Z^tAZ system.

In addition to the relationship among the richness of Λ^h, the error in the approximate solution u^h, and (for Λ^h_a in particular) the conditioning of the Z^tAZ system, it is also necessary, in order to obtain optimal convergence rates, for Λ^h and the approximation space to $H^1(\Omega)$, V^h, to satisfy an inf-sup stability condition uniformly in grid resolution [Pit79]. This ultimately has the effect of limiting the richness of Λ^h. Fortunately, based primarily on numerical evidence (see, for example, [JSC06] and [MDH07]), it is generally accepted that the pairing (V^h, Λ^h_1) satisfies an inf-sup stability condition, where we use the discrete Lagrange multiplier space Λ^h_1 associated with the single-wide constraints. More explicitly, we assume the existence of $\gamma_0, h_0 > 0$ such that, for all $h \in (0, h_0]$,

$$\inf_{\mu^h \in \Lambda^h_1} \sup_{v^h \in V^h} \alpha(\mu^h, v^h) \geq \gamma_0,$$

where $\alpha \colon H^{-1/2}(\partial\Omega) \times H^1(\Omega) \to \mathbb{R}$ is defined as

$$\alpha(\mu^h, v^h) := \frac{(\mu^h, Tv^h)_{\partial\Omega}}{\|\mu^h\|_{-1/2,\partial\Omega} \|v^h\|_{1,\Omega}}$$

and $T \colon H^1(\Omega) \to L^2(\partial\Omega)$ is the trace operator on Ω. Now if Λ^h_a is the discrete Lagrange multiplier space associated with any set of aggregate constraints, then Λ^h_a is a *subspace* of Λ^h_1, hence

$$\inf_{\mu^h \in \Lambda^h_a} \sup_{v^h \in V^h} \alpha(\mu^h, v^h) \geq \inf_{\mu^h \in \Lambda^h_1} \sup_{v^h \in V^h} \alpha(\mu^h, v^h) \geq \gamma_0,$$

and we see that (V^h, Λ^h_a) satisfies an inf-sup stability condition as well. (The same argument is used in [BBZ10] to show that, specifically, (V^h, Λ^h_2) is inf-sup stable.) Generally speaking, if (V^h, Λ^h) satisfies an inf-sup stability condition, then pairing V^h with any coarsening (i.e., subspace) of Λ^h will be inf-sup stable as well.

4.2.4 Embedded Interface

To handle the full interface problem (II.1, 4.1b, 4.1c), we combine our treatments of Neumann and Dirichlet boundary conditions in a straightforward way. We consider the equivalent minimization form of the problem (II.1, 4.1b, 4.1c):

$$\text{over all } u \in V := \{u : u^{\pm} \in H^1(\Omega^{\pm})\}, \text{ minimize}$$

$$E(u) := e(u) - (f,u)_\Omega - (b,\overline{u})_\Gamma := \int_{\Omega^+ \sqcup \Omega^-} \frac{1}{2} \nabla u \cdot \beta \nabla u d\mathbf{x} - \int_\Omega f u d\mathbf{x} - \int_\Gamma b\overline{u} d\mathbf{S}(\mathbf{x}) \quad (4.20)$$

$$\text{such that} \quad ([u], \mu)_\Gamma = (a, \mu)_\Gamma \quad \forall \mu \in H^{-1/2}(\Gamma). \quad (4.21)$$

Here $\overline{u}(\mathbf{x})|_\Gamma = (u^+ + u^-)/2$. As before, we define discretizations of V and $H^{-1/2}(\Gamma)$ and then construct the resulting discrete saddle point problem. To define $V^h \subset V$, we separately discretize $H^1(\Omega^+)$ and $H^1(\Omega^-)$ using the same virtual node representation used to discretize domain problems, employing the duplicated grid described in §4.2.1 and depicted in Figure 4.3. This discretization yields the block diagonal stiffness matrix for the interface problem,

$$A = \begin{pmatrix} A^+ & 0 \\ 0 & A^- \end{pmatrix}, \tag{4.22}$$

where A^+ is the stiffness matrix associated with the pure Neumann problem on Ω^+ and A^- is the stiffness matrix associated with the pure Neumann problem on Ω^-, as described in §4.2.2.

As for the Dirichlet problem, we first discretize the continuous constraint equations (4.21) via Λ_1^h into single-wide constraint equations,

$$\int_{c_k \cap \Gamma} \left[u^h\right] d\mathbf{S}(\mathbf{x}) = \int_{c_k \cap \Gamma} a d\mathbf{S}(\mathbf{x}), \tag{4.23}$$

and then aggregate these single-wide constraints (4.23), as described in §4.2.3.2:

$$\sum_{c_k \in C_{d_i}} \int_{c_k \cap \Gamma} \left[u^h\right] d\mathbf{S}(\mathbf{x}) = \sum_{c_k \in C_{d_i}} \int_{c_k \cap \Gamma} a d\mathbf{S}(\mathbf{x}). \tag{4.24}$$

Note that we described the constraint aggregation procedure in §4.2.3.2 within specifically in the context of Dirichlet constraints, but aggregating single-wide interface constraints is entirely analogous and straightforward. Regarding the specific implementation in Algorithm 4.1, one would use the weights $w_j = |\sum_i B_{ij}|$ (note the addition of the absolute value) to account for negative single-wide constraint coefficients on interior degrees of freedom.

Using the aggregate constraints in (4.24) results in the block interface constraint matrix $B = (B^+|-B^-)$, where B^+, B^- are, respectively, the constraint matrices associated with the embedded Dirichlet problems on the exterior and interior of the interface. In other words,

$$B_{ij} = \sigma_j \sum_{c_k \in C_{d_i}} \int_{c_k \cap \Gamma} N_j d\mathbf{S}(\mathbf{x}), \tag{4.25}$$

where $\sigma_j := +1$ if the j^{th} degree of freedom is associated with $u^{h,+}$ and $\sigma_j := -1$ if the j^{th} degree of freedom is associated with $u^{h,-}$. These discretization choices give the saddle point problem

$$\begin{pmatrix} A^+ & 0 & (B^+)^t \\ 0 & A^- & (-B^-)^t \\ B^+ & -B^- & 0 \end{pmatrix} \begin{pmatrix} \vec{u}^+ \\ \vec{u}^- \\ \vec{\lambda} \end{pmatrix} = \begin{pmatrix} \vec{f}^+ \\ \vec{f}^- \\ \vec{a} \end{pmatrix}, \tag{4.26}$$

where $\vec{u}^+$ contains the degrees of freedom associated with the exterior discretization and $\vec{u}^-$ contains the degrees of freedom associated with the interior discretization. We once again solve this saddle point system using the null space method described in §4.2.3.1 by

ordering the independent degrees of freedom first to obtain a diagonal B_{m_a}. Observe that we may restrict independent degrees of freedom to only virtual degrees of freedom, as every material degree of freedom has a geometrically co-located virtual degree of freedom that is indistiguishable as far as adjacency and weight (up to a sign change) is concerned. We have found that such a restriction results in a better-conditioned system. Contrast this observation with the Dirichlet case, where each material degree of freedom does *not* have an equivalent (as far as the constraint system is concerned) virtual degree of freedom, and hence the decision to allow or disallow the selection of material degrees of freedom as independent degrees of freedom has a much bigger impact on the final set of aggregate constraints.

4.2.4.1 Discontinuity removal

In general, our proposed method requires the solution of the symmetric positive definite system Z^tAZ. However, if the coefficient β is smooth, the IIM and similar methods achieve uniform second order accuracy without altering the standard Poisson finite difference stencil (the 5-point stencil in 2 dimensions or the 7-point stencil in 3 dimensions). In this section, we demonstrate how the virtual node framework similarly allows the use of the standard Poisson stencil when β is smooth.

Suppose $d(\mathbf{x}) \in V$ is constructed to satisfy the jump conditions (4.1b, 4.1c) and $u(\mathbf{x})$ is the exact solution. Then since $[\beta] = 0$, the difference $w(\mathbf{x}) := u(\mathbf{x}) - d(\mathbf{x})$ satisfies $\beta\,[\nabla w \cdot \hat{\mathbf{n}}] = [\beta \nabla w \cdot \hat{\mathbf{n}}] = 0$ and $[w] = 0$. Since w satisfies homogeneous jump conditions $[\nabla w \cdot \hat{\mathbf{n}}] = 0$ and $[w] = 0$, we do not require virtual degrees of freedom to capture any discontinuities across Γ. In this manner, solving for w presents an appealing alternative as the presence of virtual nodes no longer adversely affects the subsequent linear algebra problem. Therefore, when $[\beta] = 0$ we recover an approximation to (4.1b, 4.1c) by separately discretizing w and d and then setting $u = w + d$.

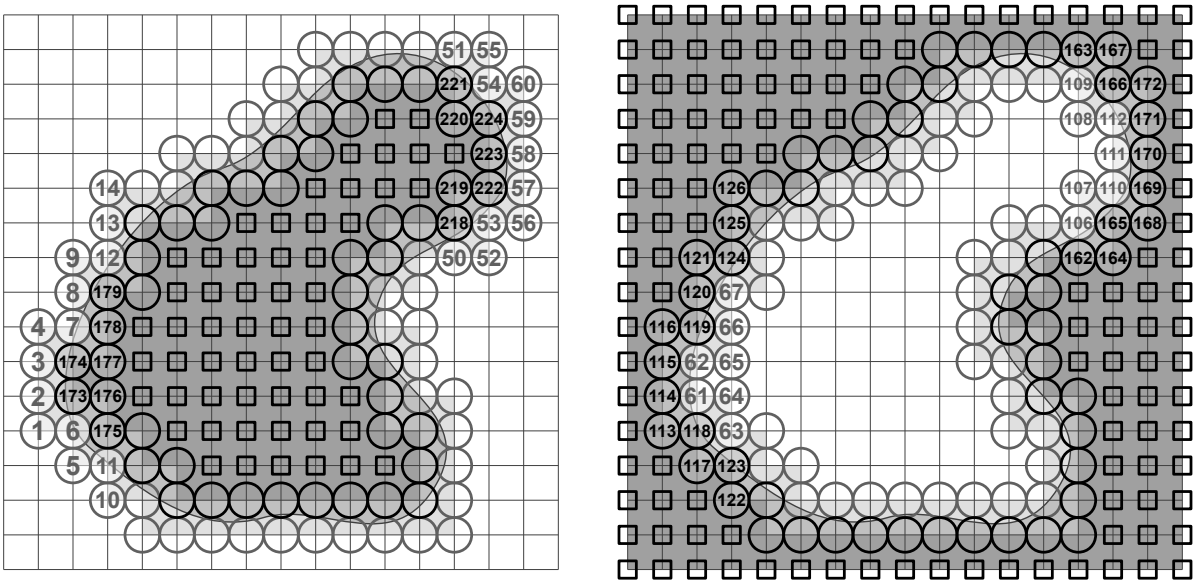
We discretize w over the unduplicated grid $\mathcal{G}^h$ using $H^1(\Omega)$ Cartesian piecewise trilinear elements. Consequently, if the grid $\mathcal{G}^h$ contains r material degrees of freedom, then $\vec{w} \in \mathbb{R}^r$ contains the coefficients in terms of the trilinear basis. We discretize u and d using the full virtual node basis V^h as they possess lower regularity across Γ. With these choices, we can represent the coefficient vector $\vec{u} \in \mathbb{R}^n$ $(n > r)$ of the approximate solution u^h in the basis of V^h as $\vec{u} = \vec{d} + T\vec{w}$, where the matrix $T \in \mathbb{R}^{n \times r}$ is an embedding of the trilinear basis into the virtual node basis. We define this transformation by a simple identification of virtual and material nodes, as a function $v^h \in V^h$ satisfies homogeneous jump conditions if and only if the value of the function v^h at a virtual node equals its value at the geometrically co-located material node. Thus, T maps the value at a given vertex in the unduplicated grid to each of its copies, material or virtual, in the duplicated grid. To be a little more explicit, assume that we order the degrees of freedom such that

$$\vec{u} = (u_1, u_2, \ldots, u_s, u_{s+1}, u_{s+2}, \ldots, u_{2s}, u_{2s+1}, \ldots, u_n)^t.$$

Here, $\{u_k\}_{k=1}^{s}$ represent the $s := n - r$ coefficients of the virtual degrees of freedom; $\{u_{s+k}\}_{k=1}^{s}$ represent the coefficients of the material degrees of freedom respectively co-located with $\{u_k\}_{k=1}^{s}$; and the remaining coefficients $\{u_k\}_{k=2s+1}^{n}$ correspond to degrees of freedom lying

outside any interfacial grid cells. See Figure 4.9 for an illustration of this ordering. Then T would take the form

$$T = \begin{pmatrix} I_s & 0 \\ I_s & 0 \\ 0 & I_{n-2s} \end{pmatrix}. \tag{4.27}$$



(a) Interior discretization

(b) Exterior discretization

Figure 4.9: Example enumeration of the interfacial degrees of freedom (circled) such that T has the representation (4.27). Only the indices of a few select interfacial degrees of freedom are shown. Here, we enumerate the $s = 112$ virtual degrees of freedom lexicographically, beginning with the interior discretization. The interior discretization has 60 virtual degrees of freedom (indexed 1 to 60) and 52 interfacial material degrees of freedom (indexed 173 to 224); likewise, the exterior discretization has 52 virtual degrees of freedom (indexed 61 to 112) and 60 interfacial material degrees of freedom (indexed 113 to 172). Notice how the the index to an interfacial material degree of freedom is offset from the index of its co-located virtual degree of freedom by exactly $s = 112$. The remaining non-interfacial degrees of freedom (squared) are enumerated starting with index $2s + 1 = 225$.

Regardless of the ordering of the degrees of freedom, each column of T corresponds to a material node in the grid and each row of T corresponds to either a material node or a virtual node. The column of T corresponding to material node j has a 1 in the row corresponding to material node j; a 1 in the row corresponding to j's geometrically co-located virtual node, if it exists (as in one of the first s columns in (4.27) above); and zeros everywhere else.

Determining $\vec{w}$ now proceeds in a manner analogous to the null space method used to solve (4.14): we wish to minimize the energy over all vectors of the form $\vec{u} = \vec{d} + T\vec{w}$. For the sake of discussion, suppose we discretize the energy (4.20) using the Cartesian trilinear

representation everywhere in the domain. Then substituting the expression $\vec{u} = \vec{d} + T\vec{w}$ into the energy (4.20) gives

$$E^h(\vec{u}) := \frac{1}{2}\vec{u}^t A\vec{u} - \vec{f}^t\vec{u} = \frac{1}{2}\vec{w}^t T^t AT\vec{w} - \vec{f}^t T\vec{w} + \vec{w}^t T^t A\vec{d} + \frac{1}{2}\vec{d}^t A\vec{d} - \vec{f}^t\vec{d},$$

which, in turn, implicitly defines an energy over only the material degrees of freedom $\vec{w} \in \mathbb{R}^r$. Differentiation with respect to w_i thus leads to the linear system

$$T^t AT\vec{w} = T^t(\vec{f} - A\vec{d}), \quad \vec{u} = \vec{d} + T\vec{w}.$$

It is not hard to show that the matrix $T^t AT$ is a straightforward, trilinear discretization over the material degrees of freedom, i.e., a 27-point second order approximation to the (variable coefficient) Laplacian. Thus, we may replace the $T^t AT$ operator with the standard 7-point Poisson stencil Δ^h_β, only introducing a second order deviation in $\vec{w}$:

$$\Delta^h_\beta\vec{w} = T^t(\vec{f} - A\vec{d}), \quad \vec{u} = \vec{d} + T\vec{w}. \tag{4.28}$$

This approach allows the application of efficient black-box solvers for Δ^h_β, and the discontinuity along the interface only enters into the right-hand side of (4.28).

We now discuss the approximation of d, the particular solution satisfying the jump conditions (4.1b, 4.1c). Observe that, without loss of generality, we may assume that d is supported only near the interface, as the jump constraints are localized to the interface. Further, we may assume that d vanishes entirely on, say, the exterior region Ω^+, as the jump constraints only involve *differences* between exterior and interior values. This latter assumption allows us to express the jump constraints on d as direct constraints on d^-:

$$-d^- = [d] = a, \quad -\beta\nabla d^- \cdot \hat{\mathbf{n}} = \beta\,[\nabla d \cdot \hat{\mathbf{n}}] = b.$$

The corresponding discretized single-wide constraints on d^h, the V^h-approximation to d, are thus

$$\int_{c_i\cap\Gamma} d^{h,-} d\mathbf{S}(\mathbf{x}) = -\int_{c_i\cap\Gamma} a d\mathbf{S}(\mathbf{x}), \quad \int_{c_i\cap\Gamma} \beta\nabla d^{h,-}\cdot\hat{\mathbf{n}} d\mathbf{S}(\mathbf{x}) = -\int_{c_i\cap\Gamma} b d\mathbf{S}(\mathbf{x})$$

for each grid cell $c_i \in \mathcal{C}^h_\Gamma$ intersecting the interface Γ; and $d^{h,+} \equiv 0$. This gives a sparse linear system for the coefficient vector $\vec{d}$ of d^h where only interior interfacial degrees of freedom participate:

$$\sum_{j\in\mathcal{N}^{h,-}_{c_i}} \left(\int_{c_i\cap\Gamma} N_j d\mathbf{S}(\mathbf{x})\right) d_j = -\int_{c_i\cap\Gamma} a d\mathbf{S}(\mathbf{x}); \tag{4.29}$$

$$\sum_{j\in\mathcal{N}^{h,-}_{c_i}} \left(\int_{c_i\cap\Gamma} \beta\nabla N_j\cdot\hat{\mathbf{n}} d\mathbf{S}(\mathbf{x})\right) d_j = -\int_{c_i\cap\Gamma} b d\mathbf{S}(\mathbf{x}); \tag{4.30}$$

where $\mathcal{N}^{h,-}_{c_i}$ denotes the indices of the 8 interior degrees of freedom geometrically located at the corners of cell c_i. This system has $2m$ rows, where $m = \left|\mathcal{C}^h_\Gamma\right|$ is the number of interfacial

grid cells; and it has one column for each interior interfacial degree of freedom. Thus, unfortunately, this system will not only be asymmetric, but will generally be overdetermined as well. Hence, one should take some care when computing an approximate solution.

Algorithm 4.2 gives one approach to constructing $\vec{d}$ which we found works well. The algorithm locally constructs a trilinear function v which approximately satisfies the constraints (4.29, 4.30) within a $3 \times 3 \times 3$-cell neighborhood centered on an interfacial grid cell $c_i \in \mathcal{C}_\Gamma^h$. We then evaluate v at the grid vertices of c_i to obtain values for the corresponding entries in $\vec{d}$. This procedure may give an interfacial degree of freedom multiple values, from multiple neighboring local construction; we average these values together, as explained below.

We alert the reader to two subtle but important details of Algorithm 4.2. First, most, if not all, of these local constructions amount to a least-squares solution to a small overdetermined system of linear equations. In order to achieve second order convergence in u, we found it necessary to scale the constraints (4.30) on $\nabla d^{h,-}$ by $h^{1+\gamma}$ (for some γ between 0 and 1), which places more emphasis on satisfying the constraints on $d^{h,-}$ than on satisfying the constraints on $\nabla d^{h,-}$ in the least-squares solves. We found $\gamma = 1/3$ gave the best convergence rate for Example 4.4.5 over the range of tested resolutions. We suggest further research is necessary to determine the optimal scaling of the $\nabla d^{h,-}$-constraints in general, both theoretically and empirically.

Second, as mentioned above, multiple neighboring local construction may yield a value for $\vec{d}$ at a given interfacial degree of freedom; indeed, the number of such local constructions around a degree of freedom equals the number of incident interfacial grid cells. We compute a final value for $\vec{d}$ at this degree of freedom by taking a *weighted* average of the values yielded by the various local constructions, with weights equal to the surface area of Γ within the interface grid cell around which the local construction is based. Other weightings of the various local construction contributions could very well give equal or better results.

Note that the computational cost of computing $\vec{d}$ in the above fashion is proportional to the number of interfacial degrees of freedom, hence contributes negligibly to the overall cost of computing $\vec{u}$.

4.3 Multigrid

One of our primary contributions is a collection of geometric multigrid algorithms to solve the linear systems arising from the discretizations of domain and interface Poisson problems described in §4.2. Multigrid methods are well-known to be more efficient than standard iterative Krylov solvers (such as conjugate gradient), as a multigrid solver can often operate in $\mathcal{O}(\#$ of degrees of freedom) time (or nearly so). Additionally, our multigrid solvers are geometric in nature, hence allow implementations with low memory requirements and scalable parallelizability.

We will begin the exposition with a discussion of the grid hierarchy, followed by details regarding the smoothing and transfer operators. Since our multigrid algorithms for the Neumann, Dirichlet, and interface discretizations share the same general principles, we will

Algorithm 4.2 Construct an approximate d satisfying (4.1b, 4.1c).

{I and J below denote multi-indices, i.e., triples of linear indices, over the unduplicated grid $\mathcal{G}^h$.}
$\vec{c} \leftarrow \vec{0}$
$w_J \leftarrow 0$ for each interior grid vertex J incident to an interfacial grid cell {the weight sum for degree of freedom J}
for all $c_I \in \mathcal{C}^h_\Gamma$ **do**
  let $S := \left\{c_J \in \mathcal{C}^h_\Gamma : \|I - J\|_\infty \leq 1\right\}$
  assert($|S| \leq 27$ and $c_I \in S$)
  Construct a trilinear function v (i.e., solve for 8 coefficients) satisfying the constraints (4.29, 4.30) on $d^{h,-}$ and $\nabla d^{h,-}$ defined over the cells in S (2 constraints per cell). If $2\,|S| < 8$, choose v to have minimum 2-norm (for some appropriate 2-norm on the trilinear coefficients); if $2\,|S| > 8$, choose v to minimize the 2-norm of the residual of the constraint equations after scaling the $\nabla d^{h,-}$-constraint equations by $h^{1+\gamma}$.
  let $w := \int_{c_I \cap \Gamma} dS(\mathbf{x})$ {the local weight for the degrees of freedom at the corners of c_I}
  for $i = 1, \ldots, 8$ **do**
    let J denote the index of the i^{th} grid vertex incident to c_I (say, lexicographically)
    $d^-_J \leftarrow d^-_J + w \cdot v(\mathbf{x}_J)$ {$\mathbf{x}_J$ denotes the spacial coordinates of grid vertex J}
    $w_J \leftarrow w_J + w$
  end for
end for
for all interfacial grid vertex indices J **do**
  $d^-_J \leftarrow d^-_J / w_J$ {average the multiple contributions to the value of d^-_J}
end for

discuss our multigrid algorithms within the context of all three discretizations simultaneously, noting important differences as they arise. We emphasize that the constraint aggregation described in §4.2.3.2 plays an integral role in our multigrid algorithms for Dirichlet and interface problems, as we base our boundary/interface-local smoother on the $Z^t A Z$ symmetric positive definite system.

We note that we follow standard geometric multigrid principles away from embedded features, and thus our primary focus is the nontrivial treatment of the multigrid components around the embedded features of the discretization. In order to minimize peripheral complexity, we assume that β (for domain problems) or β^+, β^- (interface problems) are constant.

4.3.1 Discretization

As is characteristic of geometric multigrid methods, we discretize our problem (as desribed in §4.2) within each of a hierarchy of Cartesian grids $\mathcal{G}^h, \mathcal{G}^{2h}, \ldots$, with the cell resolutions between successive grids in the hierarchy differing by a factor of 2. Thus, with each level in the hierarchy, we associate

- Cartesian grids $\mathcal{G}^h, \mathcal{G}^{2h}, \ldots$, with the domain embedded as described in §4.2.1;
- Poisson operators $A^h, A^{2h}, \ldots$ (4.6)/(4.22); and
- solution and right-hand side vectors $\vec{u}^h, \vec{u}^{2h}, \ldots$ and $\vec{f}^h, \vec{f}^{2h}, \ldots$.

For Dirichlet and interface problems, we also associate the aggregated constraint matrices $B^h, B^{2h}, \ldots$ (4.19)/(4.25). To simplify the discretization, we assume the constraint aggregation on a given level is independent of the aggregation on other levels. That is, we make no attempt to ensure coherency or geometric consistency between the sets of constraints on successive levels. However, as a result, the constructions of the multigrid components near embedded features require special consideration, as will be explained below. Note that the presence of the aggregate constraints on each level allows one to easily form the Z^tAZ system as described in §4.2.3, and, as we will see, it is this system that we base our smoothing operator on.

We emphasize that, in spirit, for Dirichlet and interface problems, we are applying multigrid to the saddle point system (4.15) or (4.26). In theory, then, we should additionally associate a Lagrange multiplier vector $\vec{\lambda}^h, \vec{\lambda}^{2h}, \ldots$ at each level of the hierarchy. However, we have designed our multigrid algorithms in such a way that, in practice, it is unnecessary (and, indeed, impractical) to explicitly operate on and store these Lagrange multiplier vectors. Instead, we ensure the (aggregate) constraint equations at each level are always satisfied, hence there is no need to restrict $\vec{\lambda}$-residuals or prolongate $\vec{\lambda}$-corrections; and, to repeat, our smoothing operator is based on the Z^tAZ system, which means we can smooth the error in $\vec{u}$ without making any explicit reference to $\vec{\lambda}$.

4.3.2 Smoothing Operator

In describing our smoothing operator, it will be useful to distinguish between non-boundary/non-interfacial and boundary/intefacial degrees of freedom. The former are squared and the latter are circled in Figures 4.1 and 4.3. Non-boundary/Non-interfacial degrees of freedom possess the standard 7-point stencil, as depicted in Figure 4.5, even within the Z^tAZ systems arising from Dirichlet and interface problems. Thus, on these degrees of freedom, one may apply standard smoothers appropriate for symmetric positive definite systems, such as weighted Jacobi, Gauss-Seidel, Red-Black Gauss-Seidel, etc.

Although the boundary discretization for Neumann problems produces a denser stencil than the standard 7-point stencil, it is still locally semidefinite, hence one may still apply standard smoothers to these degrees of freedom as well. However, the discretization for Dirichlet problems near the domain boundary and for interface problems near the interface is indefinite (recall that we are, in spirit, operating on the saddle point system (4.15) or (4.26)), so the standard smoothers mentioned above are not options. Alternative smoothers *might* work, such as Kaczmarz or box smoothers, but they will generally be slower, and they require the use of the Lagrange multiplier $\vec{\lambda}$, which we would like to avoid. We choose instead to apply a standard smoother, such as Gauss-Seidel, on the symmetric positive definite Z^tAZ

system (which coincides with the Poisson operator away from embedded features). Note that the Z^tAZ system operates on all the degrees of freedom *except* the independent degrees of freedom. In effect, we have *eliminated* the independent degrees of freedom from the system, such that each update of a boundary/interfacial non-independent degree of freedom in the Z^tAZ system during, say, a Gauss-Seidel step induces an update of one or more (eliminated) independent degrees of freedom to ensure the solution remains in the null space of the constraint system. Thus, if our initial guess at the finest level satisfies the constraints (e.g., $\vec{c} = \begin{pmatrix}(B_{m_a}^{-1}\vec{p})^t & 0\end{pmatrix}^t$), then future corrections via smoothing will keep the approximation in the solution space of the constraint system associated with that level of the hierarchy.

As we will see, to avoid complexity, we do not use specialized transfer operators near embedded features, as is done in [AL02, AC04, AC05, CS08, WL04]. However, the incoherency between the feature embeddings and constraint aggregations within successive discretization levels, as well as the absence of the $\vec{\lambda}$ vectors, precludes the successful use of standard transfer operators near these embedded features. We address this by devoting extra smoothing effort around embedded features to drive the corresponding residuals close to zero and propagate non-boundary/non-interfacial corrections toward embedded features. Thus, a full smoothing sweep will generally consist of a few boundary/interface-local Gauss-Seidel sweeps, followed by a single Gauss-Seidel sweep over all degrees of freedom, and ending with a few more boundary/interface-local Gauss-Seidel sweeps. One can use numerical experimentation to determine exactly how many boundary/interface-local sweeps are necessary, and our experiments indicate that Neumann and Dirichlet problems need only a half dozen or fewer additional boundary/interface-local smoothing sweeps on either side of the smoothing sweep over all degrees of freedom; interface problems seem to need somewhat more additional boundary/interface-local smoothing sweeps. Fortunately, for large resolutions, the single smoothing sweep over all degrees of freedom will dominate the work expended on these boundary/interface-local smoothing sweeps. For complete results, we refer the reader to §4.4.6.

4.3.3 Transfer Operators

Our multigrid algorithms use standard prolongation and restriction operators away from embedded features. We prolongate a coarse-grid correction $\vec{u}^{2h}$ to the fine-grid solution $\vec{u}^h$ via trilinear interpolation: $\vec{u}^h \leftarrow \vec{u}^h + P\vec{u}^{2h}$. We restrict a fine-grid residual in $\vec{u}^h$ to the coarse-grid right-hand side via the scaled adjoint operator $R := 8P^t$.

Often, in the presence of embedded features, one considers introducing specialized transfer operators near these features [AL02, AC04, AC05, CS08, WL04]. As stated above, we have opted to avoid this complexity and the attendant necessary additional storage. However, we cannot rely on the standard transfer operators by themselves to correctly restrict fine-grid residuals and prolongate coarse-grid corrections near embedded features. Thus, we expend extra smoothing effort to ensure the fine-grid residuals near embedded features are close to zero prior to restriction. Indeed, for Dirichlet and interface problems, we restrict identically zero residuals *from all* fine-grid equations corresponding to boundary/interfacial degrees of freedom, which correspond to precisely those rows in the saddle point system in-

volving $\vec{\lambda}$. We additionally only restrict *to a strict subset* of the coarse-grid equations (e.g., only those corresponding to material degrees of freedom, or only those corresponding to non-boundary/non-interfacial degrees of freedom). Further, we prolongate zero values *from* virtual degrees of freedom in the coarse-grid correction, and again expend extra smoothing effort to propagate toward embedded features the more reliable coarse-grid corrections away from the embedded features.

For Dirichlet and interface problems, recall again that, in spirit, we are applying our multigrid algorithms on the saddle point systems, hence obstensibly we should be restricting residuals from the constraint equations as well. However, by smoothing via the Z^tAZ system and (implicitly) propagating all updates to the independent degrees of freedom, we ensure the constraint equations are always satisfied exactly, i.e., have zero residual.

4.3.4 Details

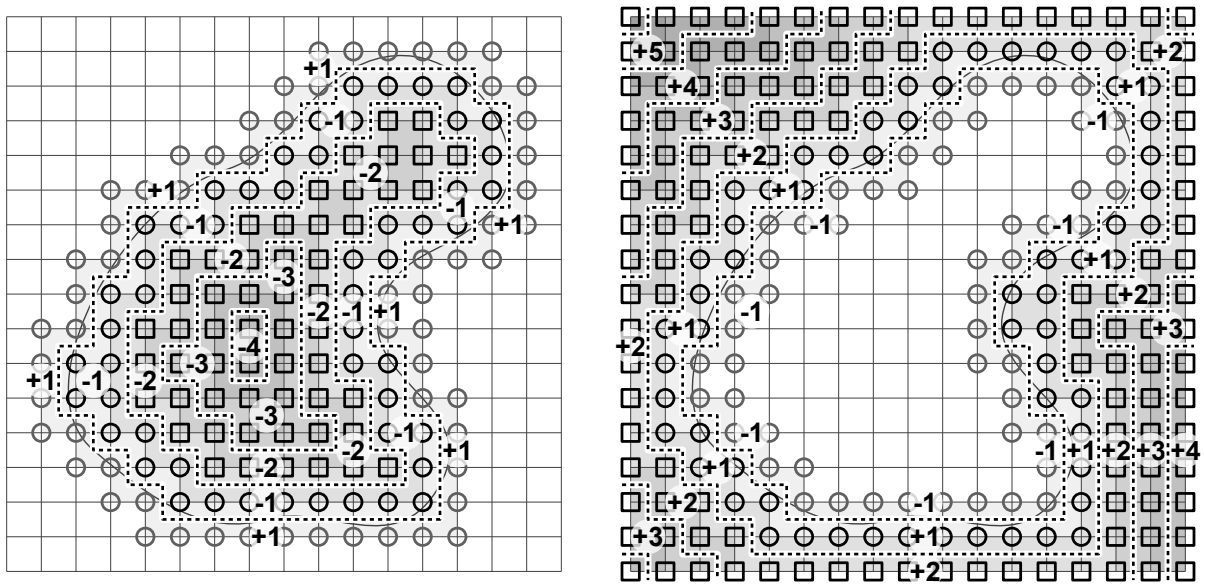
The preceding sections gave an overview of the general strategy for our multigrid algorithms, and here we only provide some additional details of our implementation of these ideas. Our primary goal is not necessarily to develop the most efficient implementation, but rather to provide a simple reference implementation which can provide a baseline for future research.

We use lexicographically ordered Gauss-Seidel iterations in all phases of our smoothers. The empirical convergence rates we obtain in our numerical examples in §4.4.6 indicate that the Gauss-Seidel method is a sufficiently good smoother away from embedded features. Technically, the pre-restriction and post-prolongation smoothing sweeps serve difference purposes, so one could tailor the details of each to perform optimally for their respective purpose. For simplicity, however, we use identical pre-restriction and post-prolongation smoothing sweeps. Furthermore, we always buttress the Gauss-Seidel sweep over all degrees of freedom with equal numbers of boundary/interface-local Gauss-Seidel sweeps on either side. We refer to this number at the finest level as the *number of boundary smoothing sweeps* (NBSS; Neumann, Dirichlet) and *number of interface smoothing sweeps* (NISS; interface). At each successively coarser level, we increase the number of boundary or interface smoothing sweeps by a factor of 2 (see Algorithms 4.3 and 4.4). Since the number of degrees of freedom in a neighborhood of an embedded feature scales as N^2 for a grid resolution of, say, $N \times N \times N$, this increase in the number of boundary or interface smoothing sweeps at coarser levels does not change the overall complexity of our algorithms. Furthermore, we found it significantly improved our v-cycle convergence rates with negligible additional cost per v-cycle.

For the boundary/interface-local Gauss-Seidel sweeps, we iterate over all degrees of freedom within a fixed L^∞-grid-distance of a boundary/interfacial degree of freedom. See Figure 4.10 for an example assignment to all degrees of freedom of the (discrete) signed grid-distance to the embedded boundary or embedded interface. We use the terms *boundary smoothing region width* (BSRW; Neumann, Dirichlet) and *interface smoothing region width* (ISRW; interface) to refer to this distance defining the boundary/interface-local region we apply extra Gauss-Seidel sweeps to. Thus, a BSRW/ISRW of 1 refers to all boundary/interfacial degrees of freedom, while a BSRW/ISRW of 2 refers to all degrees of freedom

within an L^∞-grid-distance of 1 from a boundary/interfacial degree of freedom.

Within an interface-local Gauss-Seidel sweep, we found it necessary to relax co-located interior and exterior degrees of freedom consecutively. In other words, co-located pairs of degrees of freedom resulting from a single grid vertex duplication should be relaxed one after the other. To be clear, an interface-local Gauss-Seidel sweep which iterates over *all* interior degrees of freedom followed by *all* exterior degrees of freedom (or vice versa) *fails* to reduce the residuals around the interface within a reasonable number of iterations.



(a) Ω (Neumann, Dirichlet) or Ω^- (interface) discretization

(b) Ω^+ discretization

Figure 4.10: Partitioning the degrees of freedom according to their grid-distance from the embedded boundary or embedded interface.

For completeness, we provide pseudocode for a multigrid v-cycle for Neumann problems (Algorithm 4.3) and Dirichlet problems (Algorithm 4.4) (the pseudocode for interface problems would be nearly identical to that for Dirichlet problems, so we omit it). In these algorithm listings, L denotes the number of levels, with the finest level indexed as 1; and we index all variables associated with a given level with the level index (as opposed to $h, 2h, \ldots$, as we had been doing above).

4.4 Numerical Examples

We now present some numerical examples demonstrating the second order accuracy of our methods for embedded Neumann, embedded Dirichlet, and embedded interface problems, including an example utilizing discontinuity removal for an interface problem with smooth β

Algorithm 4.3 Multigrid v-cycle algorithm for Neumann problems.

initialize Poisson operators $A^1, \ldots, A^L$ at all levels as described in §4.2.2; allocate space for solution vectors $\vec{u}^1, \ldots, \vec{u}^L$ and right-hand side vectors $\vec{f}^1, \ldots, \vec{f}^L$
set $\vec{f}^1 \leftarrow \vec{f}$ from (4.6)
set $\vec{u}^1$ as some convenient initial guess satisfying any (grid-aligned) Dirichlet conditions (if present)
for $\ell = 1, \ldots, L-1$ **do**
    perform a full smoothing sweep on $A^\ell \vec{u}^\ell = \vec{f}^\ell$ {§4.3.2, with $2^{\ell-1}$NBSS boundary-local Gauss-Seidel sweeps on each side of a Gauss-Seidel sweep over all degrees of freedom}
    restrict the fine-grid residual $\vec{r}^\ell := \vec{f}^\ell - A^\ell \vec{u}^\ell$ to the coarse-grid right-hand side: $\vec{f}^{\ell+1} \leftarrow R^\ell \vec{r}^\ell$ {§4.3.3; only restrict *to* coarse-grid material equations}
    set $\vec{u}^{\ell+1} \leftarrow \vec{0}$
end for
solve $A^L \vec{u}^L = \vec{f}^L$ exactly {using a sufficient number of Gauss-Seidel iterations, for example}
for $\ell = L-1, \ldots, 1$ **do**
    prolongate the coarse-grid correction $\vec{u}^{\ell+1}$ to the fine-grid solution: $\vec{u}^\ell \leftarrow \vec{u}^\ell + P^{\ell+1}\vec{u}^{\ell+1}$ {§4.3.3; prolongate zeros at coarse-grid virtual degrees of freedom}
    perform a full smoothing sweep on $A^\ell \vec{u}^\ell = \vec{f}^\ell$ {§4.3.2, with $2^{\ell-1}$NBSS boundary-local Gauss-Seidel sweeps on each side of a Gauss-Seidel sweep over all degrees of freedom}
end for

across the interface. We will additionally present some examples demonstrating the efficiency of our geometric multigrid algorithms.

We discretized our examples on a variety of $N \times N \times N$-cell grids (up to 416^3 for Neumann, Dirichlet, and interface problems with smooth β; up to 320^3 for interface problems with discontinuous β) within the box $[-1, +1]^3$. For each example below, we give a graphic depicting the embedded boundary or interface; a few plots showing typical slices of the discrete approximation u^h, e.g., plots of $u^h(x, y, z_0)$ against (x, y) with $z = z_0$ fixed; and log-log plots of the errors in the discrete approximation $\left\|u - u^h\right\|_\infty$ and the gradient of the discrete approximation $\left\|\nabla u - \nabla u^h\right\|_\infty$ against the resolution N, which demonstrate second order convergence in u and first order convergence in ∇u. We compute $\left\|u - u^h\right\|_\infty$ as the maximum absolute difference between the analytic solution u and the discrete approximation u^h over all material degrees of freedom. We compute $\left\|\nabla u - \nabla u^h\right\|_\infty$ as the maximum L^∞-norm between ∇u and ∇u^h over, again, all material degrees of freedom. Note that, strictly speaking, ∇u^h is discontinuous across grid cell faces, and specifically around grid vertices. Thus, we evaluate ∇u^h at a grid vertex by averaging its limits when approached from each of the (up to 8) non-boundary/non-interfacial incident grid cells (using the trilinear representation of u^h within each incident grid cell). We restrict this averaging to only non-boundary/non-interfacial grid cells to ensure we use only material degrees of freedom in the evaluation of ∇u^h.

Occasionally, at higher resolutions, a degree of freedom is so poorly supported that catas-

Algorithm 4.4 Multigrid v-cycle algorithm for Dirichlet problems.

initialize Poisson operators $A^1, \ldots, A^L$ and aggregated constraint matrices $B^1, \ldots, B^L$ (and/or fundamental basis matrices $Z^1, \ldots, Z^L$) at all levels as described in §4.2.3; allocate space for solution vectors $\vec{u}^1, \ldots, \vec{u}^L$ and right-hand side vectors $\vec{f}^1, \ldots, \vec{f}^L$

set $\vec{f}^1 \leftarrow \vec{f}$ from (4.6) (without the q contribution, of course)

let $\vec{c} := \begin{pmatrix} B_{m_a}^{-1}\vec{p} \\ 0 \end{pmatrix}$ {$\vec{c}$ satisfies the embedded Dirichlet constraints}

set $\vec{f}^1 \leftarrow Z^t\left(\vec{f}^1 - A^1\vec{c}\right)$ (note: we implicitly identify the domains and codomains of Z and Z^t)

set $\vec{u}^1$ as some convenient initial guess satisfying any grid-aligned Dirichlet conditions (if present)

for $\ell = 1, \ldots, L-1$ **do**

  perform a full smoothing sweep on $(Z^\ell)^t A^\ell Z^\ell \vec{u}^\ell = \vec{f}^\ell$ {§4.3.2, with $2^{\ell-1}$NBSS boundary-local Gauss-Seidel sweeps on each side of a Gauss-Seidel sweep over all degrees of freedom; be sure to update independent degrees of freedom as necessary to maintain $\vec{u}^\ell$ in the null space of the Dirichlet constraints}

  restrict the fine-grid residual $\vec{r}^\ell := \vec{f}^\ell - (Z^\ell)^t A^\ell Z^\ell \vec{u}^\ell$ to the coarse-grid right-hand side: $\vec{f}^{\ell+1} \leftarrow R^\ell \vec{r}^\ell$ {§4.3.3; restrict zero values *from* fine-grid boundary degrees of freedom, and only restrict *to* coarse-grid non-boundary equations}

  set $\vec{u}^{\ell+1} \leftarrow \vec{0}$

end for

solve $(Z^L)^t A^L Z^L \vec{u}^L = \vec{f}^L$ exactly {using a sufficient number of Gauss-Seidel iterations, for example}

for $\ell = L-1, \ldots, 1$ **do**

  prolongate the coarse-grid correction $\vec{u}^{\ell+1}$ to the fine-grid solution: $\vec{u}^\ell \leftarrow \vec{u}^\ell + P^{\ell+1}\vec{u}^{\ell+1}$ {§4.3.3; prolongate zeros at coarse-grid virtual degrees of freedom}

  perform a full smoothing sweep on $(Z^\ell)^t A^\ell Z^\ell \vec{u}^\ell = \vec{f}^\ell$ {§4.3.2, with $2^{\ell-1}$NBSS boundary-local Gauss-Seidel sweeps on each side of a Gauss-Seidel sweep over all degrees of freedom; be sure to update independent degrees of freedom as necessary to maintain $\vec{u}^\ell$ in the null space of the embedded Dirichlet constraints}

end for

set $\vec{u}^1 \leftarrow \vec{c} + Z\vec{u}^1$ (note: here we are implicitly identifying the domains and codomains of Z and Z^t)

trophic cancellation and/or round-off error dominates in the integration calculations (§4.2.1) associated with the degree of freedom. For all the examples below, we eliminate a *virtual* degree of freedom i from the linear system whenever $A_{ii} \leq 1 \times 10^{-12} \max_j A_{jj}$, i.e., when its corresponding diagonal entry in the stiffness matrix is vanishingly small. We found this elimination to be occasionally necessary to improve the solve times and/or reduce the error in the approximate solution. An alternative solution to this problem of poorly supported degrees of freedom is to perturb the boundary or interface away from grid vertices lying too close (via a perturbation of the level set function values), thus attempting to give sufficient support to all degrees of freedom.

4.4.1 Embedded Neumann Example 1

Our first two examples apply our method to the embedded Neumann problem:

$$\begin{aligned} -\nabla \cdot (\beta(\mathbf{x}) \nabla u(\mathbf{x})) &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega; \\ \beta \nabla u \cdot \hat{\mathbf{n}} &= q(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega_n. \end{aligned}$$

This first example uses $\beta(x, y, z) = 2+y^2+xz$ and sets f and q according to the exact solution $u(x, y, z) = x \cos y+y^2 \sin z$. The domain is given by $\Omega = \{\mathbf{x} : 0.4 < \|\mathbf{x}\|_2 \text{ and } \|\mathbf{x}\|_\infty < 1\}$, with Neumann conditions applied to the embedded portion of the boundary $\partial\Omega_n = \{\mathbf{x} : \|\mathbf{x}\|_2 = 0.4\}$ and Dirichlet conditions applied to the grid-aligned portion of the boundary $\partial\Omega_d = \{\mathbf{x} : \|\mathbf{x}\|_\infty = 1\}$. Figure 4.11 depicts the geometry at resolution $N = 32$, a convergence plot of the errors, and several z-slices of u^h at $N = 32$. A least-squares linear regression on the error data yields a convergence order of 1.893 for u and 1.002 for ∇u.

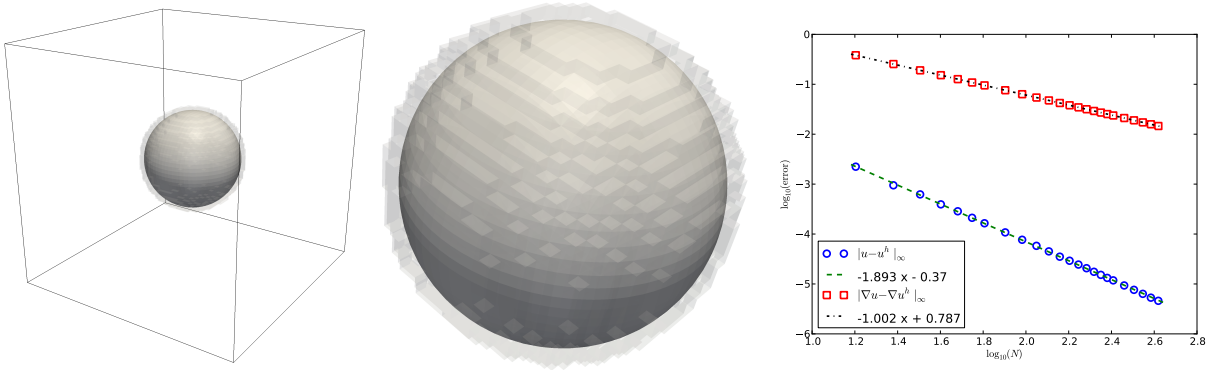
4.4.2 Embedded Neumann Example 2

Our second example is also an embedded Neumann problem, with $\beta(x, y, z) = 3 + x \cos z + y \sin z$ and f and q set according to the exact solution $u(x, y, z) = z \cos(x^2 - y^2)$. The domain Ω is bounded by the 24-point star level set given in Algorithm 4.5 with parameters $r_{\min} = 0.6$ and $r_{\max} = 0.9$. Additionally, we rotate the star surface described in Algorithm 4.5 by -0.3 radians about the $+x$-axis (to introduce some asymmetry). See Figure 4.12 for a graphic of the star level set at resolution $N = 64$.

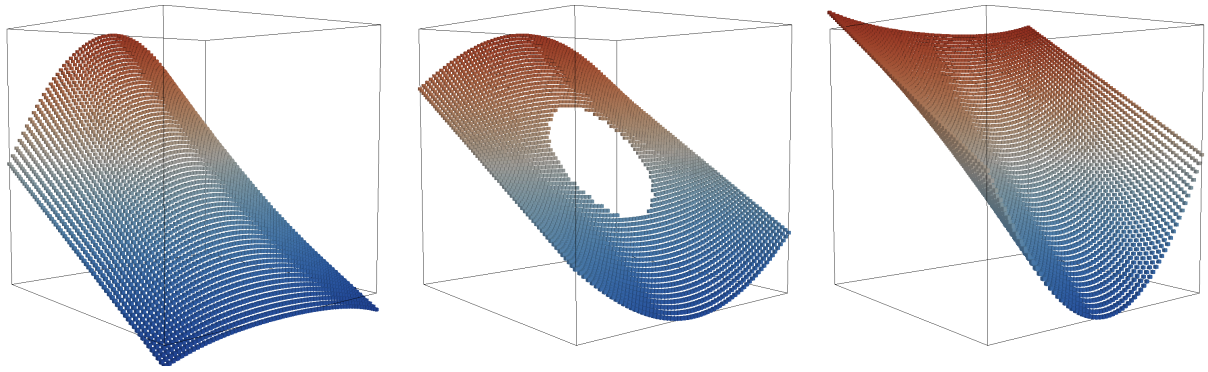
We apply Neumann boundary conditions over the entire star surface ($\partial\Omega_n = \partial\Omega$), hence the solution u is only determined up to a constant shift. We accounted for this both during during the linear solves (the stiffness matrix is indefinite) and in the evaluation of the error. Figure 4.12 shows the convergence plot of the errors and some typical z-slices of u^h at $N = 64$. We obtain convergence orders of 1.775 and 0.875 for u and ∇u, respectively.

4.4.3 Embedded Dirichlet Example

We next demonstrate our method on the embedded Dirichlet problem:

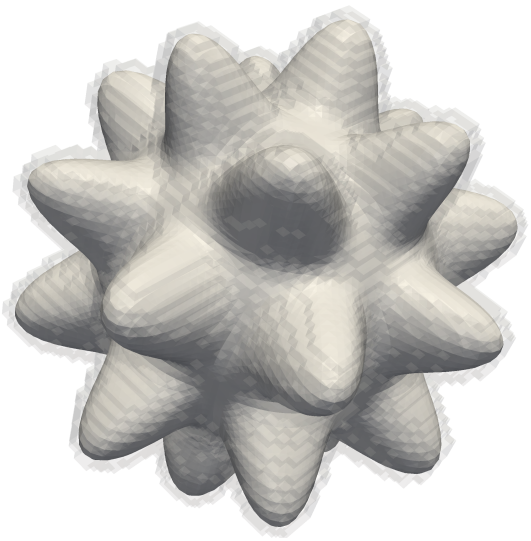


(a) Geometry of $\partial\Omega = \partial\Omega_n \sqcup \partial\Omega_d$ (b) Close-up embedded geometry of $\partial\Omega_n \subset \partial\Omega$ (c) Estimated orders of 1.893 for u, 1.002 for ∇u

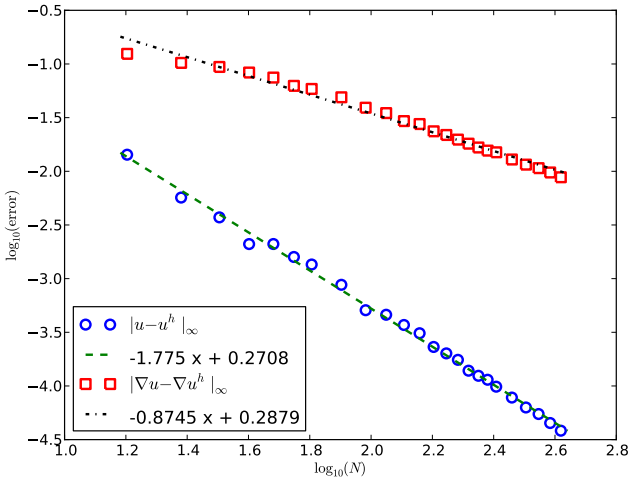


(d) $z = -5/8$ slice of u^h (e) $z = 0$ slice of u^h (f) $z = +5/8$ slice of u^h

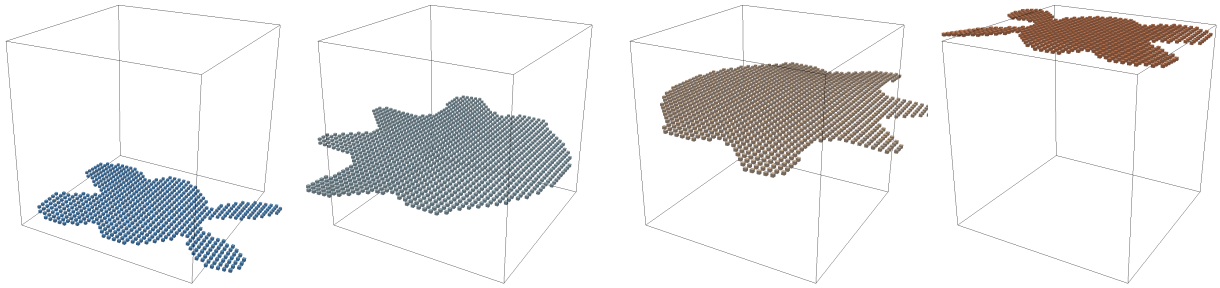
Figure 4.11: Figures for Example 4.4.1: geometry of $\partial\Omega$ at $N = 32$, convergence plot of the errors, and z-slices of u^h at $N = 32$. The black wireframe box in (c) - (e) is $\{(x, y) \in [-1, +1]^2\} \times [-1, +1]$.



(a) Embedded geometry of $\partial\Omega_n = \partial\Omega$



(b) Estimated orders of 1.775 for u, 0.875 for ∇u



(c) $z = -1/2$ slice of u^h (d) $z = -5/32$ slice of u^h (e) $z = +5/32$ slice of u^h (f) $z = +1/2$ slice of u^h

Figure 4.12: Figures for Example 4.4.2: geometry of $\partial\Omega_n$ at $N = 64$, convergence plot of the errors, and z-slices of u^h at $N = 64$. The black wireframe box in (c) - (f) is $\{(x, y) \in [-1/2, +1/2]^2\} \times [-1/2, +1/2]$.

Algorithm 4.5 Level set function for the 24-point star surface in Example 4.4.2.

{input: $\mathbf{x} \in \mathbb{R}^3$}
{parameters: $0 < r_{\min} < r_{\max}$}
let $i := \text{argmax}_i |x_i|$
if $x_i = 0$ **then**
  return $-r_{\min}$
end if
let $j_1, j_2 \in \{1, 2, 3\}$ be the other 2 indices other than i
let $s_k := x_{j_k} / |x_i|$, for $k = 1, 2$
$\{s_1, s_2$ are local coordinates on the face of the $[-1, +1]^3$ cube intersected by the ray from $\mathbf{0}$ through $\mathbf{x}\}$
assert$(-1 \le s_k \le +1)$, for $k = 1, 2$
$s_k \leftarrow \frac{1}{2}\left(s_k + \sin\frac{\pi}{2}s_k\right)$ {apply a slight distortion to give better spacing to the star's points}
let $h := (1 - \cos 2\pi s_1)(1 - \cos 2\pi s_2)$
return $|\mathbf{x}| - (r_{\min} + (r_{\max} - r_{\min})h)$

$$-\nabla \cdot (\beta(\mathbf{x})\nabla u(\mathbf{x})) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega;$$
$$u = p(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega_d.$$

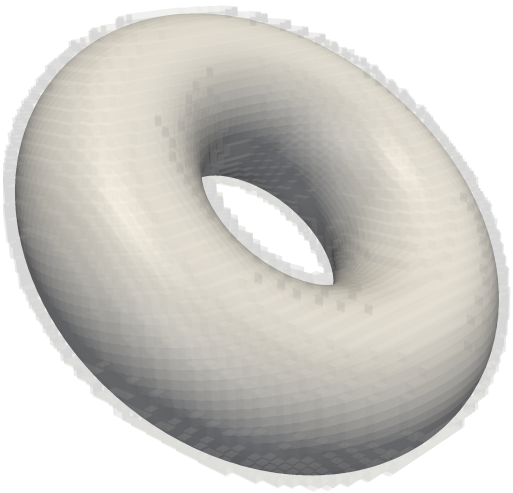
This example uses $\beta(x, y, z) = 7 + x + 2y + 3z$ and sets f and p according to the exact solution $u(x, y, z) = xe^y + \sqrt{1 + y^2}e^z$. The domain Ω is bounded by a torus centered at $\mathbf{0}$ with major radius 0.6, minor radius 0.3, and axis along $(0, -\sin 0.75, \cos 0.75)$ (the $\hat{\mathbf{k}}$ vector rotated -0.75 radians with respect to the $+x$-axis; again, to introduce some asymmetry). We apply Dirichlet boundary conditions over all of $\partial\Omega$, i.e., $\partial\Omega_d = \partial\Omega$. Figure 4.13 depicts a graphic of the torus surface at resolution $N = 64$, a convergence plot of the errors, and a few x-slices of u^h at $N = 64$ (that is, we plot $u^h(x_0, y, z)$ against (y, z) for fixed $x = x_0$). We calculated convergence orders of 1.864 and 0.977 for u and ∇u, respectively.

4.4.4 Embedded Interface Examples

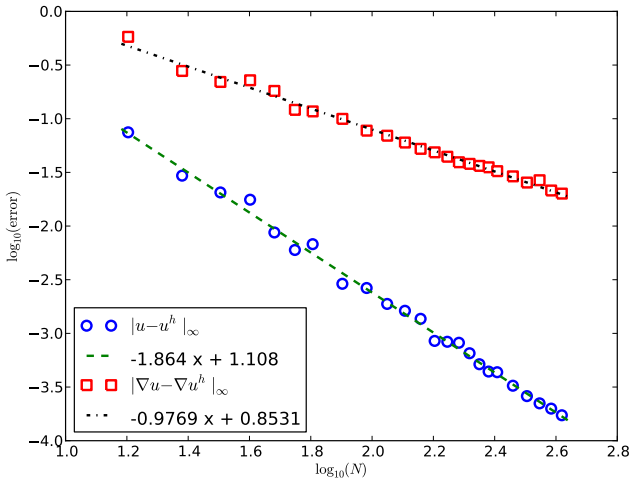
We now apply our method to the embedded interface problem:

$$-\nabla \cdot (\beta(\mathbf{x})\nabla u(\mathbf{x})) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega;$$
$$[u] = a(\mathbf{x}), \quad \mathbf{x} \in \Gamma;$$
$$[\beta\nabla u \cdot \hat{\mathbf{n}}] = b(\mathbf{x}), \quad \mathbf{x} \in \Gamma.$$

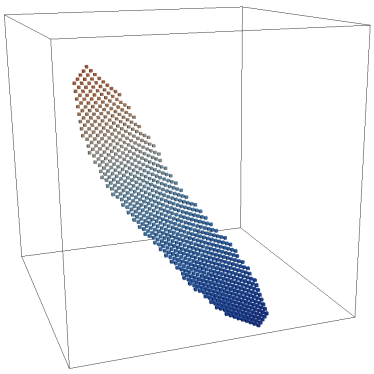
We take $\beta^-(x, y, z) = \alpha^-(10+\sin(xy+z))$ and $\beta^+(x, y, z) = \alpha^+(10+\cos(x+yz)$, where α^- and α^+ are constants. We will vary the ratio α^-/α^+ between $1/100$ and 100 to demonstrate



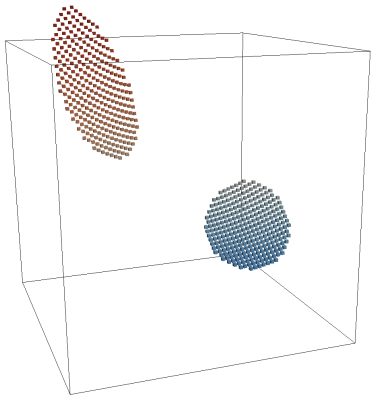
(a) Embedded geometry of $\partial\Omega_d = \partial\Omega$



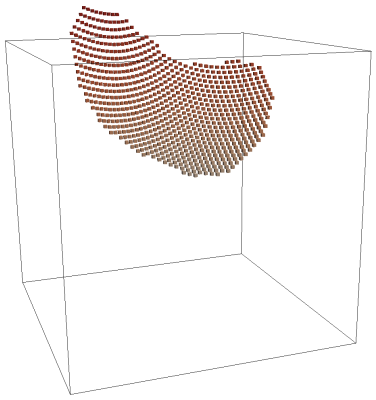
(b) Estimated orders of 1.864 for u, 0.977 for ∇u



(c) $x = -1/2$ slice of u^h



(d) $x = 0$ slice of u^h



(e) $x = +1/2$ slice of u^h

Figure 4.13: Figures for Example 4.4.3: geometry of $\partial\Omega_d$ at $N = 64$, convergence plot of the errors, and x-slices of u^h at $N = 64$. The black wireframe box in (c) - (e) is $\{(y, z) \in [-1, +1]^2\} \times [1, 3]$.

the behavior of our method with respect to the contrast in β. We set a and b according to the exact solution $u^-(x,y,z) = x^2+y^2+z^2$, $u^+(x,y,z) = (x+z)^2\sqrt{2+y}$. The interface Γ is the surface of a thickened trefoil knot, with major radius $r_{\text{major}} = 0.8$ and minor radius $r_{\text{minor}} = 0.23$. To be precise, let γ_{trefoil} denote the trefoil knot curve parameterized as

$$\gamma_{\text{trefoil}} := \left\{ \frac{r_{\text{major}}}{3} \left((2+\cos 3t)\cos 2t, (2+\cos 3t)\sin 2t, \sin 3t \right) : 0 \le t < 2\pi \right\}.$$

We then take

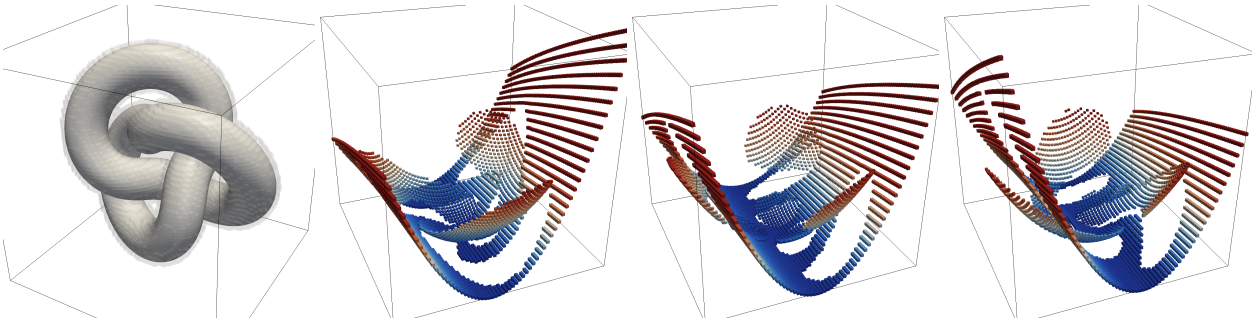
$$\Omega^- := \left\{ \mathbf{x} \in \mathbb{R}^3 : \min_{\mathbf{y}\in\gamma_{\text{trefoil}}} \|\mathbf{x}-\mathbf{y}\|_2 < r_{\text{minor}} \right\},$$

with $\Gamma = \partial\Omega^-$ and $\Omega^+ = (-1,+1)^3 \setminus (\Omega^- \sqcup \Gamma)$. See Figure 4.14 for a graphic of the trefoil knot surface at resolution $N = 64$, a few z-slices of u^h with $(\alpha^-, \alpha^+) = (2,1)$ at $N = 64$, and convergence plots of the errors for various combinations of α^- and α^+. For all tested combinations of α^- and α^+ we obtained an estimated convergence order of ≥ 1.794 and ≥ 0.923 for u and ∇u, respectively.

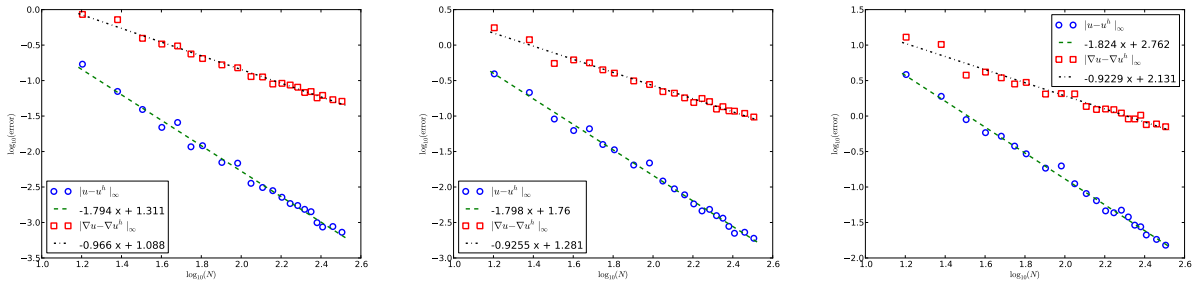
Table 4.1 shows the effect of the β contrast on the conditioning of the linear systems and the number of (preconditioned) conjugate gradient iterations. We compare the various combinations of α^- and α^+ together with, for reference, the standard 7-pt variable coefficient Laplacian with no interface. For the 7-pt Laplacian system, we show the results from using each of $\beta^- := 10 + \sin(xy+z)$ and $\beta^+ := 10 + \cos(x+yz)$ as the Laplacian coefficient throughout the whole domain. All tests are at a resolution of $N = 256$. We normalized the linear systems to have constant diagonal (Jacobi preconditioning) and solved them via PETSc's [BBG09, BBE08, BGM97] conjugate gradient function to a relative residual norm of 2.3×10^{-13} of the Jacobi preconditioned system. We configured PETSc to estimate the extreme singular values of the system upon completion of a solve and computed the condition number as the ratio of these extreme singular values. In each test case, we also demonstrate the effects of preconditioning (using PETSc's incomplete Cholesky (ICC) preconditioner, applicable since the Z^tAZ system is symmetric positive definite) on the conditioning of the system and the number of conjugate gradient iterations. We observe that high β constrasts could moderately increase solve times over low β constrasts and the standard 7-pt Laplacian matrix.

4.4.5 Discontinuity Removal

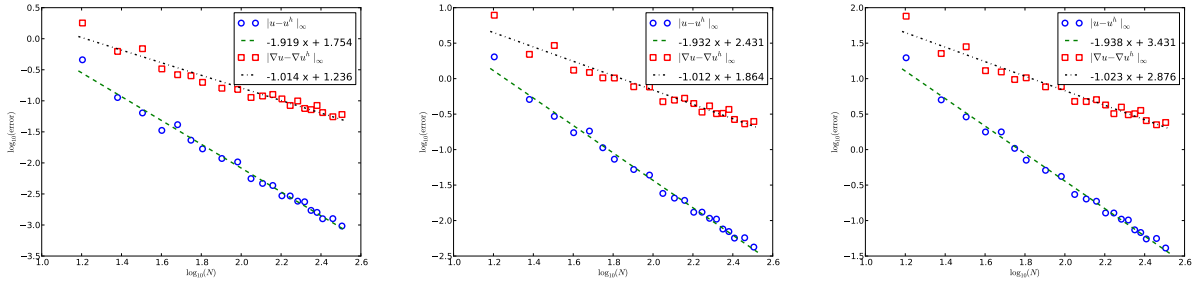
Recall from §4.2.4.1 that if β is smooth across the interface Γ, our method reduces to solving a standard 7-point Poisson system. We demonstrate the applicability of this procedure in this example. We take $\beta(x,y,z) = e^{1+x^2+z^2} + x\sin 4y$ and set a and b according to the exact solution $u^-(x,y,z) = (\cos 4x)\log(1+y^2+z^2)$, $u^+(x,y,z) = xy^2 + 3yz^2 + 7zx^2$. The interface Γ is the surface of a dumbbell, described by the level set function in Algorithm 4.6. In this example, the "balls" of the dumbbell are centered at $\mathbf{x}_0 = (-0.4,-0.4,-0.4)$ and $\mathbf{x}_1 = (0.4,0.4,0.4)$ with radii $r_{\text{ball}} = 0.5$; the "neck" of the dumbbell has radius $r_{\text{neck}} = 0.2$. See Figure 4.15 for a graphic of the dumbbell level set at $N = 64$, a convergence plot of the errors, and a few z-slices of u^h at $N = 64$. We calculated convergence orders of 1.969 and 0.984 for u and ∇u, respectively.



(a) Embedded geometry of Γ (b) $z = -1/8$ slice of u^h (c) $z = 0$ slice of u^h (d) $z = +1/8$ slice of u^h



(e) $(\alpha^-, \alpha^+) = (2, 1)$; estimated orders of 1.794 for u, 0.966 for ∇u (f) $(\alpha^-, \alpha^+) = (10, 1)$; estimated orders of 1.798 for u, 0.926 for ∇u (g) $(\alpha^-, \alpha^+) = (100, 1)$; estimated orders of 1.824 for u, 0.923 for ∇u



(h) $(\alpha^-, \alpha^+) = (1, 2)$; estimated orders of 1.919 for u, 1.014 for ∇u (i) $(\alpha^-, \alpha^+) = (1, 10)$; estimated orders of 1.932 for u, 1.012 for ∇u (j) $(\alpha^-, \alpha^+) = (1, 100)$; estimated orders of 1.938 for u, 1.023 for ∇u

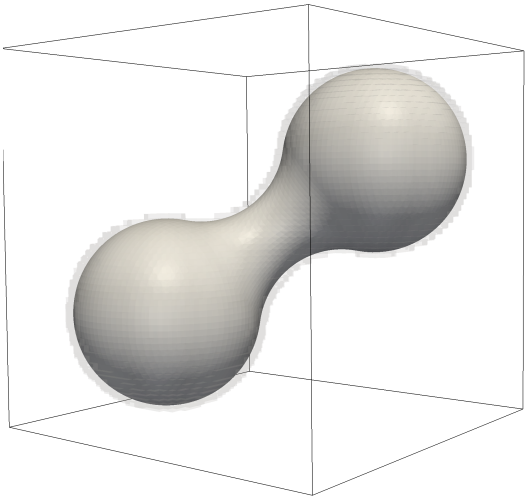
Figure 4.14: Figures for Example 4.4.4: geometry of Γ, z-slices of u^h with $(\alpha^-, \alpha^+) = (2, 1)$ at $N = 64$, and convergence plots of the errors at various combinations of α^- and α^+. The black wireframe box in (b) - (d) is $\{(x, y) \in [-1, +1]^2\} \times [0, 2]$.

Test case	cond. # (no ICC)	cond. # (w/ICC)	# CG iter.	# PCG iter.
$(2,1)$	4.0×10^5	5.6×10^3	5148	616
$(10,1)$	1.4×10^6	6.5×10^5	8421	5856
$(100,1)$	1.3×10^7	6.1×10^6	12855	8817
$(1,2)$	3.3×10^5	5.5×10^3	5168	630
$(1,10)$	4.7×10^5	2.3×10^5	6450	4529
$(1,100)$	6.6×10^5	3.1×10^5	7709	5350
7-pt Laplacian, β_-	2.7×10^4	2.7×10^3	1190	395
7-pt Laplacian, β_+	2.7×10^4	2.7×10^3	1194	427

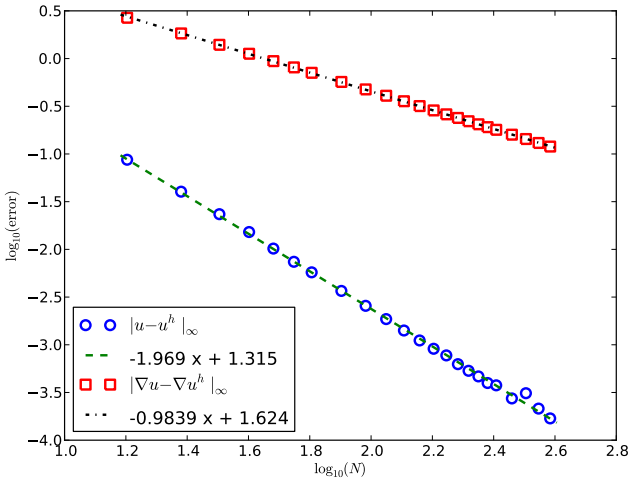
Table 4.1: Condition numbers (as estimated by PETSc) and number of (preconditioned) conjugate gradient ((P)CG) iterations for the linear systems resulting from discretizing Example 4.4.4 at resolution $N = 256$ for various combinations of (α^-, α^+). For the preconditioning, we used PETSc's incomplete Cholesky (ICC) preconditioner. We also include statistics for the standard 7-pt Laplacian matrix for reference.

Algorithm 4.6 Signed distance function for the dumbbell surface in Example 4.4.5.

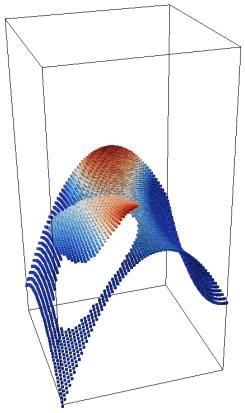
{input: $\mathbf{x} \in \mathbb{R}^3$}
{parameters: $0 < r_{\text{neck}} \leq r_{\text{ball}}$; $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{R}^3$}
let $\mathbf{y} := \mathbf{x} - \frac{1}{2}(\mathbf{x}_0 + \mathbf{x}_1)$
{(a, b) are the local coordinates of $\mathbf{x}$ projected onto the plane defined by $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}$ where $(0,0)$ corresponds to $\frac{1}{2}(\mathbf{x}_0 + \mathbf{x}_1)$ and $(\pm\tilde{a}, 0)$ corresponds to $\mathbf{x}_i$}
let $a := \mathbf{y} \cdot \frac{\mathbf{x}_1 - \mathbf{x}_0}{\|\mathbf{x}_1 - \mathbf{x}_0\|_2}$; $b := \left\| \mathbf{y} - a\frac{\mathbf{x}_1 - \mathbf{x}_0}{\|\mathbf{x}_1 - \mathbf{x}_0\|_2} \right\|$
let $\tilde{a} := \frac{1}{2}\|\mathbf{x}_1 - \mathbf{x}_0\|_2$; $\tilde{b} := \left(\tilde{a}^2 - (r_{\text{ball}} - r_{\text{neck}})^2\right) / (2(r_{\text{ball}} - r_{\text{neck}}))$
if $\tilde{b} \leq 0$ or $\frac{|a|}{\tilde{a}} + \frac{b}{\tilde{b}} \geq 1$ **then**
    let $d_0 := \sqrt{(\tilde{a} + a)^2 + b^2}$ {distance from $\mathbf{x}$ to $\mathbf{x}_0$}
    let $d_1 := \sqrt{(\tilde{a} - a)^2 + b^2}$ {distance from $\mathbf{x}$ to $\mathbf{x}_1$}
    assert $(d_i = \|\mathbf{x} - \mathbf{x}_i\|)$ for $i = 1, 2$
    return $\min\{d_0, d_1\} - r_{\text{ball}}$
else
    return $(\tilde{b} - r_{\text{neck}}) - \sqrt{a^2 + (\tilde{b} - b)^2}$
end if



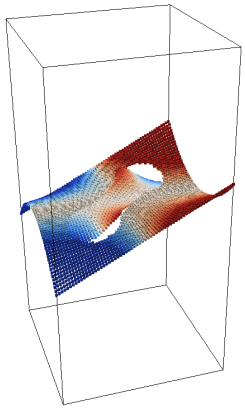
(a) Embedded geometry of Γ



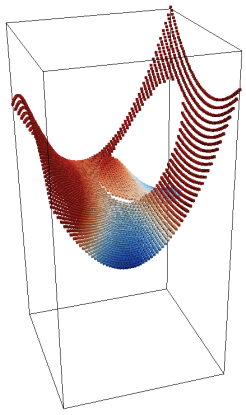
(b) Estimated orders of 1.969 for u, 0.984 for ∇u



(c) $z = -3/8$ slice of u^h



(d) $z = 0$ slice of u^h



(e) $z = +3/8$ slice of u^h

Figure 4.15: Figures for Example 4.4.5: geometry of Γ, convergence plot of the errors, and z-slices of u^h at $N = 64$. The black wireframe box in (c) - (e) is $\{(x, y) \in [-1, +1]^2\} \times [-4, 4]$.

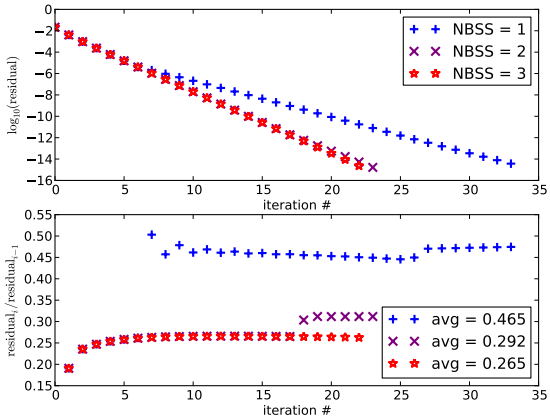
4.4.6 Multigrid

We described a collection of multigrid algorithms in §4.3 to solve domain problems with β constant (i.e., $\beta \equiv 1$) and interface problems with β^+ and β^- constant (i.e., β is constant over Ω^- and Ω^+, but not necessarily the same constant). We demonstrate the efficacy of these algorithms in this section. For each of the following examples, we study the convergence behavior of iteratively applying the multigrid v-cycle described in §4.3. We vary the number of pre-restriction and post-prolongation additional boundary/interface smoothing sweeps together with the width of the boundary/interface smoothing region, and show what kinds of parameters might be typically necessary to achieve good v-cycle convergence. We note that, generally speaking, for the class of smoothers we are using (straightforward Gauss-Seidel or variants thereof), embedded Neumann and embedded Dirichlet problems require relatively little additional smoothing effort along the boundary. Embedded interface problems, on the other hand, may require significantly more work along the interface, depending highly on the contrast in β.

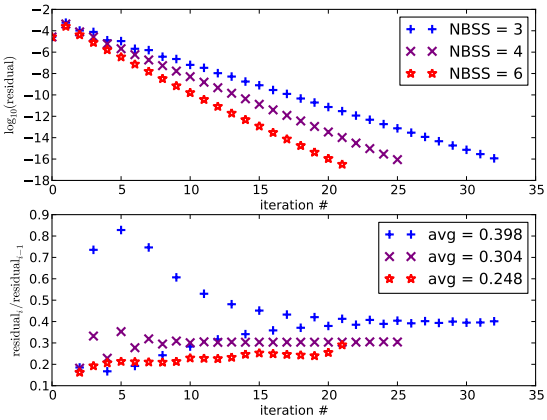
We first present the results of applying our multigrid algorithm to the embedded Neumann examples in §4.4.1 and §4.4.2, but with $\beta \equiv 1$. Figure 4.16 shows plots of the residual norm $\left\| \vec{f} - A\vec{u} \right\|_\infty$ and the ratio of successive residual norms versus the v-cycle iteration number at resolution $N = 384 = 3 \cdot 2^7$. For both examples, we were able to obtain a v-cycle convergence rate of about 0.25 with a boundary smoothing region width of only 1 (i.e., only expending extra smoothing effort on boundary degrees of freedom) and relatively few additional boundary smoothing sweeps.

Figure 4.17 shows the results of applying our multigrid algorithm to the embedded Dirichlet example in §4.4.3 (except, again, with $\beta \equiv 1$). For this example, we found it necessary to extend the boundary smoothing region out to a width of 2 or 3 to obtain good v-cycle convergence, encompassing all degrees of freedom incident to a grid cell with an L^∞-distance from a boundary grid cell of at most 1 or 2, respectively. In each case, we needed only 3 or 4 additional boundary smoothing sweeps to achieve a stable v-cycle convergence rate. Additional boundary smoothing sweeps above 3 or 4 did not significantly improve the convergence rate. Unsurprisingly, a boundary smoothing region width of 3 gives a better convergence rate (again, about 0.25) than a boundary smoothing region width of 2 (where the convergence rate is, at best, about 0.39). The former, however, requires non-negligibly more effort for smaller resolutions.

Lastly, we demonstrate our multigrid algorithm on the embedded interface example in §4.4.4 with $\beta^- \equiv \alpha^-$ and $\beta^+ \equiv \alpha^+$. See Figure 4.18 for the results. Here, we vary α^-/α^+ only between 1/10 and 10. As for the embedded Dirichlet case, an interface smoothing region width of 2 or 3 is sufficient to obtain a v-cycle convergence rate of about 0.40 or 0.25, respectively. We found that we also needed significantly more additional interface smoothing sweeps than for the embedded Neumann and embedded Dirichlet cases, especially at more extreme β contrasts (e.g., 1/100 or 100).

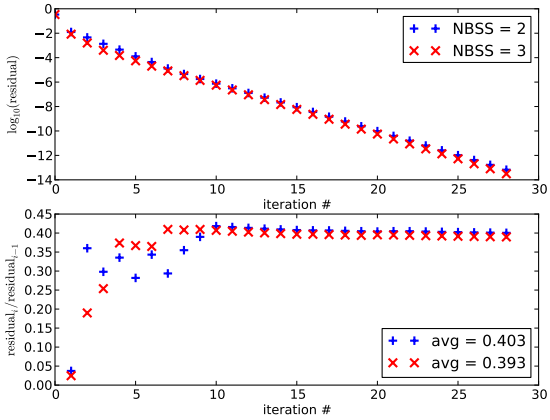


(a) Example 4.4.1

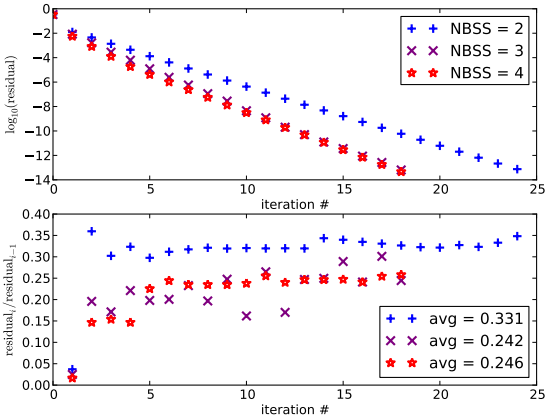


(b) Example 4.4.2

Figure 4.16: Multigrid v-cycle convergence plots for embedded Neumann Examples 4.4.1 and 4.4.2 with $\beta \equiv 1$. The grid resolution is $N = 384$ and the boundary smoothing region width is 1. The top plot in each subfigure shows the residual norm $\left\|\vec{f} - A\vec{u}\right\|_\infty$ after each v-cycle iteration for various numbers of boundary smoothing sweeps (NBSS). The bottom plots shows the ratio of successive residual norms. The estimated rate given in each bottom plot is the average ratio of successive residual norms over the final 10 iterations.

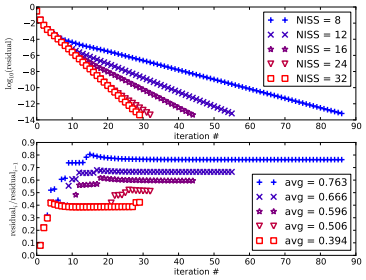


(a) BSRW = 2

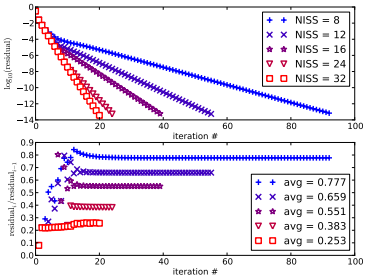


(b) BSRW = 3

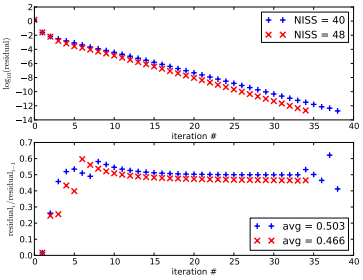
Figure 4.17: Multigrid v-cycle convergence plots for embedded Dirichlet Example 4.4.3 with $\beta \equiv 1$ for a boundary smoothing region width (BSRW) of 2 and 3. The grid resolution is $N = 384$. The top plot in each subfigure shows the residual norm $\left\|\vec{f} - A\vec{u}\right\|_\infty$ after each v-cycle iteration for various numbers of boundary smoothing sweeps (NBSS). The bottom plots shows the ratio of successive residual norms. The estimated rate given in each bottom plot is the average ratio of successive residual norms over the final 10 iterations.



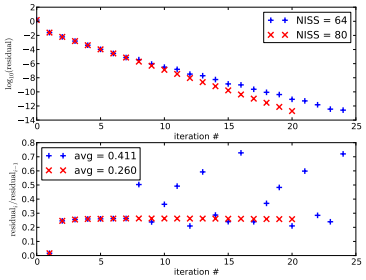
(a) $(\alpha^-, \alpha^+) = (2, 1)$, ISRW $= 2$



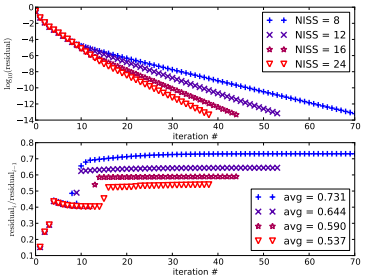
(b) $(\alpha^-, \alpha^+) = (2, 1)$, ISRW $= 3$



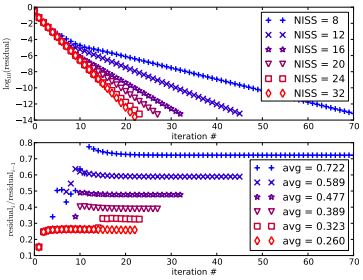
(c) $(\alpha^-, \alpha^+) = (10, 1)$, ISRW $= 2$



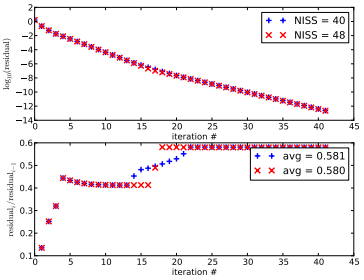
(d) $(\alpha^-, \alpha^+) = (10, 1)$, ISRW $= 3$



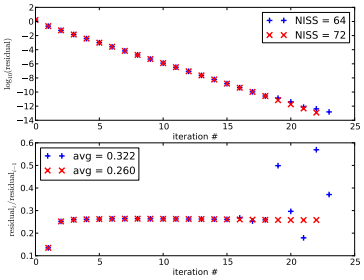
(e) $(\alpha^-, \alpha^+) = (1, 2)$, ISRW $= 2$



(f) $(\alpha^-, \alpha^+) = (1, 2)$, ISRW $= 3$



(g) $(\alpha^-, \alpha^+) = (1, 10)$, ISRW $= 2$



(h) $(\alpha^-, \alpha^+) = (1, 10)$, ISRW $= 3$

Figure 4.18: Multigrid v-cycle convergence plots for embedded interface Examples 4.4.4 with $\beta^- \equiv \alpha^-$, $\beta^+ \equiv \alpha^+$ for a interface smoothing region width (ISRW) of 2 and 3 and various combinations of α^-, α^+. The grid resolution is $N = 256$. The top plot in each subfigure shows the residual norm $\left\| \vec{f} - A\vec{u} \right\|_\infty$ after each v-cycle iteration for various numbers of interface smoothing sweeps (NISS). The bottom plots shows the ratio of successive residual norms. The estimated rate given in each bottom plot is the average ratio of successive residual norms over the final 10 iterations.

4.5 Discussion, Conclusion, and Future Work

We presented a virtual node method to solve embedded Neumann, Dirichlet, and interface problems (4.1) (cf. [BBZ10]) which uses Lagrange multipliers to enforce the Dirichlet condition (4.1d) and the jump condition (4.1b) weakly. We described a general algorithm to define the Lagrange multiplier space that ultimately yields a symmetric positive definite system with better conditioning than that yielded when using the double-wide constraints described in [BBZ10]. The geometric intuitiveness of our method makes it relatively easy to implement, and the numerical examples in §4.4 demonstrate its second order accuracy in L^∞. Although simpler embedded domain discretizations exist (see, for example, [GFC02] and [NMG09a]), we believe one distinct advantage of our embedded domain discretizations is that they naturally extend to our embedded interface discretization. Thus, it takes relatively little machinery to understand and implement all three methods.

We described a collection of multigrid algorithms in §4.3 to solve our embedded Neumann, Dirichlet, and interface problems. The results given in §4.4.6 demonstrate that simple v-cycle iteration built around our multigrid algorithms yields an efficient solver for embedded Neumann and embedded Dirichlet problems at almost any resolution. Using simple v-cycle iteration to solve embedded interface problems requires a significant amount of interface-local smoothing, so it would likely be most effective at higher resolutions. One avenue of research would be to investigate alternative grid-transfer operators or smoothers around the embedded interface with the hopes of reducing the amount of interface-local smoothing. We would also expect that far fewer interface-local (and boundary-local) smoothing sweeps would be necessary when using a single multigrid v-cycle as a preconditioner to a Krylov method, such as is done in [MST10].

CHAPTER 5

Nearly Incompressible Linear Elasticity

5.1 Background and Existing methods

[1] To review, this chapter addresses the solution of the equilibrium equations of linear elasticity, repeated here for convenience:

$$-\left(\mu\Delta\mathbf{I} + (\lambda+\mu)\nabla\nabla^t\right)\mathbf{u} = \mathbf{f} \quad \in \Omega \tag{5.1a}$$

$$\mathbf{u} = \mathbf{u}_0 \quad \in \partial\Omega_d \tag{5.1b}$$

$$\mu\left(\mathbf{u}\cdot\hat{\mathbf{n}} + \nabla\left(\mathbf{u}\cdot\hat{\mathbf{n}}\right)\right) + \lambda\left(\nabla\cdot\mathbf{u}\right)\hat{\mathbf{n}} = \mathbf{g} \quad \in \partial\Omega_n. \tag{5.1c}$$

where we wish to solve for the unknown displacement map $\mathbf{u}$.

Following the early methods of Hyman [Hym52] and Saul'ev [Sau63], the fictitious domain approach has been used with incompressible materials in a number of works [BTT97, GPP94b, GPH99, GPH01, BYZ04, Par08, Rut08, PP09, TP09]. These approaches embed the irregular geometry in a more simplistic domain for which fast solvers exist (e.g., fast Fourier transforms). The calculations include fictitious material in the complement of the domain of interest. A forcing term (often from a Lagrange multiplier) is used to maintain boundary conditions at the irregular geometry. Although these techniques naturally allow for efficient solution procedures, they depend on a smooth solution across the embedded domain geometry for optimal accuracy, which is not typically possible.

The *eXtended Finite Element Method* (XFEM) and related approaches in the finite element literature also make use of geometry embedded in regular elements. Although originally developed for crack-based field discontinuities in elasticity problems, these techniques are also used with embedded problems in irregular domains. Daux et al. first showed that these techniques can naturally capture embedded Neumann boundary conditions [DMD00, SCM01]. These approaches are equivalent to the variational cut cell method of Almgren et al. in [ABC97]. Enforcement of Dirichlet constraints is more difficult with variational cut cell approaches [MBT06, LB08] and typically involves a Lagrange multiplier or stabilization. Dolbow and Devan recently investigated the convergence of such approaches with incompressible materials and point out that much analysis in this context remains to be completed [DD04]. Despite the lack of thorough analysis, such XFEM approaches appear to be very accurate

[1] The content of this chapter is a version of "A second-order virtual node algorithm for nearly incompressible linear elasticity in irregular domains" by Yongning Zhu, Yuting Wang, Jeffrey Hellrung, Alejandro Cantarero, Eftychios Sifakis, and Joseph M. Teran (accepted for publication in *Journal of Computational Physics*, 2012) with moderate revisions.

and have been used in many applications involving incompressible materials in irregular domains [WML01, CB03, CC05, GW08, BBH09].

There are also many *Finite Difference Methods* (FDM) and *Finite Volume Methods* (FVM) that utilize cut uniform grid cells. Many of these methods have been developed in the context of incompressible flow. For example, Almgren et al. use cut uniform bilinear cells to solve the Poisson equation for pressures in incompressible flow calculations [ABC97]. Marelle et al. use collocated grids and define define sub cell interface and boundary geometry in cut cells via level sets [MKL05]. Ng et al. also use level set descriptions of the irregular domain and achieve second order accuracy in L^∞ for incompressible flows [NMG09b]. The approach of Batty et al. is similar, but not as accurate [BBB07]. Although not technically a cut cell approach, the immersed interface method has been used to improve accuracy for incompressible flow calculations in irregular domains [WB00, RW04, LWI06, CS08, Rut08, XW08]. Cut cell FDM and FVM have also been developed for incompressible and nearly incompressible elastic materials. Bijelonja et al. use cut cell FVM to enforce incompressibility more accurately than is typically seen with FEM [BDM06]. Beirão da Veiga et al. use polygonal FVM cells to avoid remeshing with irregular domains [VGL09]. Barton et al. [BD10] and Hill et al. [HPO10] use cut cells with Eulerian elastic/plastic flows.

Many approaches have been proposed to solve elasticity equations in a scalable way at high resoultions. For this class of problems, iterative methods are usually employed rather than direct methods due to memory considerations. For iterative methods to be scalable, we mean that the method requires only a constant (and small) number of iterations, independent of the grid resolution, to obtain a solution. While many methods of this type have proven quite effective, accommodating mixed boundary conditions on an embedded interface is highly nontrivial, especially when efficiency of implementation is a priority. Most methods have also been created specifically to work with either purely Dirichlet or purely traction boundary conditions, but have not been demonstrated to be effective in both cases. Constructing preconditioners for solving the KKT systems that result from discretizing the equations in a mixed formulation have been studied by Klawonn [Kla95, Kla98] and Bramble and Pasciak [BP88]. Work has also been done on using domain decomposition methods with PCG [FLP00] and GMRES [KP98] to solve Stokes and elasticity problems. *Balancing Domain Decomposition by Constraints* (BDDC) has also been used to build preconditioners for solving these problems [Doh03, PWZ10]. Many authors have also looked at applying multigrid methods to problems in solid mechanics [Ver84, KM87, Hau90, Bre93, CMM98, AP99, HH99, Sch99, Wie00, HMM04, GGL08, LWC09, ZST10], including handling issues arising from nearly incompressible materials. Mixed FEM formulations are one example that maintain good multigrid convergence properties for nearly incompressible materials demonstrated on the Dirichlet boundary case [Bre93, Sch99, LWC09]. FOSLS methods have been demonstrated to produce systems that can be effectively solved using algebraic multigrid methods by rewriting the elasticity equations as a first order system using least squares [CMM98, HMM04]. Multigrid applied to FEM discretized equations using a smoother based on a Schur complement has been studied by different authors [AP99, Wie00]. While demonstrating the ability to solve large problems, the Schur complement approach requires the action of the inverse of the displacements matrix in the smoothing process which is a more

expensive smoothing operation than that offered by other methods. Distributive smoothers offer a different option for the smoothing process that has proved effective on elasticity equations discretized with FEM [HH99] and on staggered grids [GGL08]. In our approach, we will look at using distributive smoothing similar to those described in Gaspar et al. [GGL08].

5.2 Mixed Finite Element Formulation

In order to accurately handle linear elastic materials near the incompressible limit, we use an augmented form of the equilibrium equations. By introducing a pressure variable as an unknown, we can achieve a stable numerical discretization independent of the degree of incompressibility. We will use the weak form of this augmented system to derive a mixed finite element formulation [BF91]. The augmented form of our equations arises by introducing $p := -(\lambda/\mu)\nabla \cdot \mathbf{u}$. With this definition, $\boldsymbol{\sigma}(\mathbf{u}) = \mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^t) - \mu p\mathbf{I}$ and the derived equations

$$-\mu(\Delta\mathbf{I} + \nabla\nabla^t)\mathbf{u} + \mu\nabla p = \mathbf{f} \quad \in \Omega; \tag{5.2a}$$

$$-\mu\nabla \cdot \mathbf{u} - \frac{\mu^2}{\lambda}p = 0 \quad \in \Omega; \tag{5.2b}$$

$$\mathbf{u} = \mathbf{u}_0 \quad \in \partial\Omega_d; \tag{5.2c}$$

$$\mu(\mathbf{u} \cdot \hat{\mathbf{n}} + \nabla(\mathbf{u} \cdot \hat{\mathbf{n}})) - \mu p\hat{\mathbf{n}} = \mathbf{g}; \quad \in \partial\Omega_n \tag{5.2d}$$

are then equivalent to the original equations (5.1).

We use this augmented form of the equations to derive an equivalent variational form of the equilibrium equations of linear elasticity. A weak form can be derived by taking the inner product of the strong form with an arbitrary vector-valued function $\mathbf{v} \in \mathbf{V}_0 := (H^1_{0,\partial\Omega_d}(\Omega))^d$ and by enforcing $p = -(\lambda/\mu)\nabla \cdot \mathbf{u}$ weakly:

$$\text{Find } (\mathbf{u}, p) \in (H^1(\Omega))^d \times L^2(\Omega),\ \mathbf{u}|_{\partial\Omega_d} = \mathbf{u}_0, \text{ such that}$$

$$\int_\Omega 2\mu \left(\frac{\nabla\mathbf{u} + (\nabla\mathbf{u})^t}{2}\right) : \left(\frac{\nabla\mathbf{v} + (\nabla\mathbf{v})^t}{2}\right) - \mu p(\nabla \cdot \mathbf{v})d\mathbf{x} \tag{5.3a}$$

$$= -\int_\Omega \mathbf{f} \cdot \mathbf{v}d\mathbf{x} + \int_{\partial\Omega_n} \mathbf{g} \cdot \mathbf{v}d\mathbf{S}(\mathbf{x}) \quad \forall\mathbf{v} \in (H^1_{0,\partial\Omega_d}(\Omega))^d, \tag{5.3b}$$

$$\int_\Omega \left(-\mu q\nabla \cdot \mathbf{u} - \frac{\mu^2}{\lambda}pq\right)d\mathbf{x} = 0 \quad \forall q \in L^2(\Omega). \tag{5.3c}$$

5.2.1 Discretization

We discretize this variational formulation using a mixed finite element method defined on a MAC-type staggered grid. Han et al. demonstrated the stability and optimal convergence of this formulation applied to the Stokes equations on a square domain [HW98]. We generalize

this approach to the case of nearly incompressible linear elasticity in embedded domains. We approximate the Sobolev space $\mathbf{V} := (H^1(\Omega))^d$ with a finite element subspace $\mathbf{V}^h$, where each displacement component of a function in $\mathbf{V}^h$ is represented as a piecewise bilinear scalar function defined on a staggered quadrilateral grid (see Figure 5.1). To be more specific, consider the staggered grids

$$\mathcal{G}_h^x := \left\{(ih, (j-1/2)h) : (i,j) \in \mathcal{I}_x \subset \mathbb{Z}^2\right\},$$
$$\mathcal{G}_h^y := \left\{((i-1/2)h, jh) : (i,j) \in \mathcal{I}_y \subset \mathbb{Z}^2\right\}.$$

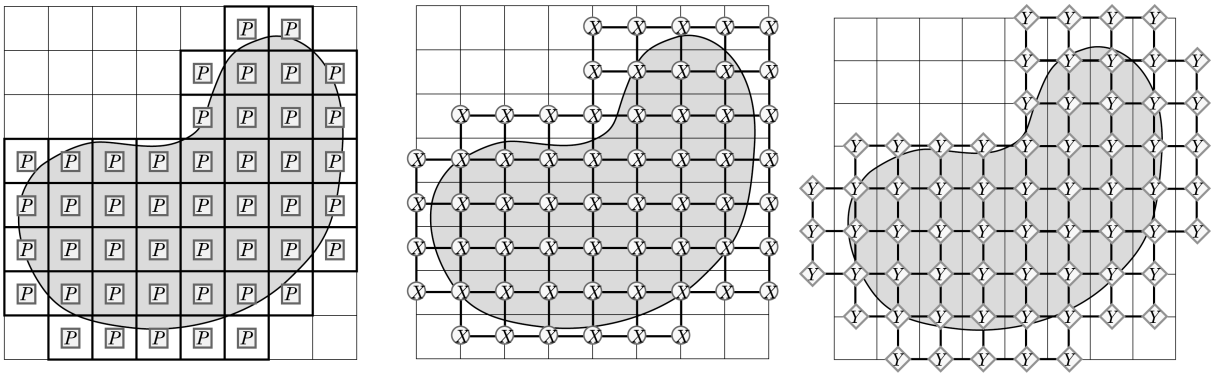
Here, h is the discrete spacing between grid points. Furthermore, we use the following notation to denote quadrilaterals defined by these grids:

$$T_{ij}^x := \{(x,y) : ih < x < (i+1)h,\ (j-1/2)h < y < (j+1/2)h\},$$
$$T_{ij}^y := \{(x,y) : (i-1/2)h < x < (i+1/2)h,\ jh < y < (j+1)h\}.$$

The sets $\mathcal{I}_x$ and $\mathcal{I}_y$ used in the definition of grids $\mathcal{G}_h^x$ and $\mathcal{G}_h^y$ are defined as the collection of vertices incident on some quadrilateral T_{ij}^x or T_{ij}^y, respectively, whose intersection with the domain Ω is non-empty. In other words, $\mathcal{I}_x$ and $\mathcal{I}_y$ are the sets of vertices in the staggered lattices that are at most an L^∞-distance of h away from Ω. Henceforth, we will use

$$\mathcal{T}_h^x := \left\{T_{ij}^x : T_{ij}^x \cap \Omega \neq \emptyset\right\},$$
$$\mathcal{T}_h^y := \left\{T_{ij}^y : T_{ij}^y \cap \Omega \neq \emptyset\right\}$$

to denote the collection of x and y grid quadrilaterals that intersect (or embed) the domain Ω.



(a) $\mathcal{T}_h^p$ (b) $\mathcal{T}_h^x$ (c) $\mathcal{T}_h^y$

Figure 5.1: Staggered grid finite element quadrangulation and embedded domain boundary.

We construct two subspaces of $H^1(\Omega)$ based on these respective quadrangulations:

$$V_x^h := \left\{v_h \in C^0(\Omega) : v_h|_{T_{ij}^x} \in Q_1(T_{ij}^x)\ \forall\, T_{ij}^x \in \mathcal{T}_h^x\right\},$$
$$V_y^h := \left\{v_h \in C^0(\Omega) : v_h|_{T_{ij}^y} \in Q_1(T_{ij}^y)\ \forall\, T_{ij}^y \in \mathcal{T}_h^y\right\},$$

where $Q_1(T_{ij}^k)$ is the space of bilinear functions on the quadrilateral T_{ij}^k. For simplicity of notation in subsequent equations we will also use the mappings

$$\eta_1 \colon I_1 := \{1, 2, \ldots, N_x\} \to \mathcal{I}_x,$$
$$\eta_2 \colon I_2 := \{1, 2, \ldots, N_y\} \to \mathcal{I}_y$$

to associate each x and y grid vertex with a unique integer between 1 and $N_x := |\mathcal{I}_x|$ and 1 and $N_y := |\mathcal{I}_y|$, respectively. With this convention, any approximate solution $\mathbf{u}^h \in V_x^h \times V_y^h$ can be expressed as

$$\mathbf{u}^h(\mathbf{x}) := \begin{pmatrix} \sum_{k_1 \in I_1} u_{k_1}^1 N_{k_1}^1(\mathbf{x}) \\ \sum_{k_2 \in I_2} u_{k_2}^2 N_{k_2}^2(\mathbf{x}) \end{pmatrix}, \tag{5.4}$$

where $N_{k_1}^1$ and $N_{k_2}^2$ are the commonly used piecewise bilinear interpolating basis functions associated with nodes k_1 and k_2, respectively, in $\mathcal{T}_h^x$ and $\mathcal{T}_h^y$. Our discrete equations for the approximate solution $\mathbf{u}^h$ can thus be seen to be over $N_x + N_y$ scalar unknowns.

We additionally approximate the pressure space $V_p := L^2(\Omega)$ with a piecewise constant finite element space V_p^h defined on a quadrangulation $\mathcal{T}_h^p$ over the primary grid (or, henceforth, the pressure grid) $\mathcal{G}_h^p$:

$$\begin{aligned}
\mathcal{G}_h^p &:= \left\{((i+1/2)h, (j+1/2)h) : (i,j) \in \mathcal{I}_p \subset \mathbb{Z}^2\right\}, \\
T_{ij}^p &:= \left\{(x,y) : ih < x < (i+1)h,\ jh < y < (j+1)h\right\}, \\
\mathcal{T}_h^p &:= \left\{T_{ij}^p : T_{ij}^p \cap \Omega \neq \emptyset\right\}, \\
V_p^h &:= \left\{p_h \in L^2(\Omega) : p_h|_{T_{ij}^p} \in P_0(T_{ij}^p)\ \forall\, T_{ij}^p \in \mathcal{T}_h^p\right\},
\end{aligned}$$

where $P_0(T_{ij}^p)$ is the space of constant functions on the quadrilateral T_{ij}^p. The grid $\mathcal{G}_h^p$ is a cell-centered grid (as opposed to a node-centered grid, such as $\mathcal{G}_h^x$ or $\mathcal{G}_h^y$); there is one pressure degree of freedom associated with the center of each pressure cell $T_{ij}^p \in \mathcal{T}_h^p$. The set $\mathcal{I}_p$ is defined similarly to $\mathcal{I}_x$ and $\mathcal{I}_y$, however here it refers to the collection of cell-centered indices in the grid $\mathcal{G}_h^p$ whose associated quadrilaterals T_{ij}^p have a non-empty intersection with Ω. For the sake of simplicity in subsequent equations, we again use a mapping

$$\eta_3 \colon I_3 := \{1, 2, \ldots, N_p\} \to \mathcal{I}_p$$

to associate each cell in the pressure grid with a unique integer between 1 and $N_p := |\mathcal{I}_p|$. Thus, any approximate pressure solution p^h has the representation

$$p^h(\mathbf{x}) := \sum_{k_3 \in I_3} p_{k_3} \chi_{T_{k_3}^p}(\mathbf{x}) \tag{5.5}$$

where (with some abuse of notation) $\chi_{T_k^p}(\mathbf{x})$ is the characteristic function associated with the quadrilateral T_k^p. That is,

$$\chi_{T_k^p}(\mathbf{x}) := \begin{cases} 1, & \mathbf{x} \in T_k^p \\ 0, & \mathbf{x} \notin T_k^p \end{cases}.$$

We choose test functions $\mathbf{v}^h(\mathbf{x}) = N^m_{k_m}(\mathbf{x})\mathbf{e}_m$ ($m \in \{1,2\}$, $k_m \in I_m$) and substitute the finite element discretization (5.4), (5.5) into each term in the mixed variational form (5.3):

$$2\mu \int_\Omega \left(\frac{\nabla \mathbf{u}^h + (\nabla \mathbf{u}^h)^t}{2} \right) : \left(\frac{\nabla \mathbf{v}^h + (\nabla \mathbf{v}^h)^t}{2} \right) d\mathbf{x} \tag{5.6a}$$

$$= \mu \int_\Omega \left(\nabla \mathbf{u}^h + (\nabla \mathbf{u}^h)^t \right) : \nabla \mathbf{v}^h d\mathbf{x} = \mu \sum_{i,j \in \{1,2\}} \int_\Omega (u^h_{i,j} + u^h_{j,i}) v^h_{j,i} d\mathbf{x} \tag{5.6b}$$

$$= \mu \sum_{i,j \in \{1,2\}} \int_\Omega (u^h_{i,j} + u^h_{j,i}) N^m_{k_m,i} \delta_{mj} d\mathbf{x} = \mu \sum_{i \in \{1,2\}} \int_\Omega (u_{i,m} + u_{m,i}) N^m_{k_m,i} d\mathbf{x} \tag{5.6c}$$

$$= \mu \sum_{i \in \{1,2\}} \int_\Omega \left(\sum_{k_i \in I_i} u^i_{k_i} N^i_{k_i,m} + \sum_{k_m \in I_m} u^m_{k_m} N^m_{k_m,i} \right) N^m_{k_m,i} d\mathbf{x} \tag{5.6d}$$

$$= \mu \sum_{i \in \{1,2\}} \sum_{k_i \in I_i} u^i_{k_i} \int_\Omega N^i_{k_i,m} N^m_{k_m,i} d\mathbf{x} + \mu \sum_{k_m \in I_m} u^m_{k_m} \sum_{i \in \{1,2\}} \int_\Omega \left(N^m_{k_m,i} \right)^2 d\mathbf{x}; \tag{5.6e}$$

$$-\mu \int_\Omega p \nabla \cdot \mathbf{v}^h d\mathbf{x} = -\mu \sum_{k_3 \in I_3} p_{k_3} \int_{T^p_{k_3} \cap \Omega} N^m_{k_m,m} d\mathbf{x}; \tag{5.6f}$$

$$\int_\Omega \mathbf{f} \cdot \mathbf{v}^h d\mathbf{x} = \int_\Omega f_m N^m_{k_m} d\mathbf{x}; \tag{5.6g}$$

$$\int_{\partial\Omega_n} \mathbf{g} \cdot \mathbf{v}^h d\mathbf{S}(\mathbf{x}) = \int_{\partial\Omega_n} g_m N^m_{k_m} d\mathbf{S}(\mathbf{x}). \tag{5.6h}$$

We can also choose $\mathbf{v}^h \equiv \mathbf{0}$ and $q^h(\mathbf{x}) = \chi_{T^p_{k_3}}(\mathbf{x})$ ($k_3 \in I_3$) to give the corresponding pressure equations:

$$-\mu \sum_{i \in \{1,2\}} \sum_{k_i \in I_i} u^i_{k_i} \int_{T^p_{k_3} \cap \Omega} N^i_{k_i,i} d\mathbf{x} - \frac{\mu^2}{\lambda} p_{k_3} \int_{T^p_{k_3} \cap \Omega} d\mathbf{x} = 0.$$

Since the variational form is derived from an energy minimization problem, the discretized linear system can trivially be seen to be symmetric. Specifically, if $\vec{u} \in \mathbb{R}^{N_x+N_y}$ is our vector of displacement unknowns (where, say, the x degrees of freedom are ordered first followed by the y degrees of freedom second) and $\vec{p} \in \mathbb{R}^{N_p}$ our vector of pressure unknowns, then our system over the vector $\tilde{u}$ of $N := N_x + N_y + N_p$ degrees of freedom is of the form:

$$\begin{pmatrix} A_u & G^t \\ G & D_p \end{pmatrix} \begin{pmatrix} \vec{u} \\ \vec{p} \end{pmatrix} = \begin{pmatrix} \vec{f} \\ \vec{0} \end{pmatrix} \quad \text{or} \quad \tilde{A}\tilde{u} = \tilde{f} \tag{5.7}$$

where $\tilde{u} = (\vec{u}\ \vec{p})$ and $\tilde{f} = (\vec{f}\ \vec{0})$. Furthermore, our use of regular grids gives the discrete equations a finite difference interpretation. If we scale the system by $1/h^2$, each block in the discrete system approximates the corresponding differential operator in (5.2), i.e., (5.7) discretizes the following equation:

$$h^2 \begin{pmatrix} -\mu(\Delta + \nabla\nabla^t) & \mu\nabla \\ -\mu\nabla^t & -\mu^2/\lambda \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} h^2\mathbf{f} \\ 0 \end{pmatrix}. \tag{5.8}$$

The linear system is the Hessian matrix of a saddle point problem, therefore the discretized system is symmetric but indefinite. Indeed, the upper-left block A_u is positive definite whie the lower-right block D_p is negative definite.

5.2.2 Implementation Details

For ease of implementation, we perform the integrations involved in the discrete equations (5.6) in an element-by-element fashion. Each area integral is represented as a sum of integrals over spatially disjoint elements whose union is the embedded domain. Specifically, we individually address the integration over the intersection of each quadrilateral of the pressure grid with the domain $T^p_{k_3} \cap \Omega$:

$$\begin{aligned}
\int_\Omega \left(N^m_{k_m,i}\right)^2 d\mathbf{x} &= \sum_{k_3 \in I_3} \int_{T^p_{k_3} \cap \Omega} \left(N^m_{k_m,i}\right)^2 d\mathbf{x};\\
\int_\Omega N^i_{k_i,m} N^m_{k_m,i} d\mathbf{x} &= \sum_{k_3 \in I_3} \int_{T^p_{k_3} \cap \Omega} N^i_{k_i,m} N^m_{k_m,i} d\mathbf{x};\\
\int_\Omega N^m_{k_m} d\mathbf{x} &= \sum_{k_3 \in I_3} \int_{T^p_{k_3} \cap \Omega} N^m_{k_m} d\mathbf{x}\\
\int_{\partial\Omega_n} N^m_{k_m} d\mathbf{S}(\mathbf{x}) &= \sum_{k_3 \in I_3} \int_{T^p_{k_3} \cap \partial\Omega_n} N^m_{k_m} d\mathbf{S}(\mathbf{x}).
\end{aligned}$$

In the interior, this simply amounts to evaluating the same integrals over each uncut quadrilateral $T^p_{k_3}$. However, at the boundary, care must be taken to respect the material region when the intersection between a pressure cell and the domain is non-trivial. In both the boundary and interior cases there will be 13 degrees of freedom involved in the integration over such a pressure cell. This is because the staggering of various grids leads to 13 interpolating basis functions supported over a given pressure cell: 6 x-basis functions, 6 y-basis functions, and 1 pressure basis function. In other words, we express the stiffness matrix A_u of our discrete linear system as a sum of 13×13 element stiffness matrices $A^{k_3}_u$. We further break the integrals involved in a given pressure cell $T^p_{k_3}$ up into four subintegrals over the sub-cell quadrants $\{\omega_1, \omega_2, \omega_3, \omega_4\}$ of $T^p_{k_p}$ (see Figure 5.2). This is because the integrands are smooth over each quadrant; indeed, the integrands are quadratic, and we simply perform these integrations analytically. This observation effectively decomposes the element stiffness matrix $A^{k_3}_u$ over the sub-cell quadrants $\{\omega_i\}$ into the sub-element stiffness matrices $\left\{(A^{k_3}_u)_{\omega_i}\right\}$. For example, referring to Figure 5.2(c), $(A^{k_3}_u)_{\omega_1}$ involves X_1, X_2, X_3, X_4, Y_7, Y_8, Y_{10}, Y_{11}, and P_{13} so it only has nonzero values on rows and columns involving these 9 degrees

of freedom. The resulting equations based on those degrees of freedom are

$$(A_u^{k_3})_{\omega_1} = \mu \left(\begin{array}{c|c|c|c} {}_i\diagdown^j & X_1 \quad X_2 \quad X_3 \quad X_4 & Y_7 \quad Y_8 \quad Y_{10} \quad Y_{11} & P_{13} \\ \hline\hline X_1 \\ X_2 \\ X_3 \\ X_4 & \int_{\omega_1} \nabla N_i^1 \cdot \nabla N_j^1 + N_{i,1}^1 N_{j,1}^1 & \int_{\omega_1} N_{i,2}^1 N_{j,1}^2 & -\int_{\omega_1} N_{i,1}^1 \\ \hline Y_7 \\ Y_8 \\ Y_{10} \\ Y_{11} & \int_{\omega_1} N_{i,1}^2 N_{j,2}^1 & \int_{\omega_1} \nabla N_i^2 \cdot \nabla N_j^2 + N_{i,2}^2 N_{j,2}^2 & -\int_{\omega_1} N_{i,2}^2 \\ \hline P_{13} & -\int_{\omega_1} N_{j,1}^1 & -\int_{\omega_1} N_{j,2}^2 & -\frac{\mu}{\lambda}\int_{\omega_1} 1 \end{array}\right)$$

If we order the 13 nodes with indices shown in Figure 5.2(a), then on the interior of the domain, where $T_{k_3}^p \subset \Omega$, the sum of these four subintegrals is always the same:

$$A_u^{k_3} = \mu \left(\begin{array}{c|c} \frac{1}{64}\left(\begin{array}{cccccc|cccccc} +16 & 0 & 0 & -16 & 0 & 0 & +9 & -6 & -3 & +3 & -2 & -1 \\ 0 & +16 & -16 & 0 & 0 & 0 & +3 & +6 & -9 & +1 & +2 & -3 \\ 0 & -16 & +96 & -64 & 0 & -16 & -6 & +4 & +2 & +6 & -4 & -2 \\ -16 & 0 & -64 & +96 & -16 & 0 & -2 & -4 & +6 & +2 & +4 & -6 \\ 0 & 0 & 0 & -16 & +16 & 0 & -3 & +2 & +1 & -9 & +6 & +3 \\ 0 & 0 & -16 & 0 & 0 & +16 & -1 & -2 & +3 & -3 & -6 & +9 \\ \hline +9 & +3 & -6 & -2 & -3 & -1 & +16 & 0 & 0 & 0 & -16 & 0 \\ -6 & +6 & +4 & -4 & +2 & -2 & 0 & +96 & 0 & -16 & -64 & -16 \\ -3 & -9 & +2 & +6 & +1 & +3 & 0 & 0 & +16 & 0 & -16 & 0 \\ +3 & +1 & +6 & +2 & -9 & -3 & 0 & -16 & 0 & +16 & 0 & 0 \\ -2 & +2 & -4 & +4 & +6 & -6 & -16 & -64 & -16 & 0 & +96 & 0 \\ -1 & -3 & -2 & -6 & +3 & +9 & 0 & -16 & 0 & 0 & 0 & +16 \end{array}\right) & -\frac{h}{8}\left(\begin{array}{c} -1 \\ +1 \\ -6 \\ +6 \\ -1 \\ +1 \\ \hline -1 \\ -6 \\ -1 \\ +1 \\ +6 \\ +1 \end{array}\right) \\ \hline -\frac{h}{8}\left(\begin{array}{cccccc|cccccc} -1 & +1 & -6 & +6 & -1 & +1 & -1 & -6 & -1 & +1 & +6 & +1 \end{array}\right) & -h^2\mu/\lambda \end{array}\right). \quad (5.9)$$

The global stiffness matrix A_u generated from the sum of all the element stiffness matrices $\left\{A_u^{k_3}\right\}$ at an interior degree of freedom (x, y, or p) has a stencil shown in Figure 5.3.

For boundary cells where $T_{k_3}^p \not\subset \Omega$, we must perform the integrations involved in each of the entries of $(A_u^{k_3})_{\omega_i}$ more carefully, taking into account the boundary geometry. The technique is similar to that presented in Chapter 4, specifically §4.2.1, and will be discussed in more detail in the following subsection. The process of constructing the global stiffness matrix A_u from each of the 13×13 element stiffness matrices $A_u^{k_3}$ is explained in Algorithm 5.1.

5.2.3 Discrete Geometric Representation and Integration

We discretize the domain Ω by embedding it in a regular Cartesian grid. Specifically, we use a level set function defined over a subgrid doubly refined with respect to $\mathcal{G}_h^x, \mathcal{G}_h^y, \mathcal{G}_h^p$:

$$\mathcal{G}^\phi = \{(ih/2, jh/2)\}\,.$$

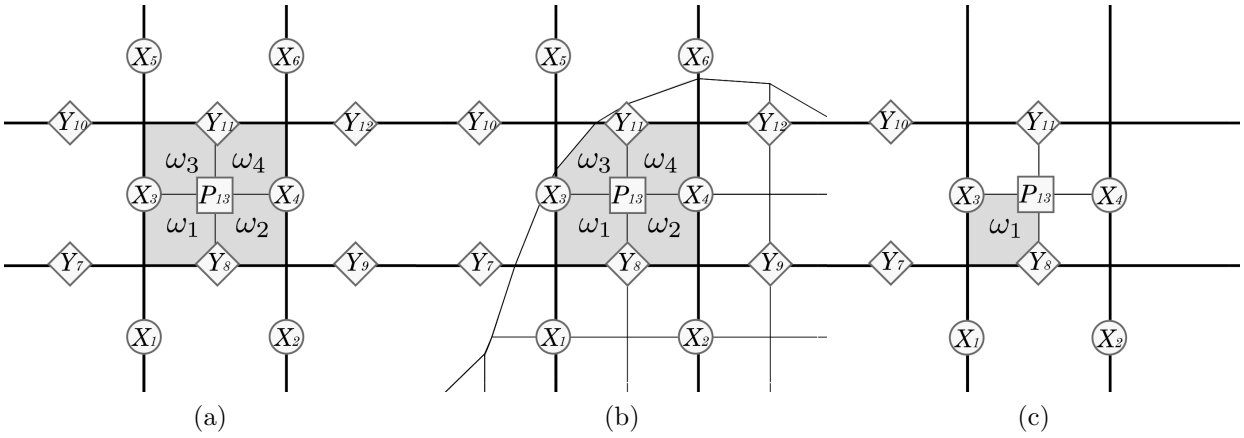


Figure 5.2: (a) A interior pressure cell and the 13 degrees of freedom involved in the corresponding element stiffness matrix. (b) A typical boundary pressure cell. (c) The degrees of freedom involved in the sub-elemental stiffness matrix corresponding to quadrant ω_1.

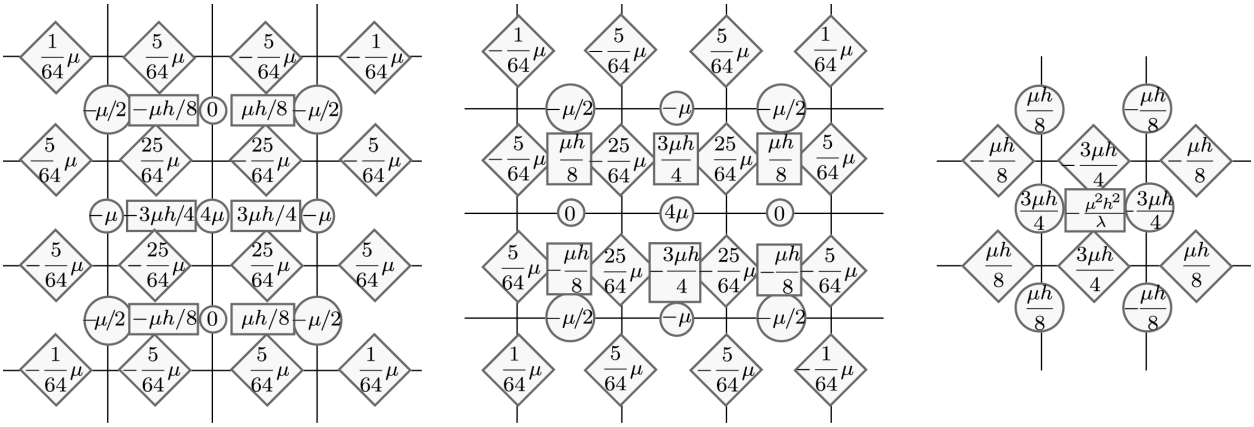


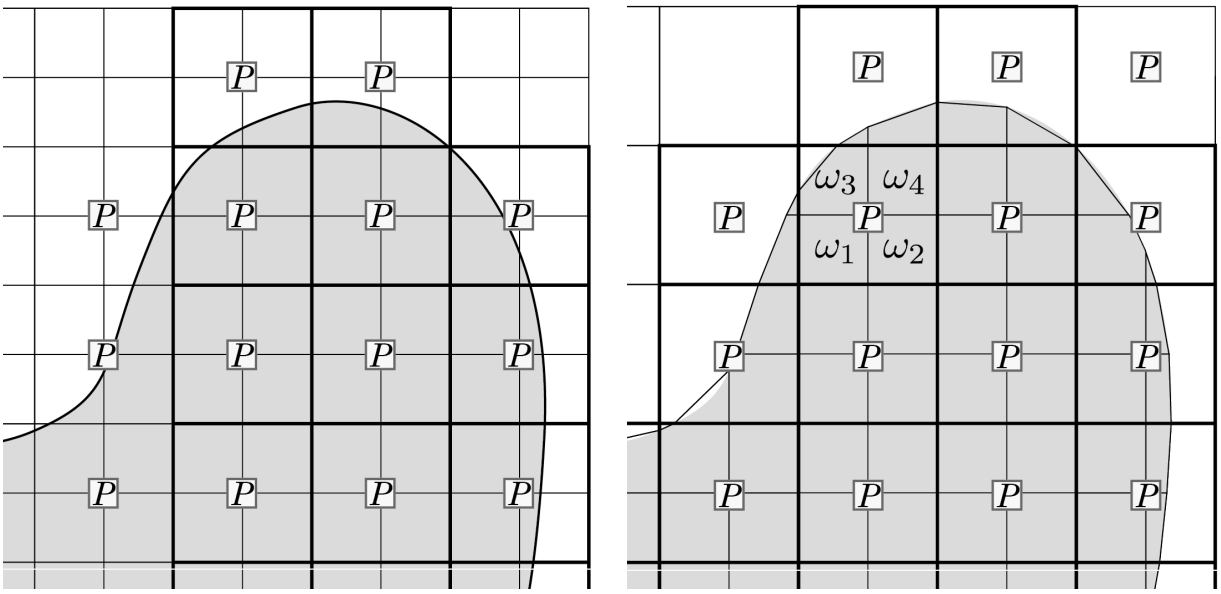
Figure 5.3: Global stiffness matrix stencils centered at an interior x degree of freedom (left), y degree of freedom (middle), and p degree of freedom (right).

Algorithm 5.1 Construction of global stiffness matrix A_u from the element stiffness matrices $A_u^{k_3}$

$A_u \leftarrow 0$
for $k_3 = 1, \ldots, N_p$ **do**
  if $T_{k_3}^p \subset \Omega$ **then**
    Use $A_u^{k_3}$ from (5.9).
  else
    Evaluate integrations over each quadrant ω_i to compute $(A_u^{k_3})_{\omega_i}$
    $A_u^{k_3} = \sum_i (A_u^{k_3})_{\omega_i}$
  end if
  for $i' = 1, \ldots, 13$ **do**
    $i := \text{mesh}(k_3, i')$ {i is the global index corresponding to the local element index i'}
    for $j' = 1, \ldots, 13$ **do**
      $j := \text{mesh}(k_3, j')$
      $(A_u)_{ij}$ += $(A_u)_{i'j'}^{k_3}$
    end for
  end for
end for

This doubly refined subgrid is thus a superset of the grids $\mathcal{G}_h^x, \mathcal{G}_h^y, \mathcal{G}_h^p$. The level set function values at the vertices of the doubly refined subgrid $\mathcal{G}^\phi$ are used to determine the points of intersection between the zero isocontour and the coordinate axes-aligned edges of $\mathcal{G}^\phi$. The boundary of Ω is then approximated by a segmented curve $\partial\Omega^h$ connecting these intersection points. The geometric domain is approximated within the region enclosed by $\partial\Omega^h$ (see Figure 5.4). Near the boundary, the domain within each subgrid cell is approximated by a polygon determined from the boundary edges of the subgrid cell and by straight lines that connect boundary intersection points as demonstrated in Figure 5.4. Thus we can think of our discrete domain as a union of doubly refined uncut quadrilaterals on the interior and polygonal regions contained within doubly refined cut quadrilaterals on the boundary.

This partitioning of the domain into doubly refined quadrilaterals naturally supports our integration conventions needed for the sub-element stiffness matrices $(A_u^{k_3})_{\omega_i}$ discussed in the previous section. The integrals needed for these matrices are trivially precomputed analytically when ω_i is uncut. On the other hand, when ω_i *is* cut by the boundary, we can still perform the integrations analytically in the same fashion as in [BBZ10] and analogous to §4.2.1. To summarize, the integrands of the requisite integrals necessary to compute the entries of $(A_u^{k_3})_{\omega_i}$ are (at most) degree 2 polynomials, hence one may easily apply the divergence theorem to explicitly transform the area integrals over $\omega_i \cap \Omega$ into line integrals over $\partial(\omega_i \cap \Omega)$. One may then analytically evaluate these latter integrals via an explicit parameterization of the individual line segments; or via an appropriate 1-dimensional Gaussian quadrature rule (which is, of course, more similar to the approach taken in §4.2.1). As with the discretization described in Chapter 4, this careful treatment of the integrals near the boundary is key to obtaining second order accuracy in L^∞.



(a) (b)

Figure 5.4: A zoomed-in view of Figure 5.1(a). We sample a the level set function implicitly defining Ω on the doubly refined subgrid depicted in (a), and use this to generate a segmend curve approximation $\partial\Omega_h$ to $\partial\Omega$, as in (b).

5.3 Dirichlet Boundary Conditions

We have thus far assumed that our solution satisfies the Dirichlet boundary conditions (5.1b) and that our test functions vanish on the Dirichlet boundary. However, because we use a regular Cartesian grid that does not conform to the actual domain, it is not convenient to directly define a finite element space with a specific value at the irregular boundary. Instead, we enforce these conditions weakly (cf. (5.3)):

$$\text{Find } (\mathbf{u}, p) \in (H^1(\Omega))^d \times L^2(\Omega) \text{ such that}$$

$$\int_\Omega 2\mu \left(\frac{\nabla \mathbf{u} + (\nabla \mathbf{u})^t}{2} \right) : \left(\frac{\nabla \mathbf{v} + (\nabla \mathbf{v})^t}{2} \right) - \mu p (\nabla \cdot \mathbf{v}) d\mathbf{x} \tag{5.10a}$$

$$= -\int_\Omega \mathbf{f} \cdot \mathbf{v} d\mathbf{x} + \int_{\partial\Omega_n} \mathbf{g} \cdot \mathbf{v} d\mathbf{S}(\mathbf{x}) \quad \forall \mathbf{v} \in (H^1_{0,\partial\Omega_d}(\Omega))^d, \tag{5.10b}$$

$$\int_\Omega \left(-\mu q \nabla \cdot \mathbf{u} - \frac{\mu^2}{\lambda} pq \right) d\mathbf{x} = 0 \quad \forall q \in L^2(\Omega), \tag{5.10c}$$

$$\int_{\partial\Omega_d} \mathbf{u} \cdot \mathbf{w} d\mathbf{S}(\mathbf{x}) = \int_{\partial\Omega_d} \mathbf{u}_0 \cdot \mathbf{w} d\mathbf{S}(\mathbf{x}) \quad \forall \mathbf{w} \in (H^{-1/2}(\partial\Omega_d))^d. \tag{5.10d}$$

Here, we introduce the Dirichlet condition as a (weak) constraint. Specifically, we require that the L^2 inner product of the solution and an arbitrary function $\mathbf{w} \in (H^{-1/2}(\partial\Omega_d))^d$ is

the same as the inner product of the Dirichlet data $\mathbf{u}_0$ with $\mathbf{w}$. This makes the problem a constrained minimization.

5.3.1 Discretizing the Dirichlet Problem

We discretize the Dirichlet constraints in a similar manner as that described in Chapter 4, §4.2.3. We approximate $(H^{-1/2}(\partial\Omega_d))^d$ with a subspace $\Lambda_x^h \times \Lambda_y^h := P_0(\mathcal{T}^x \cap \partial\Omega_d^h) \times P_0(\mathcal{T}^y \cap \partial\Omega_d^h)$, which is composed of piecewise constant functions over x and y component grid cells that intersect the Dirichlet boundary. Here we use $\partial\Omega_d^h$ to denote the portion of $\partial\Omega^h$ over which the Dirichlet constraints are being enforced. We call any x or y cell T^i with $T^i \cap \partial\Omega^h \neq \emptyset$ a boundary cell. The superscript i is used to denote whether the cell is in the x or y grids with $i = 1$ signifying an x cell and $i = 2$ signifying a y cell. We use $\{\mathbf{w}_{T^i} := \chi_{T^i}\mathbf{e}_i\}$ as the basis functions for $\Lambda_x^h \times \Lambda_y^h = P_0(\mathcal{T}^x \cap \partial\Omega_d^h) \times P_0(\mathcal{T}^y \cap \partial\Omega_d^h)$. Here, χ_{T^i} is the characteristic function of the cell T^i:

$$\chi_{T^i}(\mathbf{x}) := \begin{cases} 1, & \mathbf{x} \in T^i \\ 0, & \mathbf{x} \notin T^i \end{cases}.$$

Note that we have one basis function per boundary x cell and one per y cell. If we use N_x^d and N_y^d to denote the number of x and y boundary cells, respectively, we can see that the dimension of the space $\Lambda_x^h \times \Lambda_y^h$ is $N_x^d + N_y^d$.

With this approximation, the discretized Dirichlet constraint can be expressed as a linear system $B\mathbf{u}^h = \mathbf{u}_0^h$ ($B \in \mathbb{R}^{(N_x^d+N_y^d)\times(N_x+N_y)}$, $\mathbf{u}_0^h \in \mathbb{R}^{N_x^d+N_y^d}$). Each equation enforces an integral constraint over the intersection of the discrete boundary $\partial\Omega_d^h$ with some x or y boundary cell T^i:

$$\sum_{k_i \in I_i} u_{k_i}^i \int_{T^i \cap \partial\Omega_d^h} N_{k_i}^i d\mathbf{S}(\mathbf{x}) = \int_{T^i \cap \partial\Omega_d^h} u_0^i d\mathbf{S}(\mathbf{x}), \tag{5.11}$$

where $\mathbf{u}_0 =: (u_0^1, u_0^2)$. In practice, similar to the integrals discussed in §5.2.2, we evaluate the above integrals for a given boundary cell T^i from the four quadrants of T^i arising from the doubly refined subgrid (see §5.2.3).

The discrete constrained minimization problem may be formulated as an equivalent saddle point system involving Lagrange multipliers:

$$\begin{pmatrix} A_u & G^t & B^t \\ G & D_p & 0 \\ B & 0 & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ \vec{p} \\ \vec{\lambda} \end{pmatrix} = \begin{pmatrix} \vec{f} \\ \vec{0} \\ \vec{u}_0 \end{pmatrix}. \tag{5.12}$$

We have one Lagrange multiplier degree of freedom per discrete Dirichlet constraint; that is, $\vec{\lambda} \in \mathbb{R}^{N_x^d+N_y^d}$. When we consider boundary equations in the sections that follow, we temporarily eliminate pressure degrees of freedom $\vec{p}$ with the substitution $A := A_u - G^t D_p^{-1} G$:

$$\begin{pmatrix} A & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ \vec{\lambda} \end{pmatrix} = \begin{pmatrix} \vec{f} \\ \vec{u}_0 \end{pmatrix}. \tag{5.13}$$

Although this latter system is extremely ill-conditioned in the nearly incompressible regime, it will simplify the exposition of the forthcoming discussion of Dirichlet boundary condition

treatment. Furthermore, our multigrid algorithms make use of this elimination near the boundary during relaxation.

Ultimately, we use a constraint aggregation algorithm similar to that described in Chapter 4, §4.2.3.2, to reformulate the discretized Dirichlet constraints. But before continuing, we would like to point out a couple important details related to the constraint matrix B (cf. §4.2.3).

- $B \in \mathbb{R}^{(N_x^d+N_y^d)\times(N_x+N_y)}$, and $N_x^d+N_y^d \ll N_x+N_y$, so the presence of B in (5.12) or (5.13) only directly affects a small subset of the $N_x + N_y$ displacement degrees of freedom.
- B consists of two decoupled blocks: one for the x boundary equations and one for the y boundary equations:

$$B = \begin{pmatrix} B_x & 0 \\ 0 & B_y \end{pmatrix}, \tag{5.14}$$

  where $B_x \in \mathbb{R}^{N_x^d \times N_x}$ and $B_y \in \mathbb{R}^{N_y^d \times N_y}$. Further, the only nonzero columns of B are associated with vertices incident to an x or y boundary grid cell. Therefore, for sufficiently interior degrees of freedom, the saddle point system (5.12) is exactly the same as (5.7).

As in Chapter 4, and specifically motivated in §4.2.3.1, we reduce the saddle point system (5.13) by eliminating the Lagrange multiplier $\vec{\lambda}$ via a *null space method* using a *fundamental basis* of the constraint matrix B. The discussion from §4.2.3.1 carries over almost entirely without modification to the present setting. To summarize, we aim to construct a matrix $Z \in \mathbb{R}^{(N_x+N_y)\times((N_x+N_y)-(N_x^d+N_y^d))}$ whose columns span the null space of B and a vector $\vec{c} \in \mathbb{R}^{N_x+N_y}$ satisfying the constraint system $B\vec{c} = \vec{u}_0$. Given such a Z and $\vec{c}$, we may solve (5.13) by first solving for $\vec{v} \in \mathbb{R}^{(N_x+N_y)-(N_x^d+N_y^d)}$ in

$$Z^t A Z \vec{v} = Z^t \left(\vec{f} - A\vec{c} \right) \tag{5.15}$$

and setting $\vec{u} = \vec{c} + Z\vec{v}$. We construct Z and $\vec{c}$ by reordering the degrees of freedom such that the leading square block of B is easily invertible (indeed, as in Chapter 4, diagonal!), say, $B = (B_m | B_{n-m})$, and setting

$$Z := \begin{pmatrix} -B_m^{-1} B_{n-m} \\ I_{n-m} \end{pmatrix}, \quad \vec{c} := \begin{pmatrix} B_m^{-1} \vec{u}_0 \\ 0_{n-m} \end{pmatrix}.$$

Note that the reordering of the degrees of freedom only makes the notation more convenient and, in practice, is not done explicitly.

It turns out that, as for the Λ_1^h-induced (i.e., *single-wide*) constraints from Chapter 4, the present discretization (5.11) given above of the Dirichlet constraint (5.10d) does not readily lend itself to constructing a fundamental basis matrix Z of B (see §4.2.3.2). Additionally, while the Λ_2^h-induced (i.e., *double-wide*) constraints from §4.2.3.2 and [BBZ10] appear to yield an adequately conditioned linear system for Poisson in 2 dimensions, the analogous

discrete constraints in the present context of nearly incompressible linear elasticity yield a relatively poorly conditioned linear system, similar to the situation of Poisson in 3 dimensions (see Appendix C for a specific example). We instead reformulate our discretization of (5.10d) using a constraint aggregation algorithm nearly identical to that described in §4.2.3.2 and Algorithm 4.1. Indeed, since the B_x and B_y blocks from (5.14) are decoupled, one may aggregate each discrete constraint set independently, and each of these aggregations is identical to the case for Poisson Dirichlet constraints described in §4.2.3.2 and Algorithm 4.1. We henceforth presume the aggregation of the single-wide constraints (5.11) into *aggregate constraints*.

5.4 Multigrid

We develop an efficient multigrid framework for the discrete linear system (5.12) / (5.13). Our method is purely geometric and based on the Multigrid Correction Scheme (see Algorithm 5.2). The framework admits a simple implementation, and at a high level, it is very similar to that described in Chapter 4, §4.3. However, in contrast §4.3, the following multigrid components are significantly more sophisticated and customized to the present discretization, which is necessary to retain near-textbook multigrid convergence rates in the presence of highly irregular domains or near the incompressible limit. The subsections that follow will detail the key components of our multigrid algorithm: a hierarchy of discretizations, a smoothing operator, and appropriate transfer operators (i.e., restriction and prolongation) between levels of the hierarchy.

Algorithm 5.2 Multigrid Defect Correction

function V-Cycle $\left(\tilde{A}^h, \tilde{u}^h, \tilde{f}^h\right)$ **:**
if resolution is low enough **then**
  $\tilde{u} \leftarrow \left(\tilde{A}^h\right)^{-1} \tilde{f}^h$
  return
end if
Pre-Relaxation $\left(\tilde{A}^h, \tilde{u}^h, \tilde{f}^h\right)$
$\tilde{f}^{2h} \leftarrow R_h^{2h}\left(\tilde{f}^h - \tilde{A}^h\tilde{u}^h\right)$ {Restriction}
V-Cycle $\left(\tilde{A}^{2h}, \tilde{u}^{2h}, \tilde{f}^{2h}\right)$
$\tilde{u}^h \leftarrow \tilde{u}^h + P_{2h}^h\tilde{u}^{2h}$ {Prolongation}
Post-Relaxation $\left(\tilde{A}^h, \tilde{u}^h, \tilde{f}^h\right)$
return

5.4.1 Discretization Hierarchy

We consider a hierarchy of grids, each corresponding to a discretization of (5.10) at a progressively larger grid resolutions. In particular, we employ a grid spacing of h on the finest level of the hierarchy (level index zero), followed by discretizations with grid spacings of $2h, 4h, \ldots, 2^L h$, for a total of $L+1$ grid levels. In detail, we construct the hierarchy as follows:

- At level ℓ of the hierarchy we define the background grids $\mathcal{G}^x_{2^\ell h}, \mathcal{G}^y_{2^\ell h}, \mathcal{G}^p_{2^\ell h}$ corresponding to the x, y, and p degrees of freedom, respectively.
- We sample the level set function implicitly defining Ω over the respective doubly refined subgrids $\mathcal{G}^\phi_h, \mathcal{G}^\phi_{2h}, \mathcal{G}^\phi_{4h}, \ldots$ at each level. Clearly, coarser grids may fail to resolve some high frequency features of the domain geometry, leading to possible incoherency between the discrete systems at neighboring levels. We will addressed such issues in our discussion of the smoothing and transfer operators.
- Using the level set function values associated with a given grid, we generate the discretized domains $\mathcal{T}^x_{2^\ell h}, \mathcal{T}^y_{2^\ell h}, \mathcal{T}^p_{2^\ell h}$ and allocate the arrays of unknowns $\vec{u}^{2^\ell h}, \vec{p}^{2^\ell h}$ and right-hand sides $\vec{f}^{2^\ell h}, \vec{f}^{2^\ell h}_p$ of the respective equations. The discrete operators $A^{2^\ell h}_u$, $G^{2^\ell h}, D^{2^\ell h}_p$ of the system (5.7) are likewise defined on the discretized domain associated with hierarchy level ℓ, following the same process detailed in §5.2.1. Note that although $\vec{f}^h_p \equiv \vec{0}$ at the finest level of our hierarchy (at least initially), the right hand side $\vec{f}^{2^\ell h}_p$ at coarser levels ($\ell \geq 1$) will generally be nonzero in the Multigrid Correction Scheme (see Algorithm 5.2).

From this point on, we will simply use h instead of $2^\ell h$ to denote the grid spacing at any specific level of the multigrid hierarchy, whenever this does not incur any ambiguity. In the presence of a Dirichlet boundary condition (5.1b), we construct an *aggregate constraint matrix* B at each level, as summarized in §5.3.1 and detailed in Chapter 4, §4.2.3.2. Each row of B corresponds to one Lagrange multiplier. Following the null space method (again, summarized in §5.3.1 and detailed in §4.2.3.1), we eliminate these multipliers by solving for the null basis coefficients $\vec{v}$ in $\vec{u} = \vec{c} + Z\vec{v}$. By definition (5.3.1) of $\vec{c}$ and Z, there is a one-to-one mapping between $\vec{v}$ components and x and y degrees of freedom which are not independent (recall that the independent degrees of freedom correspond to the leading diagonal block B_m of B; see Chapter 4, §4.2.3.2). Thus, the reduced system (5.15) is defined precisely over all non-independent degrees of freedom. Due to the fact that the reconstructed $\vec{u} := \vec{c} + Z\vec{v}$ satisfies the (aggregate) constraint equations automatically, we need not restrict any residuals for the constraint system. Further, we need not solve for $\vec{\lambda}$, thanks to our null space method, hence do not store any $\vec{\lambda}$ components nor prolongate any $\vec{\lambda}$ corrections. When we restrict the residuals of the governing equation, i.e., $\vec{r} = \vec{f} - A\vec{u} - B\vec{\lambda}$, we simply restrict zero for all equations that involve a $\vec{\lambda}$ component. In other words, we restrict zero residuals from equations involving any x or y degrees of freedom on an x or y boundary grid cell, respectively. Although omitting these equations from the intergrid transfers is

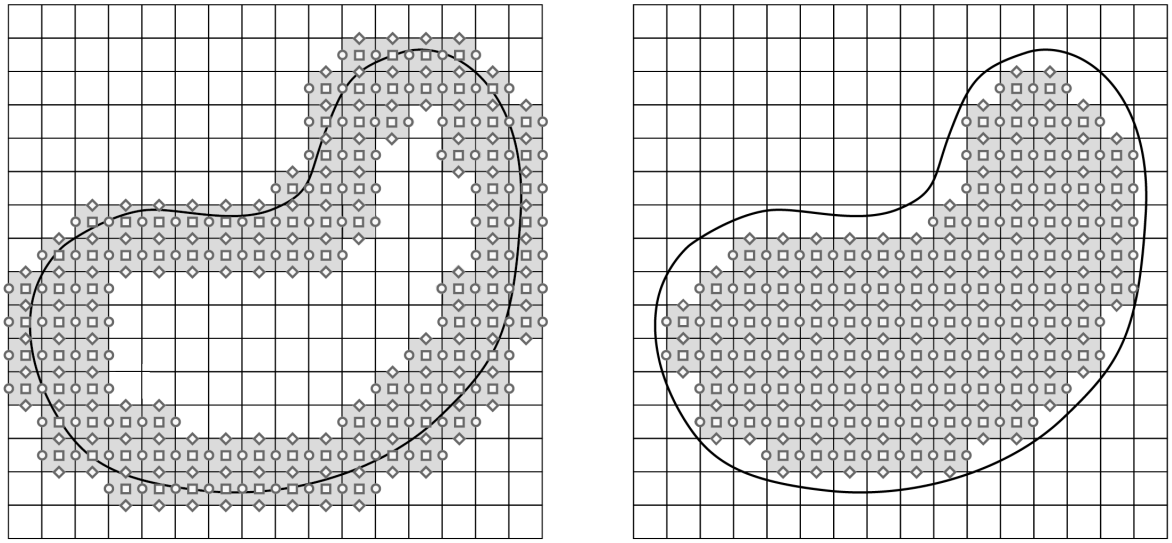
a deviation from conventional practice, we compensate by moderately increasing the pre-restriction smoothing effort in a boundary band, effectively driving the residuals in this band very close to zero (which is the value that is actually restricted!). This approach avoids the use of specialized, elaborate transfer operators involving the $\vec{\lambda}$ variables, which are not in perfect correspondence between neighboring levels due to the independent constraint aggregations employed at each level. Note that this aspect of our multigrid framework is the same as that in Chapter 4, §4.3.

5.4.2 Relaxation

The interior equations are uniform and have the same properties, while near the boundary, the equations have very different stencils. In order to design a stable and efficient relaxation scheme while keeping the computational cost low, we define two (overlapping) sets of equations, and apply an appropriate relaxation subscheme to each one. The two sets correspond to equations in the interior of the discretized domain and equations near the boundary, respectively. We define the extent of the interior region by excluding 5×5 blocks of cells, each centered around any cell that intersects the Dirichlet boundary. See Figure 5.5(b) for an example using 1×1 cell blocks. This interior region is relaxed with the distributive process detailed in §5.4.2.1.

We then define the boundary band to be the union of all 7×7 blocks of cells centered at any cell that intersects the Dirichlet boundary (excluding any cells completely exterior to the domain). This defines the set of equations to which we will apply our boundary relaxation scheme. In each single level relaxation, we first sweep over the boundary band, apply a few iterations of boundary relaxations, then apply one iteration of interior relaxation, and finally follow this by another few iterations of boundary relaxations. In Figure 5.5(a), we show an example of the cells and equations that end up in a boundary region calculated with the method described above. Soely for the sake of example, due to the coarseness of the illustrated grid, Figure 5.5 uses 3×3 cell blocks rather than the 7×7 cell blocks used in our examples.

The efficiency of a multigrid algorithm is closely related to the smoothing efficiency of a single level relaxation. With Poisson's equation, simple Jacobi or Gauss-Seidel will typically suffice as an efficient smoother. These techniques effectively reduce the high-frequency components of the error and make it possible for a coarse grid to provide a meaningful correction to a fine grid. This property is fundamentally important for the efficiency of the geometrically hierarchical approach. Unfortunately, the equations of nearly incompressible linear elasticity with augmented pressure require more care than the comparably simplistic discrete Poisson equation (cf. Chapter 4). Although our system is not symmetric positive definite, we can reformulate the equations in a more convenient form as in [ZST10] to design an effective geometric multigrid smoother. Indeed, our change of variables leads to an approximate block triangularization of the discrete system with each diagonal block being a symmetric semi-definite discretization of the Laplacian. Our smoother is then constructed to be an application of the Gauss-Seidel relaxation on each block.



(a) An example boundary band of pressure cells and boundary variables using 3×3 blocks of cells centered at each cell that intersects the boundary.

(b) An example of distributive pressure cells and variables relaxed using distributive relaxation. This example region is defined by excluding 1×1 blocks of cells (i.e., a single cell) centered at each cell intersecting the boundary.

Figure 5.5: Boundary band and distributive region.

5.4.2.1 Approximate distributive relaxation

We follow the idea in [ZST10] and develop a *distributive relaxation scheme*. We apply the following transformation on the continuous variable $\tilde{\mathbf{u}} := (\mathbf{u}\ p)$:

$$\begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{I} & -\nabla \\ \nabla^t & -2\Delta \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ q \end{pmatrix} \quad \text{or} \quad \tilde{\mathbf{u}} = \tilde{\mathbf{M}}\tilde{\mathbf{v}}; \tag{5.16}$$

substituting into (5.2) yields the auxiliary system

$$\begin{pmatrix} \mu\Delta\mathbf{I} & 0 \\ \mu(1+\mu/\lambda)\nabla^t & -\mu(1+2\mu/\lambda)\Delta \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ q \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} \quad \text{or} \quad \tilde{\mathbf{A}}\tilde{\mathbf{M}}\tilde{\mathbf{v}} = \tilde{\mathbf{f}} \tag{5.17}$$

for some auxiliary variable $\tilde{\mathbf{v}} := (\mathbf{v}\ q)$. This auxiliary system is a block lower triangular system and can be solved via forward substitution: first solve the $\mathbf{v}$ equations (which don't involve q), then solve for q given the previously solved $\mathbf{v}$. Moreover, with an intelligent discretization, the same triangulation can be realized on the discretized system, i.e., $\tilde{A}\tilde{M}$ is also a block lower triangular linear system [ZST10].

Since each of the diagonal blocks of the discrete auxiliary system is a discretization of the Laplacian operator, we can relax the whole system using a Gauss-Seidel relaxation on each component of the $\vec{v}$ degrees of freedom followed by a (3^{rd}) Gauss-Seidel relaxation on the

$\vec{q}$ degrees of freedom and achieve the same smoothing efficiency as Gauss-Seidel relaxation simply applied to Poisson's equation. Note that given any $\vec{v}$ and $\vec{q}$, one can reconstruct $\vec{u}$ and $\vec{p}$ via the discrete form of (5.16).

In practice, we never explicitly construct $\tilde{v} := (\vec{v}\ \vec{q})$. In a Gauss-Seidel relaxation applied to $\tilde{v}$, we iteratively solve for local corrections $\tilde{v}_i \leftarrow \tilde{v}_i + \delta\tilde{e}_i$, such that the local residual $\tilde{r}_i := (\tilde{f} - \tilde{A}\tilde{M}\tilde{v})_i$ is zeroed out. From this, we see that $\delta = \tilde{r}_i/(\tilde{A}\tilde{M})_{ii}$. Such local corrections to $\tilde{v}$ induce local, *distributive* corrections to $\tilde{u}$ as well via the discrete form of (5.16): $\tilde{u}_i \leftarrow \tilde{u}_i + \delta\tilde{M}\tilde{e}_i$. This analysis is encapsulated in Algorithm 5.3.

Algorithm 5.3 Distributive relaxation.

function DistributiveSmooth $\left(\tilde{A}, \tilde{M}, \tilde{u}, \tilde{f}\right)$ **:**
  for $\vec{w} \in \{\vec{u}^1, \vec{u}^2, \vec{p}\}$ **do**
    for $i \in$ Lattice($\vec{w}$) **do**
      $r \leftarrow \tilde{f}_i - \tilde{A}\tilde{u}$
      $\delta \leftarrow r/(\tilde{A}\tilde{M})_{ii}$
      $\tilde{u}$ += $\delta\tilde{M}\tilde{e}_i$
    end for
  end for

For a staggered finite difference discretization, the triangularization of the discretized system can be effected by discretizing the transformation operator $\tilde{\mathbf{M}}$ in (5.16) using centered differences for the gradient and divergence operators and the standard 5-point stencil for the Laplacian operator [ZST10]. Unfortunately, when one uses a finite element discretization, there exists no discrete change of variables with the same sparsity that leads to an exact triangularization. Instead, we discretize the gradient operator in (5.16) using the stencils derived from a finite element method, i.e.,

$$\nabla \approx \frac{1}{\mu h^2} G^t = \begin{pmatrix} D_x \\ D_y \end{pmatrix},$$

which maps p degrees of freedom onto x and y degrees of freedom with the locations illustrated in Figure 5.3(right) and stencils as

$$D_x := \frac{1}{h}\begin{bmatrix} -1/8 & +1/8 \\ -3/4 & +3/4 \\ -1/8 & +1/8 \end{bmatrix}, \quad D_y := \frac{1}{h}\begin{bmatrix} +1/8 & +3/4 & +1/8 \\ -1/8 & -3/4 & -1/8 \end{bmatrix}.$$

Similarly, the Laplacian operator in (5.16) is discretized from a standard piecewise bilinear finite element discretization:

$$M_p := \frac{1}{h^2}\begin{bmatrix} +1/3 & +1/3 & +1/3 \\ +1/3 & -8/3 & +1/3 \\ +1/3 & +1/3 & +1/3 \end{bmatrix}.$$

Although the linear system $\tilde{A}\tilde{M}$ is not strictly block triangular, our numerical results in §5.5 show that the derived distributive relaxation is able to reduce the high-frequency error components efficiently.

5.4.2.2 High order defect correction

We can decrease the cost of our distributive relaxation via *high order defect correction*. To introduce the idea, suppose we wish to solve the linear system $L\vec{u} = \vec{f}$ for the unknown $\vec{u}$. In a *defect correction* scheme, we solve for a correction $\delta\vec{u}$ to a current approximation $\vec{u}$ via some alternate, typically easier system $L'\delta\vec{u} = \vec{f} - L\vec{u}$. For example, in a multigrid correction scheme with L a fine grid operator, L' (roughly) corresponds to the coarse grid operator. In a *high order* defect correction scheme, a lower order discretization alternate system is used to to solve for a higher order discretization correction. In our case, the lower order operator is based on a finite difference approximation while the high order operators corresponds to our finite element discretization described in §5.2.1. In other words, we solve the following correction equation:

$$\tilde{A}^{\mathrm{FD}}\delta\tilde{u} = \tilde{f} - \tilde{A}^{\mathrm{FEM}}\tilde{u}.$$

We use the staggered finite difference operator $\tilde{A}^{\mathrm{FD}}$ detailed in [ZST10]. Note that this operator is only lower order near the boundary; indeed, it is still second order in the interior. However, we still use this finite difference operator as the lower order operator in the defect correction scheme, even when focused on the domain interior.

One of the benefits we obtain from such an approximation is that we can use the distributive relaxation described above with an exact triangulation of the discretized system (5.17). To be specific, let us write the finite difference system as

$$\tilde{A}^{\mathrm{FD}}\tilde{u} = \begin{pmatrix} A_u^{\mathrm{FD}} & (G^{\mathrm{FD}})^t \\ G^{\mathrm{FD}} & D_p^{\mathrm{FD}} \end{pmatrix} \begin{pmatrix} \vec{u}^{\mathrm{FD}} \\ \vec{p}^{\mathrm{FD}} \end{pmatrix}, \quad \tilde{f}^{\mathrm{FD}} = \begin{pmatrix} \vec{f}^{\mathrm{FD}} \\ \vec{0} \end{pmatrix},$$

and scale finite element system by $1/h^2$ to match the scaling of the both the finite difference equation and the differential equation:

$$\frac{1}{h^2}\tilde{A}^{\mathrm{FEM}}\tilde{u} = \frac{1}{h^2}\begin{pmatrix} A_u^{\mathrm{FEM}} & (G^{\mathrm{FEM}})^t \\ G^{\mathrm{FEM}} & D_p^{\mathrm{FEM}} \end{pmatrix} \begin{pmatrix} \vec{u}^{\mathrm{FEM}} \\ \vec{p}^{\mathrm{FEM}} \end{pmatrix}, \quad \frac{1}{h^2}\tilde{f}^{\mathrm{FEM}} = \frac{1}{h^2}\begin{pmatrix} \vec{f}^{\mathrm{FEM}} \\ \vec{0} \end{pmatrix}.$$

(Note that $\tilde{A}^{\mathrm{FEM}}$ is precisely the operator $\tilde{A}$ defined in (5.7), and similarly for the other variables marked with $\cdot^{\mathrm{FEM}}$; we only add the superscript to emphasize the difference with the $\cdot^{\mathrm{FD}}$ variables.) We thus solve for a correction $\delta\tilde{u} = \tilde{M}^{\mathrm{FD}}\delta\tilde{v}$ via

$$\tilde{A}^{\mathrm{FD}}\tilde{M}^{\mathrm{FD}}\delta\tilde{v} = \frac{1}{h^2}\left(\tilde{f}^{\mathrm{FEM}} - \tilde{A}^{\mathrm{FEM}}\tilde{u}^{\mathrm{current}}\right).$$

This induces a sparser distributive relaxation scheme; the details are given in Algorithm 5.4.

5.4.2.3 Boundary relaxation

Neither of the distributive relaxation algorithms discussed in the previous subsubsections may be applied near the domain boundary as some variables in the distribution stencil may not even exist. Instead, we follow [ZST10] and temporarily build an unaugmented system

Algorithm 5.4 Distributive relaxation with *finite difference defect correction* (FDDC).

function DistributiveSmoothWithFDDC $\left(\tilde{A}^{\mathrm{FD}}, \tilde{M}^{\mathrm{FD}}, \tilde{A}^{\mathrm{FEM}}, \tilde{M}^{\mathrm{FEM}}, \tilde{u}, \tilde{f}^{\mathrm{FEM}}\right)$ **:**
for $\vec{w} \in \{\vec{u}^1, \vec{u}^2, \vec{p}\}$ **do**
  for $i \in \mathrm{Lattice}(\vec{w})$ **do**
    $r \leftarrow \tilde{f}_i^{\mathrm{FEM}} - \tilde{A}^{\mathrm{FEM}}\tilde{u}$
    $\delta \leftarrow r/(\tilde{A}^{\mathrm{FD}}\tilde{M}^{\mathrm{FD}})_{ii}$
    $\tilde{u}$ += $\delta\tilde{M}^{\mathrm{FD}}\tilde{e}_i$
  end for
end for

in the boundary band (e.g., see Figure 5.5(a)). In the case of Neumann boundary condition, we eliminate $\vec{p}$ from the augmented system (5.7) by left multiplying by

$$\tilde{U} := \begin{pmatrix} I & -G^t D_p^{-1} \\ 0 & I \end{pmatrix}$$

yielding

$$\tilde{U}\tilde{f} = \tilde{U}\tilde{A}\tilde{u} = \begin{pmatrix} A_u - G^t D_p^{-1} G & 0 \\ G & D_p \end{pmatrix} \begin{pmatrix} \vec{u} \\ \vec{p} \end{pmatrix}. \tag{5.18}$$

In the first equation for $\vec{u}$, the equation is symmetric and positive definite, and hence can be solved by using, e.g., Gauss-Seidel relaxation. This unaugmented system is a consistent discretization to the original PDE (5.2). Although Gauss-Seidel relaxation is not an efficient smoother for the unaugmented system if defined everywhere, for the purposes of boundary relaxation, we only build the unaugmented system temporarily, relax it within a very narrow boundary band, as in Figure 5.5, and temporarily freeze the interior degrees of freedom. The solution is strongly restricted by nearby interior values, so Gauss-Seidel relaxation is still efficient and stable. Typically, with about 5 to 10 boundary relaxation sweeps before and after each interior relaxation sweep, the boundary residual is reduced to as small as the interior residual. Once we have relaxed $\vec{u}$ sufficiently, we freeze $\vec{u}$ and substitute into the second equation of (5.18) to relax the pressure degrees of freedom.

5.4.2.4 Boundary relaxation with Dirichlet boundary conditions

In the case of Dirichlet boundary conditions, the system (5.13) is strongly indefinite and cannot be relaxed using, e.g., Gauss-Seidel. Alternative approaches such as Kaczmarz relaxation or box relaxation may be efficient smoothers; however, they have a high computational cost. Instead, we utilize the fundamental basis of the constraint matrix to solve $\vec{v}$ in the Z^tAZ system (5.15) and then reconstruct $\vec{u}$ in (5.13) via $\vec{u} := \vec{c} + Z\vec{v}$. Since Z^tAZ is symmetric positive definite (see Chapter 4, §4.2.3.1), one may confidently apply Gauss-Seidel relaxation; see Algorithm 5.5 and compare Chapter 4, §4.3.2.

In practice, a Gauss-Seidel iteration on (5.15) iteratively solves for a correction on each

Algorithm 5.5 Boundary relaxation with Dirichlet boundary conditions - $\vec{v}$.

$\vec{v} \leftarrow \vec{0}$
for $i = 1, \ldots, m$ **do**
  $\delta \leftarrow \vec{e}_i^t Z^t \left(\vec{f} - A\vec{c} - AZ\vec{v} \right) / A_{ii}$
  $\vec{v} \mathrel{+}= \delta \vec{e}_i$
end for

single degree of freedom by solving the scalar equation

$$\vec{e}_i^t Z^t AZ \left(\vec{v} + \delta \vec{e}_i \right) = \vec{e}_i^t Z^t \left(\vec{f} - A\vec{c} \right),$$

i.e.,

$$\begin{aligned} A_{ii}\delta &= \vec{e}_i^t Z^t \left(\vec{f} - A\vec{c} - AZ\vec{v} \right) \\ &= \vec{e}_i^t Z^t \left(\vec{f} - A\vec{u} \right), \end{aligned}$$

and then applying the correction $\vec{v} \leftarrow \vec{v} + \delta \vec{e}_i$. Equivalently, one may update the $\vec{u}$ degrees of freedom directly: $\vec{u} \leftarrow \vec{u} + \delta Z \vec{e}_i$; see Algorithm 5.6.

Algorithm 5.6 Boundary relaxation with Dirichlet boundary conditions - $\vec{u}$.

$\vec{u} \leftarrow \vec{c}$
for $i = 1, \ldots, m$ **do**
  $\delta \leftarrow \vec{e}_i^t Z^t \left(\vec{f} - A\vec{u} \right) / A_{ii}$
  $\vec{u} \mathrel{+}= \delta Z \vec{e}_i$
end for

5.4.3 Coarsening

In contrast to our smoothing operator discussed in the preceding subsections, our grid transfer operators are relatively straightforward. On the interior of the domain, we restrict fine grid residuals to the coarse grid by applying a restriction operator R with the stencils illustrated in Figure 5.6. We consider two kinds of prolongation operators for our numerical examples in §5.5.2. First, we consider simply the transpose of the restriction operator: $P_{\text{lo}} := 4R^t$. Second, we also consider piecewise bilinear interpolation for $\vec{u}$ in combination with the same pressure prolongation given by P_{lo}, which we denote as P_{hi}.

However, near the boundary, there is no guarantee that all dependencies of the coarse grid restriction stencils are legitimate fine grid degrees of freedom. Therefore, we truncate our restriction stencils to the actual degrees of freedom, which is equivalent to restricting zero residuals from fine grid vertices which are not degrees of freedom. Also, in the presence of Dirichlet boundary conditions, we cannot compute the components of the residual $\vec{r} =$

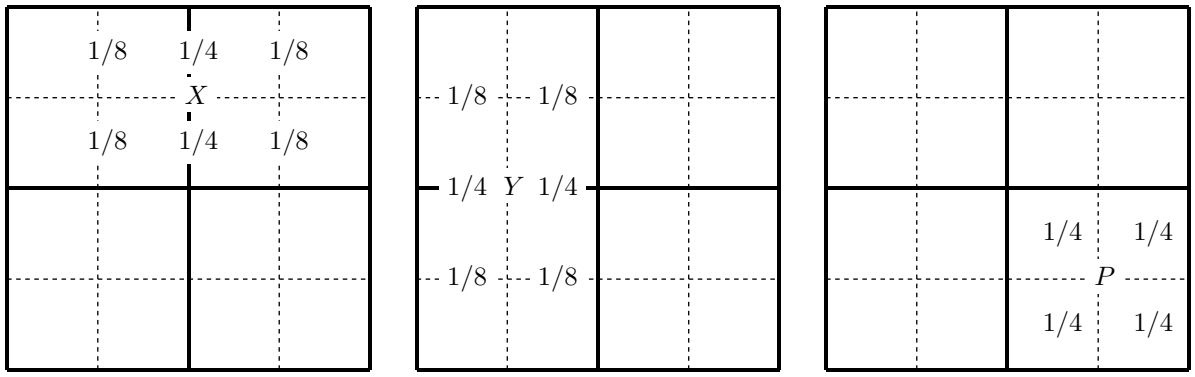


Figure 5.6: Stencils for the restriction operator R.

$\vec{f} - A\vec{u} - B^t\vec{\lambda}$ when $\vec{\lambda}$ components are nontrivially involved. In this case, we apply enough boundary relaxation sweeps to ensure these boundary residuals are smaller than interior residuals, then simply restrict zero boundary residual values for these equations. Now, normally, the right hand side to the coarse grid constraint system has been computed from the restriction of the residual of the fine grid constraint system. However, due to the fact that our solutions $\vec{u} = \vec{c} + Z\vec{v}$ always satisfy the Dirichlet boundary constraints exactly, the Dirichlet boundary condition on all coarse grids should be zero.

We implement prolongation distributively, i.e., we iterate over the coarse grid degrees of freedom and distribute a coarse grid scalar correction to all appropriate fine level degrees of freedom. Near the domain boundary this is equivalent to prolongating a zero correction from exterior coarse grid vertices, which we believe is reasonable. Notice that such a prolongation may shift the fine grid solution away from a fundamental basis solution. Hence we apply a projection onto the solution space after prolongation. We use the projection $\vec{u}' := \vec{c} + ZQ\vec{u}$ where $\vec{u}$ is the (possibly) shifted fine grid solution and Q is a projection which simply removes those components corresponding to independent degrees of freedom with respect to the aggregate constraint system.

5.5 Numerical Examples

We numerically investigate two aspects of our discretization: order of convergence and multigrid performance. In this section, we apply our method on various domains with Neumann or Dirichlet boundary conditions and with a wide range of Poisson's ratios. We considered three deformations defined on three geometric domains:

1. **Keyhole domain.** A Keyhole domain enclosed by a smooth curve connecting 8

tangential circles with centers

$$\begin{aligned}
\mathbf{c}_1 &= (0.2500, 0.2500); & \mathbf{s}_1 &= (0.5000, 0.6875);\\
\mathbf{c}_2 &= (0.7500, 0.2500); & \mathbf{s}_2 &= (0.5000, 0.3125);\\
\mathbf{c}_3 &= (0.2500, 0.7500); & \mathbf{s}_3 &= (0.3125, 0.5000);\\
\mathbf{c}_4 &= (0.7500, 0.7500); & \mathbf{s}_4 &= (0.6875, 0.5000);
\end{aligned}$$

and radius 0.2 for the "$\mathbf{c}$" circles and $r_s = \sqrt{17}/4 - 0.2$ for the "$\mathbf{s}$" circles. The radius r_s is chosen such that the circle curves are tangential and hence generate a smooth boundary. The boundary of the keyhole domain can also be represented by the zero isocontour of the level set function

$$\varphi(\mathbf{x}) := \max\left\{\min\left\{\alpha\left(\mathbf{x}, \mathbf{0}, r_0\right), \min_i\left\{\alpha\left(\mathbf{x}, \mathbf{c}_i, 0.2\right)\right\}\right\}, -\min_i\left\{\alpha\left(\mathbf{x}, \mathbf{s}_i, r_s\right)\right\}\right\}$$

where

$$\alpha\left(\mathbf{x}, \mathbf{x}_0, r\right) := \|\mathbf{x} - \mathbf{x}_0\| - r, \quad r_0 := \left\|\frac{0.2}{\sqrt{17}}(4, 1) - (0.25, 0.25)\right\|.$$

A constant divergence deformation is considered, giving the exact boundary conditions and the exact solution for the purpose of error computation:

$$\begin{aligned}
\phi_1(x, y) &= 2x + \frac{1}{2}\cos \pi x \sin \pi y,\\
\phi_2(x, y) &= 2y - \frac{1}{2}\sin \pi x \cos \pi y.
\end{aligned}$$

2. **Flower domain.** A flower-shaped domain with inner radius 0.2 and outer radius 0.4. We represent the boundary of this domain by the zero isocontour of the level set function

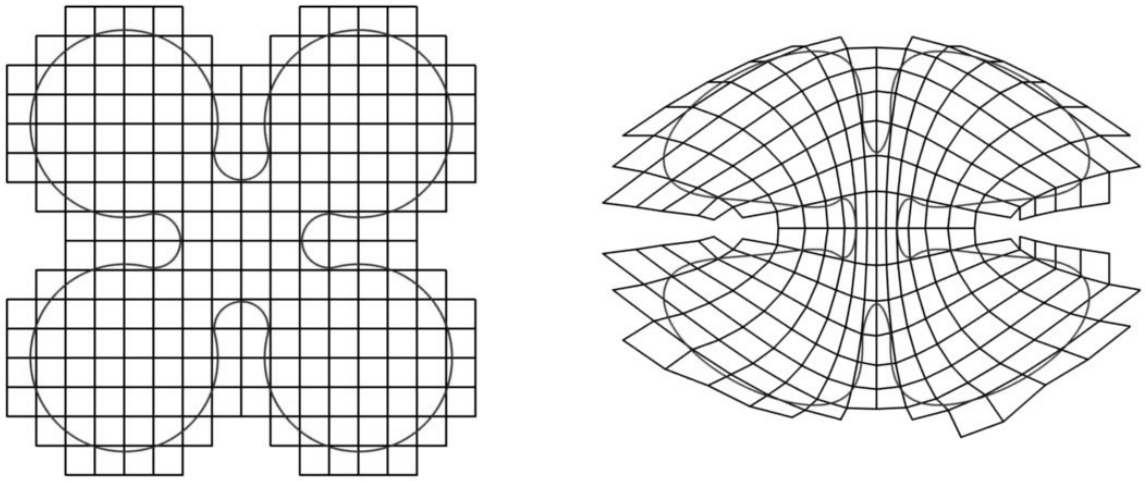
$$\varphi(\mathbf{x}) := \alpha\left(\mathbf{x}, \mathbf{0.5}, 0.3 + 0.1\cos 5\theta\right),$$

where $\mathbf{0.5} := (0.5, 0.5)$ and θ is the argument of $\mathbf{x}$. We use the following deformation with spatially varying divergence:

$$\begin{aligned}
\phi_1(x, y) &:= \frac{2x}{\sqrt{\pi}}\cos\frac{\pi}{2}y,\\
\phi_2(x, y) &:= \frac{2x}{\sqrt{\pi}}\sin\frac{\pi}{2}y.
\end{aligned}$$

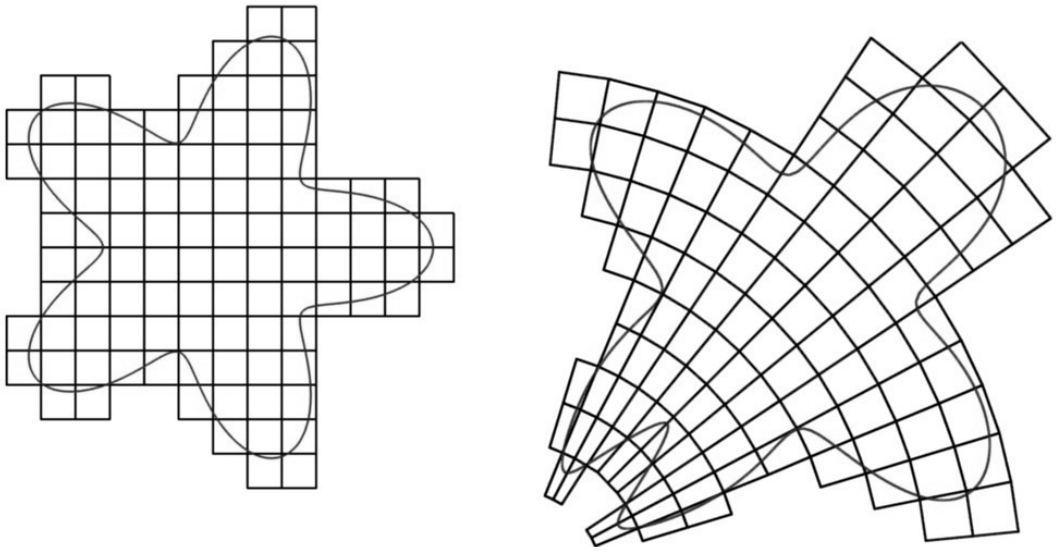
3. **Spiral domain** We represent the boundary of the spiral domain as the zero isocontour of the level set function

$$\varphi(\mathbf{x}) := r(\mathbf{y}) - \left(0.33 + 0.08\cos 5\theta(\mathbf{y})^{1/3}\right)$$



(a) Undeformed. (b) Deformed.

Figure 5.7: Keyhole domain.

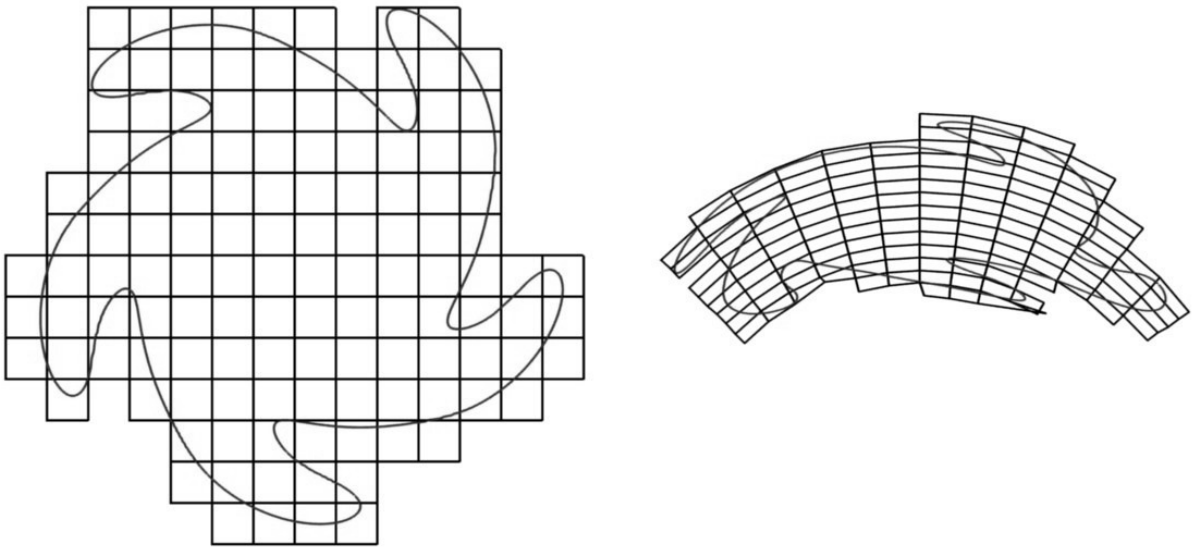


(a) Undeformed. (b) Deformed.

Figure 5.8: Flower domain.

where $\mathbf{y} = \mathbf{y}(\mathbf{x})$ is $\mathbf{x} - (0.5, 0.5)$ rotated around (0.5,0.5) by $\theta = 14\,(2r(\mathbf{x}))^{1/6}$. We use the deformation

$$\phi_1(x, y) = \left(\frac{1}{2}x + \frac{1}{2}\right) \cos\left(\frac{\pi}{6} + \frac{2}{3}\pi y\right),$$
$$\phi_2(x, y) = \left(\frac{1}{2}x + \frac{1}{2}\right) \sin\left(\frac{\pi}{6} + \frac{2}{3}\pi y\right).$$

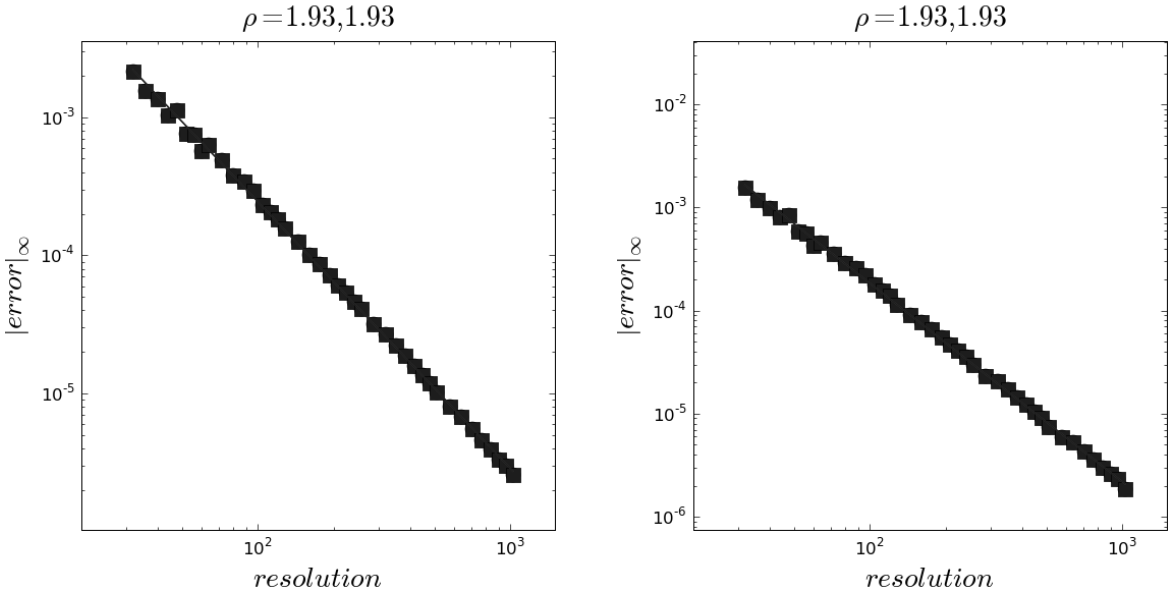


(a) Undeformed. (b) Deformed.

Figure 5.9: Spiral domain.

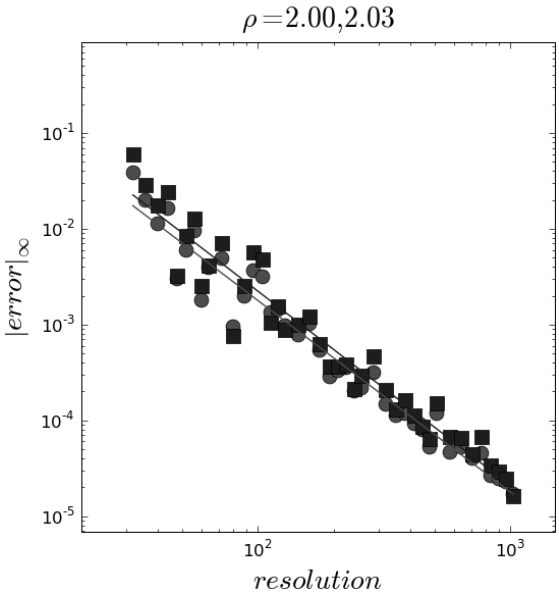
5.5.1 Convergence

We embed all our testing domains in a regular square Cartesian grid over $[0, 1] \times [0, 1]$ with resolutions ranging from 32×32 to 1024×1024. We use a log-log plot of the error ($\log_2 \|\vec{u}^{\text{exact}} - \vec{u}^{\text{approx}}\|_\infty$) versus the resolution ($\log_2$ resolution) to estimate the order of convergence via a least squares linear regression. We remove any solution null space in an example with entirely Neumann boundary conditions by enforcing a non-embedded Dirichlet condition on all degrees of freedom within the domain $[7/16, 9/16] \times [7/16, 9/16]$. We observe second order convergence over all three example domains for both Neumann and Dirichlet boundary conditions and for a wide range of material parameters, including in the nearly incompressible regime (see Figures 5.10, 5.11, and 5.12).

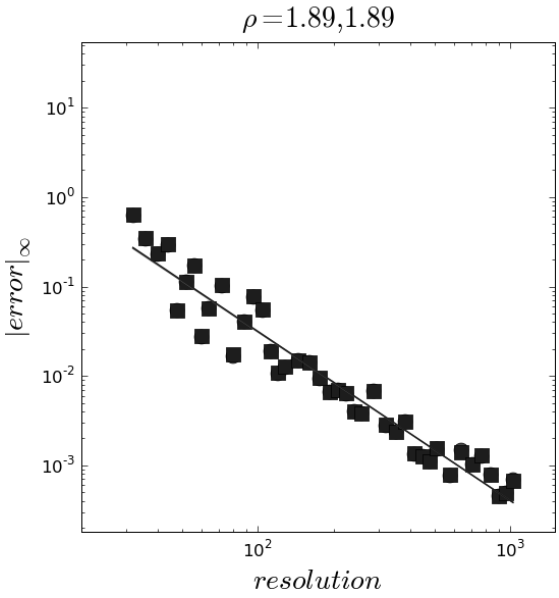


(a) Neumann BCs; $\nu = 0.3$

(b) Neumann BCs; $\nu = 0.49$

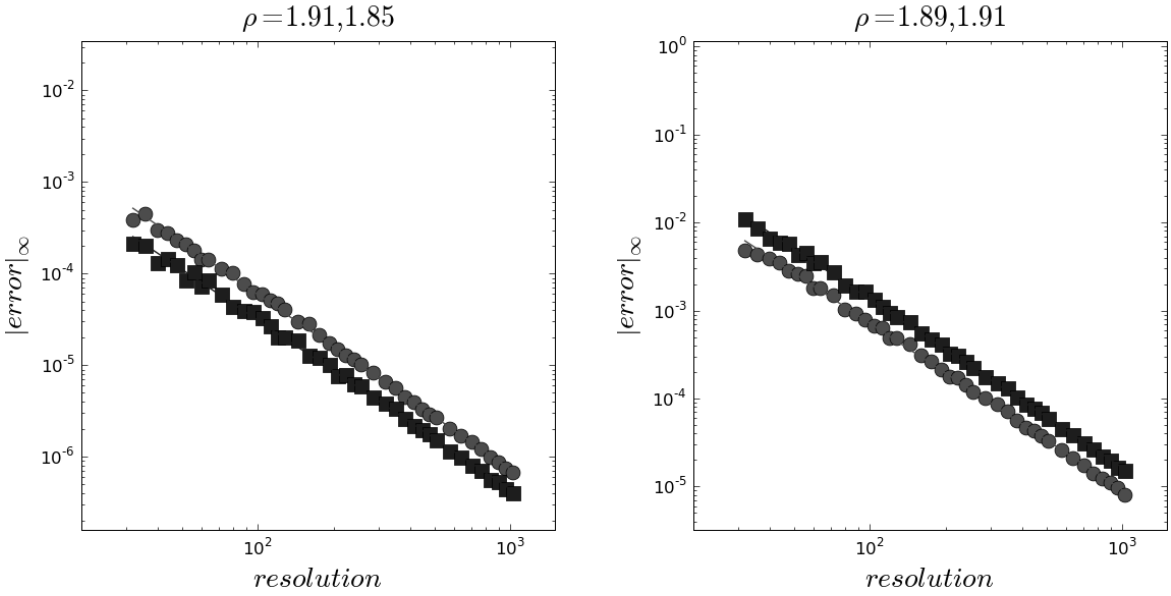


(c) Dirichlet BCs; $\nu = 0.3$



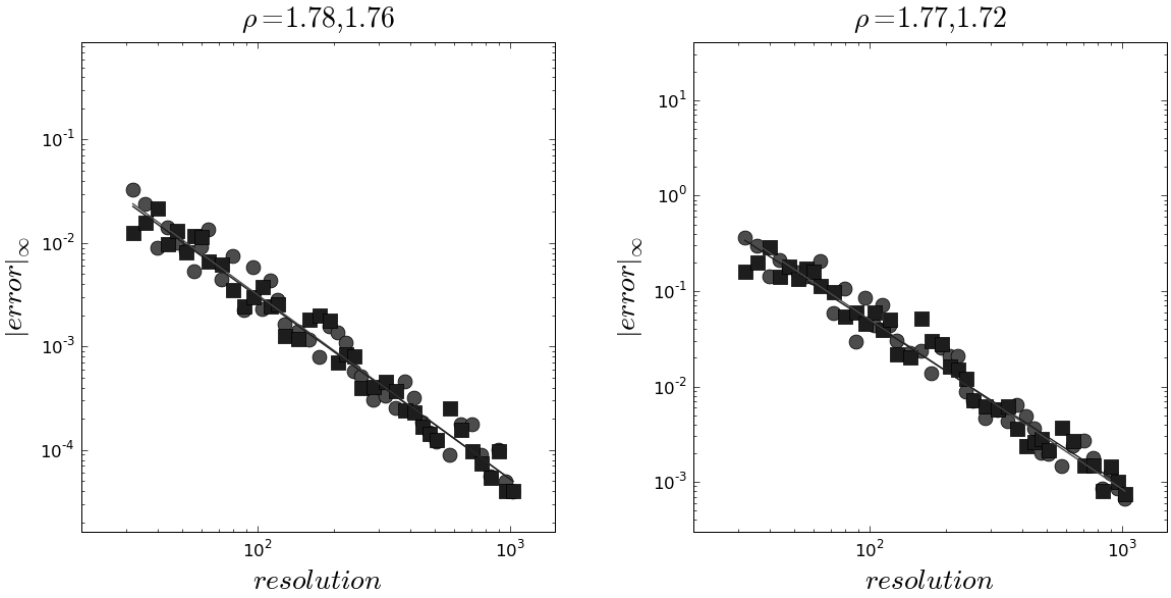
(d) Dirichlet BCs; $\nu = 0.49$

Figure 5.10: Log-log plots of the L^∞-error of the approximate solution versus the grid resolution, and the corresponding computed orders of convergence, ρ, for the keyhole domain. We stipulate an embedded Neumann boundary condition for the top ((a), (b)) examples and an embedded Dirichlet boundary condition for the bottom ((c), (d)) examples. The left ((a), (c)) examples have a Poisson's ratio of $\nu = 0.3$, while the right ((b), (d)) use a Poisson's ratio of $\nu = 0.49$, very close to the incompressible limit. Square (circle) markers in the above error plots correspond to errors in the x (y) component.



(a) Neumann BCs; $\nu = 0.3$

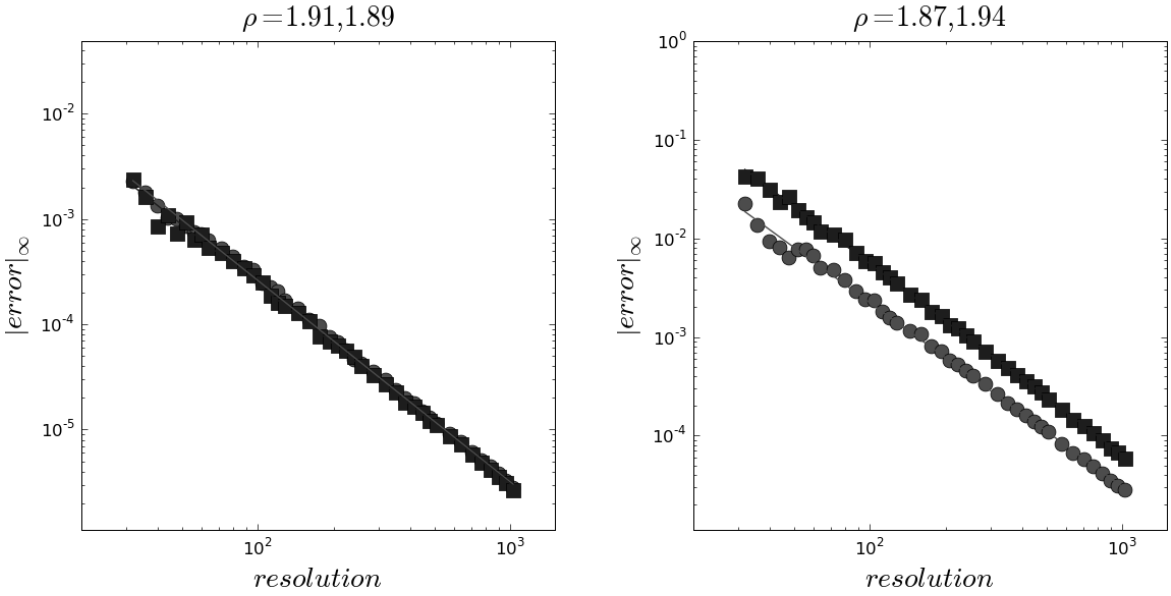
(b) Neumann BCs; $\nu = 0.49$



(c) Dirichlet BCs; $\nu = 0.3$

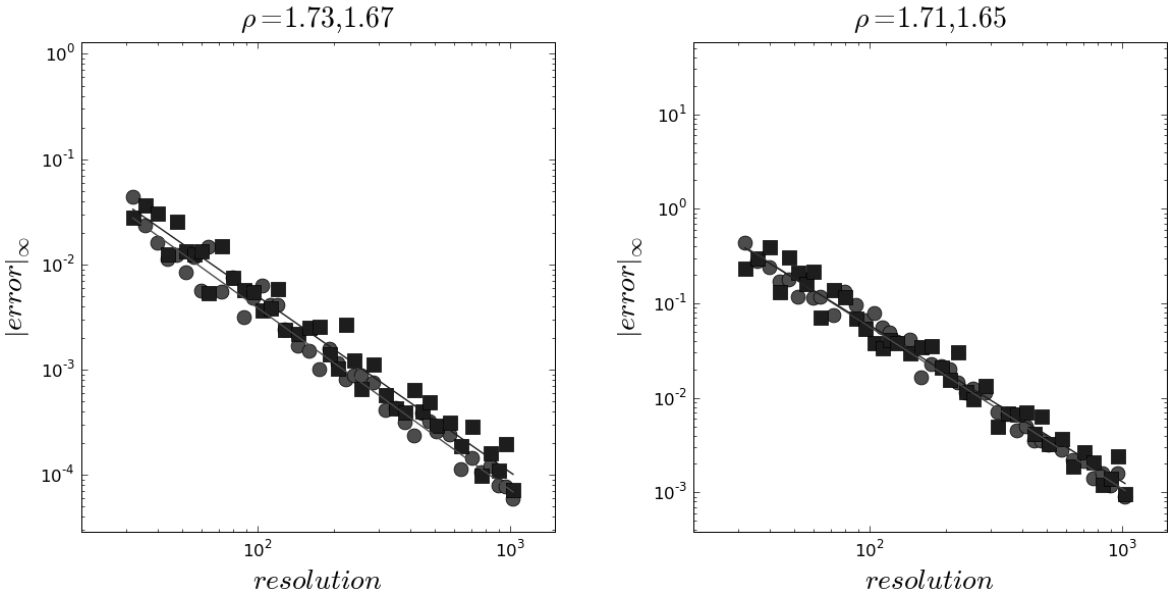
(d) Dirichlet BCs; $\nu = 0.49$

Figure 5.11: Log-log plots of the L^∞-error of the approximate solution versus the grid resolution, and the corresponding computed orders of convergence, ρ, for the flower domain. We stipulate an embedded Neumann boundary condition for the top ((a), (b)) examples and an embedded Dirichlet boundary condition for the bottom ((c), (d)) examples. The left ((a), (c)) examples have a Poisson's ratio of $\nu = 0.3$, while the right ((b), (d)) use a Poisson's ratio of $\nu = 0.49$, very close to the incompressible limit. Square (circle) markers in the above error plots correspond to errors in the x (y) component.



(a) Neumann BCs; $\nu = 0.3$

(b) Neumann BCs; $\nu = 0.49$



(c) Dirichlet BCs; $\nu = 0.3$

(d) Dirichlet BCs; $\nu = 0.49$

Figure 5.12: Log-log plots of the L^∞-error of the approximate solution versus the grid resolution, and the corresponding computed orders of convergence, ρ, for the spiral domain. We stipulate an embedded Neumann boundary condition for the top ((a), (b)) examples and an embedded Dirichlet boundary condition for the bottom ((c), (d)) examples. The left ((a), (c)) examples have a Poisson's ratio of $\nu = 0.3$, while the right ((b), (d)) use a Poisson's ratio of $\nu = 0.49$, very close to the incompressible limit. Square (circle) markers in the above error plots correspond to errors in the x (y) component.

5.5.2 Multigrid

We now numerically investigate the efficiency of our multigrid framework from §5.4. First, we consider a periodic boundary condition problem defined on $[0,1] \times [0,1]$ with exact solution

$$\phi_1(x,y) = \sin 2\pi x + \cos 2\pi y,$$
$$\phi_2(x,y) = \cos 2\pi x + \sin 2\pi y.$$

Although periodic boundary conditions will not appear in practical elasticity problems, we initially consider periodic boundary conditions to evaluate various algorithm parameters while avoiding complications due to boundary relaxation.

We first consider a fixed resolution of 128×128 and compare finite element distributive relaxation (described §5.4.2.1) and the distributive relaxation derived from the finite difference defect correction (FDDC) (described in §5.4.2.2) as the interior relaxations. We also compare the prolongation operators P_{lo} (transpose of restriction) and P_{hi} (bilinear interpolation) (both described in §5.4.3).

While relatively compressible problems exhibit stablized convergence rates no larger than 0.3 for a multigrid V-$(1,1)$ cycle with all discussed combinations of prolongation and distributive relaxation, we focus on the harder nearly incompressible case with Poisson's ratio $\nu = 0.49$ and investigate both V-$(1,1)$ cycle and W-$(1,1)$ cycle convergence. As shown in Table 5.1, both finite element distributive relaxation and FDDC distributive relaxation give convergence rates less than 0.5 with a multigrid V-$(1,1)$ cycle. Although the FDDC distributive relaxation generally gives a slower convergence than the finite element distributive relaxation for the V-$(1,1)$ cycle, combining it with bilinear interpolation prolongation or using a W-$(1,1)$ cycle brings the convergence rate down to 0.23.

We now investigate the multigrid cycle convergence rates at various resolutions by fixing the multigrid cycle to a V-$(1,1)$ cycle with finite element distributive relaxation and low order prolongation P_{lo}. We observe a consistent convergence rate across a spectrum of resolutions from 32×32 to 1024×1024; see Figure 5.13.

While all combinations of parameters discussed thus far give nice convergence rates for periodic boundary conditions, convergence rates in the presence of embedded Neumann and Dirichlet boundary conditions varies; the main bottleneck is the efficacy of the boundary relaxation. Therefore, a W-$(1,1)$ cycle provides little if any advantage over a V-$(1,1)$ cycle; similarly, our two prolongation operators under consideration, P_{lo} and P_{hi}, yield very similar convergence rates, everything else equal. The choice of interior relaxation seems to be the most differentiating characteristic when embedded boundary conditions are present; see Table 5.1.

This time with embedded boundary conditions, we again investigate the multigrid cycle convergence rate over a variety of resolutions. Since, in the presence of embedded boundary conditions, the multigrid cycles and prolongation operators discussed previously yield largely similar convergence rates, we only give results for a V-$(1,1)$ cycle with prolongation operator P_{lo}. Further, motivated by the results in Table 5.1, we use finite element distributive relaxation in the interior. As shown in Figures 5.14 and 5.15, we observe a consistent convergence

BC	relaxation	MG cycle	P_{hi}	P_{lo}
periodic	FD	V-(1, 1)	0.24	0.42
	FD	W-(1, 1)	0.23	0.25
	FEM	V-(1, 1)	0.13	0.24
	FEM	W-(1, 1)	0.13	0.30
Dirichlet	FD	V-(1, 1)	0.72	0.72
	FD	W-(1, 1)	0.72	0.72
	FEM	V-(1, 1)	0.37	0.36
	FEM	W-(1, 1)	0.42	0.42
Neumann	FD	V-(1, 1)	0.70	0.70
	FD	W-(1, 1)	0.68	0.68
	FEM	V-(1, 1)	0.50	0.50
	FEM	W-(1, 1)	0.35	0.35

Table 5.1: Asymptotic multigrid cycle convergence rates for different combinations of boundary conditions, distributive relaxation ("FEM" refers to the finite element distributive relaxation described in §5.4.2.1; "FD" refers to the distributive relaxation based on the finite difference defect correction described in §5.4.2.2), and prolongation (P_{lo} and P_{hi}; see §5.4.3). For these results, we used the flower domain at resolution 128×128 with Poisson's ratioa $\nu = 0.49$.

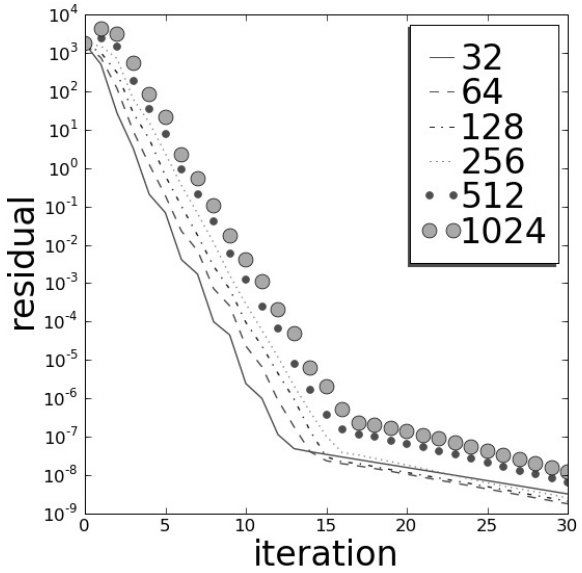
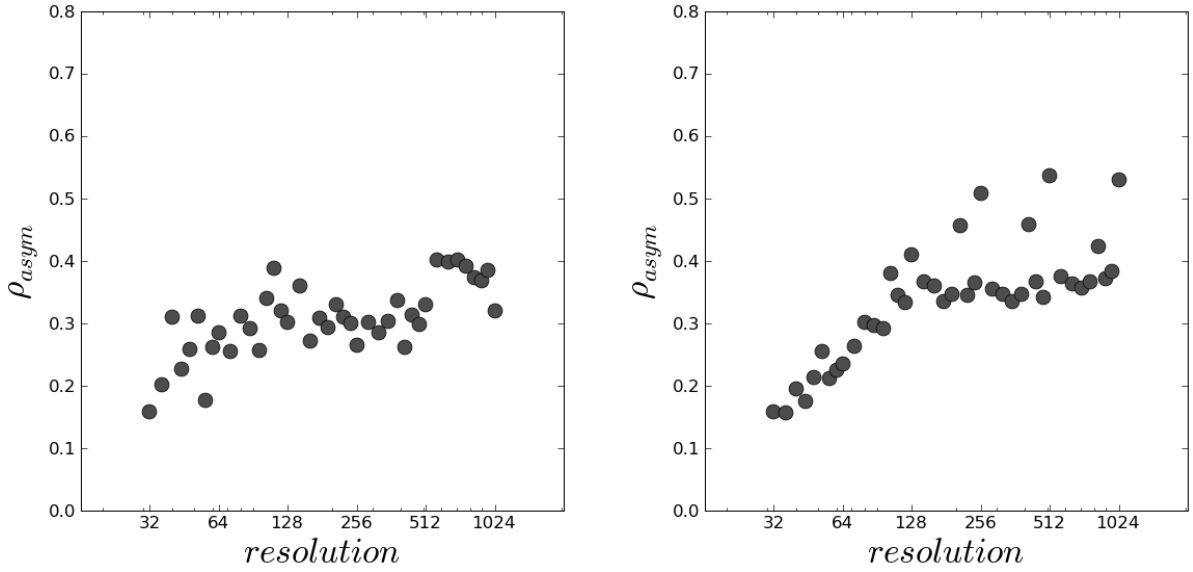


Figure 5.13: Multigrid V-(1, 1) cycle convergence for a variety of resolutions between 32×32 and 1024×1024 ($\nu = 0.49$, periodic boundary conditions, finite element distributive relaxation, and low order prolongation P_{lo}).

rate over a spectrum of resolutions from 32×32 to 1024×1024.



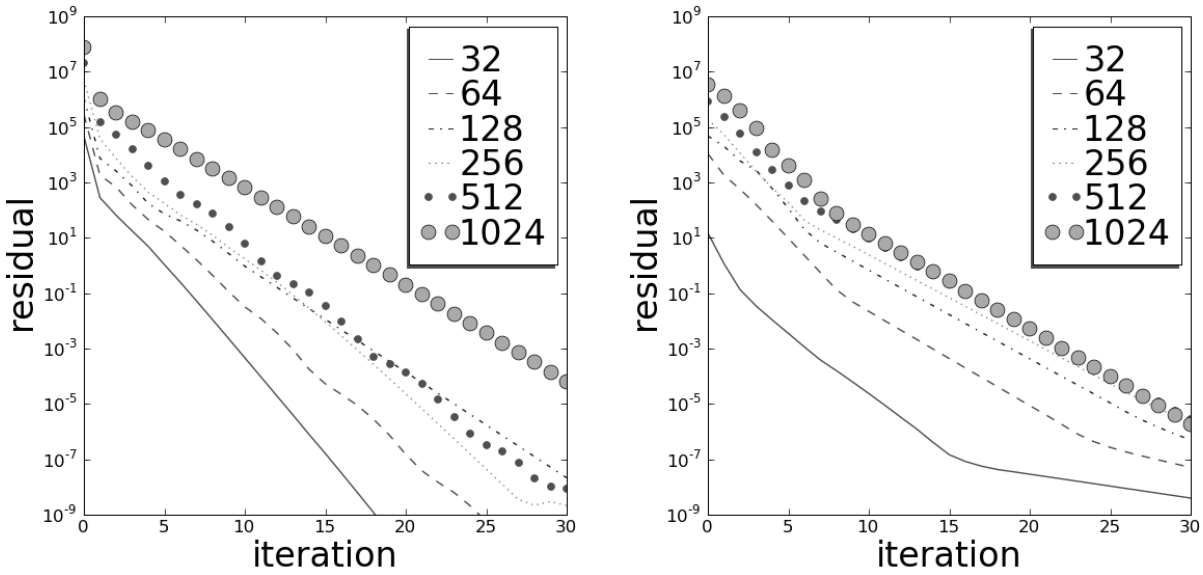
(a) Dirichlet BCs

(b) Neumann BCs

Figure 5.14: Multigrid V-$(1,1)$ cycle convergence rates at various resolutions from 32×32 to 1024×1024 ($\nu = 0.49$).

5.6 Discussion, Conclusion, and Future Work

We developed a second order mixed finite element discretization for the equilibrium equations of linear elasticity practical over all material parameters from compressible to nearly incompressible. We additionally formulated a geometric multigrid framework to solve the linear system derived from the discretization. Utilizing an approximated distributive relaxation, we achieve a fast and parameter-independent multigrid cycle convergence rate in the absence of embedded boundary conditions. With the addition of embedded boundary conditions, we can still maintain good multigrid cycle convergence rates with only a small number of boundary relaxations. However, we were unable to reproduce the optimum convergence rates observed with periodic boundary conditions. Thus, future research may investigate a more efficient boundary smoother that avoids unaugmentation. We may also be interested in extending our methods to the Stokes equation.



(a) Dirichlet BCs

(b) Neumann BCs

Figure 5.15: Residual norm reduction as a function of iteration number for a multigrid V-$(1, 1)$ cycle at various resolutions from 32×32 to 1024×1024 ($\nu = 0.49$).

APPENDIX A

Quadrature

In Chapter 4, §4.2.1 we mention the use of Gaussian quadrature rules to compute surface integrals over triangles of polynomial integrands. For convenience, we reproduce the triangle Gaussian quadrature rules of various orders from [Cow73] in Table A.1. For the quadrature points with multiplicity 3, the coordinates should be permuted to give 3 total symmetrically distributed quadrature points, all with the same given quadrature weight. For example, to integrate a cubic polynomial $p(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}$ over a triangle T with vertices $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\} \subset \mathbb{R}^3$, one would use the order 3 quadrature rule from Table A.1, which manifests itself as

$$\begin{aligned}\int_T p(\mathbf{x})d\mathbf{x} = \text{ area}(T) \Bigg(&-\frac{27}{48}\tilde{p}\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right) \\ &+ \frac{25}{48}\left(\tilde{p}\left(\frac{1}{5},\frac{1}{5},\frac{3}{5}\right) + \tilde{p}\left(\frac{1}{5},\frac{3}{5},\frac{1}{5}\right) + \tilde{p}\left(\frac{3}{5},\frac{1}{5},\frac{1}{5}\right)\right)\Bigg),\end{aligned} \tag{A.1}$$

where $\tilde{p}\left(\alpha_1, \alpha_2, \alpha_3\right) = p\left(\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2 + \alpha_3\mathbf{x}_3\right)$. Note how the multiplicity 3 quadrature point with barycentric coordinates $(1/5, 1/5, 3/5)$ in Table A.1 represents all of the latter 3 quadrature points in (A.1) via permutation of the coordinates.

order	mult.	weight	barycentric coordinates
1	1	1	$(1/3, 1/3, 1/3)$
2	3	$1/3$	$(1/6, 1/6, 2/3)$
3	1	$-27/48$	$(1/3, 1/3, 1/3)$
	3	$25/48$	$(1/5, 1/5, 3/5)$
4	3	0.109951743655322	$(0.091576213509771, 0.0915\ldots, 0.816847572980459)$
	3	0.223381589678011	$(0.108103018168070, 0.445948490915965, 0.4459\ldots)$
5	1	$9/40$	$(1/3, 1/3, 1/3)$
	3	0.125939180544827	$(0.101286507323456, 0.1012\ldots, 0.797426985353087)$
	3	0.132394152788506	$(0.059715871789770, 0.470142064105115, 0.4701\ldots)$

Table A.1: Triangle Gaussian quadrature rules of order 1 through 5, as given in [Cow73]. [Some repeated barycentric coordinates have been abbreviated with "..." for formatting purposes.]

APPENDIX B

Cell Averages

The Poisson discretizations described in Chapter 4, §4.2 require computing cell averages $\overline{\beta}$, $\overline{f}$, and $\overline{q}$ of β, f, and q, respectively. One can use any of a variety of techniques to compute these averages, and one's choice would likely depend upon whether one has β, f, and q immediately defined pointwise at grid vertices; pointwise at grid edge, face, or cell centers; or analytically throughout the domain, domain boundary, or interface.

We used evaluations of β and f at grid vertices to compute their cell averages. For non-boundary and non-interfacial grid cells, the cell average amounts to a straightforward, equal-weighted average of the values at the 8 grid vertices of the cell. For boundary and interfacial grid cells, we used trilinear interpolation to compute the cell average. For example, in the domain discretizations, we compute the cell average of β over grid cell $c_k \in \mathcal{C}^h_{\partial\Omega}$ as

$$\overline{\beta} := \frac{\int_{c_k \cap \Omega} \beta d\mathbf{x}}{\int_{c_k \cap \Omega} d\mathbf{x}} \approx \frac{\sum_{i \in \mathcal{N}^h_{c_k}} \beta_i \int_{c_k \cap \Omega} N_i d\mathbf{x}}{\int_{c_k \cap \Omega} d\mathbf{x}},$$

where, as introduced in §4.2.2, $\left\{N_i : i \in \mathcal{N}^h_{c_k}\right\}$ denotes the set of trilinear basis functions associated to the 8 grid vertices of c_k. Note that all integrands remaining in the right-most expression are polynomials, hence the integrals may be evaluated as described in §4.2.1. We compute $\overline{f}$, as well as cell averages in embedded interface discretizations, in a completely analogous fashion.

For embedded Neumann discretizations, to simplify implementation, we assume that $q \equiv \beta \nabla u \cdot \hat{\mathbf{n}}$ is available everywhere along the polyhedral representation of $\partial\Omega$. We use the second order quadrature rule from Table A.1 in Appendix A over each polygon of $\mathcal{P}^{c_k}_{\partial\Omega}$ (where $c_k \in \mathcal{C}^h_{\partial\Omega}$) to approximate $\overline{q}$:

$$\overline{q} := \frac{\int_{c_k \cap \partial\Omega} q d\mathbf{S}(\mathbf{x})}{\int_{c_k \cap \partial\Omega} d\mathbf{S}(\mathbf{x})} \approx \frac{\sum_{g \in \mathcal{P}^{c_k}_{\partial\Omega}} \int_g q d\mathbf{S}(\mathbf{x})}{\sum_{g \in \mathcal{P}^{c_k}_{\partial\Omega}} \text{area}(g)}.$$

APPENDIX C

Double-Wide Constraint Conditioning

In the context of the constraint aggregation algorithm for the discretization of Dirichlet boundary conditions and interfacial jump conditions for Poisson's equation, as mentioned in Chapter 4, §4.2.3.2, we found that the double-wide constraints introduced in [BBZ10] present significant conditioning issues in 3 dimensions that do not exist in 2 dimensions. Table C.1 shows the condition numbers and the number of conjugate gradient solve iterations for the Z^tAZ matrices resulting from the discretization of a simple Dirichlet problem and from the discretization of a similarly simple interface problem using each of two alternate discretizations Λ^h of the Lagrange multiplier space Λ: Λ_2^h, which corresponds to the double-wide constraints; and Λ_a^h, which corresponds to the aggregate constraints constructed via the algorithm described in §4.2.3.2. We calculated these statistics using PETSc in exactly the same way as for Table 4.1. This includes applying Jacobi preconditioning and then solving with (incomplete Cholesky-preconditioned) conjugate gradient to a relative residual norm of 2.3×10^{-13} of the Jacobi preconditioned system. Clearly, the conditioning of the Z^tAZ system arising from the double-wide constraints is several orders of magnitude worse than that arising from the constraint aggregation algorithm described in §4.2.3.2.

Test case	cond. # (no ICC)	cond. # (w/ICC)	# CG iter.	# PCG iter.
Dirichlet, $\Lambda^h = \Lambda_2^h$	3.7×10^{12}	1.1×10^{12}	59846	61568
Interface, $\Lambda^h = \Lambda_2^h$	4.4×10^{12}	1.4×10^{13}	97061	80225
Dirichlet, $\Lambda^h = \Lambda_a^h$	9.3×10^{2}	2.3×10^{1}	200	44
Interface, $\Lambda^h = \Lambda_a^h$	3.9×10^{3}	4.1×10^{1}	494	61

Table C.1: Condition numbers and (preconditioned) conjugate gradient ((P)CG) solve iterations, both with and without Incomplete Cholesky (ICC) preconditioning, for the Z^tAZ system arising from the discretization of a Dirichlet and from the discretization of an interface problem at grid resolution $32 \times 32 \times 32$. The Dirichlet problem has $\Omega = \{\mathbf{x} : |\mathbf{x}| \leq 0.8\}$ and $\beta \equiv 1$; the interface problem has $\Gamma = \{\mathbf{x} : |\mathbf{x}| = 0.8\}$ and $(\beta^-, \beta^+) \equiv (1, 2)$.

References

[AB05] Pedro M. A. Areias and Ted Belytschko. "Analysis of three-dimensional crack initiation and propagation using the extended finite element method." *International Journal for Numerical Methods in Engineering*, **63**(5):760–788, June 2005.

[AB06] P. Areias and T. Belytschko. "A comment on the article "A finite element method for simulation of strong and weak discontinuities in solid mechanics" by A. Hansbo and P. Hansbo [Comput. Methods Appl. Mech. Engrg. 193 (2004) 3523-3540]." *Computer Methods in Applied Mechanics and Engineering*, **195**(9–12):1275–1276, February 2006.

[ABC97] Ann S. Almgren, John B. Bell, Phillip Colella, and Tyler Marthaler. "A Cartesian grid projection method for the incompressible euler equations in complex geometries." *SIAM J. Sci. Comput.*, **18**:1289–1309, September 1997.

[AC04] Loyce Adams and Timothy P. Chartier. "New Geometric Immersed Interface Multigrid Solvers." *SIAM J. Sci. Comput.*, **25**:1516–1533, May 2004.

[AC05] Loyce Adams and Timothy P. Chartier. "A comparison of algebraic multigrid and geometric immersed interface multigrid methods for interface problems." *SIAM J. Sci. Comput.*, **26**:762–784, March 2005.

[AH08] Yazid Abdelaziz and Abdelmadjid Hamouine. "A survey of the extended finite element." *Computers & Structures*, **86**(1112):1141 – 1151, 2008.

[AJT04] Grégoire Allaire, François Jouve, and Anca-Maria Toader. "Structural optimization using sensitivity analysis and a level-set method." *J. Comput. Phys.*, **194**(1):363–393, February 2004.

[AL02] Loyce Adams and Zhilin Li. "The Immersed Interface/Multigrid Methods for Interface Problems." *SIAM J. Sci. Comput.*, **24**:463–479, February 2002.

[And05] T.L. Anderson. *Fracture Mechanics: Fundamentals and Applications*. Taylor & Francis, 2005.

[AP99] Owe Axelsson and Alexander Padiy. "On a robust and scalable linear elasticity solver based on a saddle point formulation." *International Journal for Numerical Methods in Engineering*, **44**(6):801–818, February 1999.

[APN07] J. L. Asferg, P. N. Poulsen, and L. O. Nielsen. "A consistent partly cracked XFEM element for cohesive crack growth." *International Journal for Numerical Methods in Engineering*, **72**:464–485, October 2007.

[Bab70] Ivo Babuška. "The finite element method for elliptic equations with discontinuous coefficients." *Computing*, **5**:207–213, 1970.

[Bab73] Ivo Babuška. "The finite element method with Lagrangian multipliers." *Numerische Mathematik*, **20**:179–192, 1973.

[BB99] T. Belytschko and T. Black. "Elastic crack growth in finite elements with minimal remeshing." *International Journal for Numerical Methods in Engineering*, **45**(5):601–620, June 1999.

[BBB07] Christopher Batty, Florence Bertails, and Robert Bridson. "A fast variational framework for accurate solid-fluid coupling." *ACM Trans. Graph.*, **26**(3), July 2007.

[BBE08] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. "PETSc Users Manual." Technical Report ANL-95/11 - Revision 3.0.0, Argonne National Laboratory, 2008.

[BBG09] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. "PETSc Web page.", 2009. http://www.mcs.anl.gov/petsc.

[BBH09] Roland Becker, Erik Burman, and Peter Hansbo. "A Nitsche extended finite element method for incompressible elasticity with discontinuous modulus of elasticity." *Computer Methods in Applied Mechanics and Engineering*, **198**(4144):3352 – 3360, 2009.

[BBZ10] Jacob Bedrossian, James H. von Brecht, Siwei Zhu, Eftychios Sifakis, and Joseph M. Teran. "A second order virtual node method for elliptic problems with interfaces and irregular domains." *Journal of Computational Physics*, **229**(18):6405 – 6426, 2010.

[BC04] Ted Belytschko and Hao Chen. "Singular enrichment finite element method for elastodynamic crack propagation." *International Journal of Computational Methods*, **1**(1):1–15, 2004.

[BCL08] J. Thomas Beale, Davod L. Chopp, Randall J. LeVeque, and Zhilin Li. "Correction to the article a comparison of the extended finite element method with the immersed interface method for elliptic equations with discontinuous coefficients and singular sources by Vaughan et al." *Comm. App. Math. and Comp. Sci.*, **3**(1):95–101, 2008.

[BD10] Philip T. Barton and Dimitris Drikakis. "An Eulerian method for multicomponent problems in non-linear elasticity with sliding interfaces." *J. Comput. Phys.*, **229**(15):5518–5540, August 2010.

[BDL07] Stephane Pierre Alain Bordas, M. Duflot, and P Le. "A simple error estimator for extended finite elements." *Communications in Numerical Methods in Engineering*, **24**(11):961–971, 2007.

[BDM06] I. Bijelonja, I. Demirdic, and S. Muzaferija. "A finite volume method for incompressible linear elasticity." *Computer Methods in Applied Mechanics and Engineering*, **195**(4447):6378 – 6390, 2006.

[Ber04] Petter Andreas Berthelsen. "A decomposed immersed interface method for variable coefficient elliptic equations with non-smooth and discontinuous solutions." *J. Comput. Phys.*, **197**:364–386, June 2004.

[BF91] Franco Brezzi and Michel Fortin. *Mixed and hybrid finite element methods.* Springer-Verlag New York, Inc., New York, NY, USA, 1991.

[BG09] Fedderik van der Bos and Volker Gravemeier. "Numerical simulation of premixed combustion using an enriched finite element method." *J. Comput. Phys.*, **228**:3605–3624, June 2009.

[BGL05] Michele Benzi, Gene H. Golub, and Jrg Liesen. "Numerical solution of saddle point problems." *ACTA NUMERICA*, **14**:1–137, 2005.

[BGM97] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries." In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pp. 163–202. Birkhäuser Press, 1997.

[BGV09] Ted Belytschko, Robert Gracie, and Giulio Ventura. "A review of extended/generalized finite element methods for material modeling." *Modelling and Simulation in Materials Science and Engineering*, **17**(4):043001, 2009.

[BGW04] René De Borst, Miguel A. Gutiérrez, Garth N. Wells, Joris J. C. Remmers, and Harm Askes. "Cohesive-zone models, higher-order continuum theories and reliability methods for computational failure analysis." *International Journal for Numerical Methods in Engineering*, **60**(1):289–315, 2004.

[BHT07] Zhaosheng Bao, Jeong-Mo Hong, Joseph Teran, and Ronald Fedkiw. "Fracturing rigid materials." *IEEE Transactions on Visualization and Computer Graphics*, **13**:370–378, March 2007.

[BK96] James Bramble and J. King. "A finite element method for interface problems in domains with smooth boundaries and interfaces." *Advances in Computational Mathematics*, **6**:109–138, 1996.

[BL06] J. Thomas Beale and Anita T. Layton. "On the accuracy of finite difference methods for elliptic problems with interfaces." *Commun. Appl. Math. Comput. Sci.*, **1**:91–119, 2006.

[BMM05] E. Béchet, H. Minnebo, N. Moës, and B. Burgardt. "Improved implementation and robustness study of the X-FEM for stress analysis around cracks." *International Journal for Numerical Methods in Engineering*, **64**(8):1033–1056, October 2005.

[BMU01] T. Belytschko, N. Moës, S. Usui, and C. Parimi. "Arbitrary discontinuities in finite elements." *Int. J. Numer. Meth. Engng*, **50**:993–1013, February 2001.

[BP88] James H. Bramble and Joseph E. Pasciak. "Corrigenda: "A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems"." *Math Comp*, **51**(181):387–388, 1988.

[Bra07] D. Braess. *Finite Elements: Theory, Fast Solvers, and Applications in Elasticity Theory*. Cambridge University Press, 2007.

[Bre93] Susanne C. Brenner. "A nonconforming mixed multigrid method for the pure displacement problem in planar linear elasticity." *SIAM J. Numer. Anal.*, **30**(1):116–135, February 1993.

[BTT97] F. Bertrand, P. A. Tanguy, and F. Thibault. "A three-dimensional fictitious domain method for incompressible fluid flow problems." *International Journal for Numerical Methods in Fluids*, **25**(6):719–736, 1997.

[BTV08] E. Benvenuti, A. Tralli, and Giulio Ventura. "A regularized XFEM model for the transition from continuous to discontinuous displacements." *International Journal for Numerical Methods in Engineering*, **74**(6):911–944, 2008.

[BYZ04] George Biros, Lexing Ying, and Denis Zorin. "A fast solver for the Stokes equations with distributed forces in complex geometries." *J. Comput. Phys.*, **193**(1):317–348, January 2004.

[BZM04] É. Budyn, G. Zi, N. Moës, and T. Belytschko. "A method for multiple crack growth in brittle materials without remeshing." *International Journal for Numerical Methods in Engineering*, **61**(10):1741–1770, November 2004.

[CB93] D. Chapelle and K.J. Bathe. "The inf-sup test." *Computers & Structures*, **47**(4-5):537 – 545, 1993.

[CB03] J. Chessa and T. Belytschko. "An Extended Finite Element Method for Two-Phase Fluids." *ASME J of Appl Mech*, **70**(1):10, 2003.

[CC05] A. H. Coppola-Own and R. Codina. "Improving Eulerian two-phase flow finite element approximation with discontinuous gradient pressure shape functions." *International Journal for Numerical Methods in Fluids*, **49**(12):1287–1304, December 2005.

[CCG10] R. K. Crockett, P. Colella, and D. T. Graves. "A Cartesian grid embedded boundary method for solving the Poisson and heat equations with discontinuous coefficients in three dimensions." June 2010. LBNL Paper LBNL-2929E.

[CGL09] Bernardo Cockburn, Jayadeep Gopalakrishnan, and Raytcho Lazarov. "Unified hybridization of discontinuous Galerkin, mixed, and conforming Galerkin methods for second order elliptic problems." *SIAM J. Numer. Anal.*, **47**:1319–1365, February 2009.

[CLR06] Elie Chahine, Patrick Laborde, and Yves Renard. "A quasi-optimal convergence result for fracture mechanics with XFEM." *Comptes Rendus Mathematique*, **342**(7):527 – 532, 2006.

[CLR08] E. Chahine, Patrick Laborde, and Yves Renard. "Crack tip enrichment in the XFEM using a cutoff function." *International Journal for Numerical Methods in Engineering*, **75**(6):629–646, 2008.

[CMM98] Zhiqiang Cai, Thomas A. Manteuffel, Stephen F. McCormick, and Seymour V. Parter. "First-Order System Least Squares (FOSLS) for Planar Linear Elasticity: Pure Traction Problem." *SIAM J. Numer. Anal.*, **35**(1):320–335, February 1998.

[Cow73] G. R. Cowper. "Gaussian quadrature formulas for triangles." *International Journal for Numerical Methods in Engineering*, **7**:405–408, 1973.

[CRW08] Vivien Challis, Anthony Roberts, and Andrew Wilkins. "Fracture resistance via topology optimization." *Structural and Multidisciplinary Optimization*, **36**:263–271, 2008. 10.1007/s00158-007-0160-0.

[CS07] I-Liang Chern and Yu-Chen Shu. "A coupling interface method for elliptic interface problems." *J. Comput. Phys.*, **225**:2138–2174, August 2007.

[CS08] Tianbing Chen and John Strain. "Piecewise-polynomial discretization and Krylov-accelerated multigrid for elliptic interface problems." *J. Comput. Phys.*, **227**:7503–7542, August 2008.

[CZ96] Zhiming Chen and Jun Zou. "Finite element methods and their convergence for elliptic and parabolic interface problems." *Numer. Math*, **79**:175–202, 1996.

[DD04] J. E. Dolbow and A. Devan. "Enrichment of enhanced assumed strain approximations for representing strong discontinuities: addressing volumetric incompressibility and the discontinuous patch test." *International Journal for Numerical Methods in Engineering*, **59**(1):47–67, 2004.

[DF08] J.E. Dolbow and L.P. Franca. "Residual-free bubbles for embedded Dirichlet problems." *Comput. Methods Appl. Mech. Engrg.*, **197**:3751–3759, August 2008.

[DH09] John Dolbow and Isaac Harari. "An efficient finite element method for embedded interface problems." *Int. J. Numer. Meth. Engng*, **78**:229–252, April 2009.

[DIL03] Shaozhong Deng, Kazufumi Ito, and Zhilin Li. "Three-dimensional elliptic solvers for interface problems and applications." *J. Comput. Phys.*, **184**:215–243, January 2003.

[DMD00] Christophe Daux, Nicolas Moës, John Dolbow, Natarajan Sukumar, and Ted Belytschko. "Arbitrary branched and intersecting cracks with the extended finite element method." *Int. J. Numer. Meth. Engng*, **48**:1741–1760, August 2000.

[DMJ06] Pierre Duysinx, Laurent Miegroet, Thibault Jacobs, and Claude Fleury. "Generalized Shape Optimization Using X-FEM and Level Set Methods." In Martin Philip Bendse, Niels Olhoff, Ole Sigmund, and G. M. L. Gladwell, editors, *IUTAM Symposium on Topological Design Optimization of Structures, Machines and Materials*, volume 137 of *Solid Mechanics and Its Applications*, pp. 23–32. Springer Netherlands, 2006.

[Doh03] Clark R. Dohrmann. "A Preconditioner for Substructuring Based on Constrained Energy Minimization." *SIAM J. Sci. Comput.*, **25**(1):246–258, January 2003.

[Dol99] John Dolbow. *An extended finite element method with discontinuous enrichment for applied mechanics*. PhD thesis, Northwestern University, December 1999.

[Dry05] Maksymilian Dryja. "A Neumann-Neumann algorithm for a mortar discretization of elliptic problems with discontinuous coefficients." *Numer. Math.*, **99**:645–656, February 2005.

[DSM09] Qinglin Duan, Jeong-Hoon Song, Thomas Menouillard, and Ted Belytschko. "Element-local level set method for three-dimensional dynamic crack growth." *International Journal for Numerical Methods in Engineering*, **80**(12):1520–1543, December 2009.

[Duf07] M. Duflot. "A study of the representation of cracks with level sets." *International Journal for Numerical Methods in Engineering*, **70**(11):1261–1302, June 2007.

[FB06] Thomas-Peter Fries and Ted Belytschko. "The intrinsic XFEM: a method for arbitrary discontinuities without additional unknowns." *Int. J. Numer. Meth. Engng*, **68**:1358–1385, December 2006.

[FLP00] Charbel Farhat, Michael Lesoinne, and Kendall Pierson. "A scalable dual-primal domain decomposition method." *Numerical Linear Algebra with Applications*, **7**(7-8):687–714, October-December 2000.

[GF05] Frédéric Gibou and Ronald Fedkiw. "A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem." *J. Comput. Phys.*, **202**:577–601, January 2005.

[GFC02] Frederic Gibou, Ronald P. Fedkiw, Li-Tien Cheng, and Myungjoo Kang. "A second-order-accurate symmetric discretization of the Poisson equation on irregular domains." *J. Comput. Phys.*, **176**:205–227, February 2002.

[GGL08] F. J. Gaspar, J. L. Gracia, F. J. Lisbona, and C. W. Oosterlee. "Distributive smoothers in multigrid for problems with dominating graddiv operators." *Numerical Linear Algebra with Applications*, **15**(8):661–683, October 2008.

[GLJ09] Grégory Guyomarc'h, Chang-Ock Lee, and Kiwan Jeon. "A discontinuous Galerkin method for elliptic interface problems with application to electroporation." *Commun. Numer. Meth. Engng*, **25**:991–1008, October 2009.

[GMB02] A. Gravouil, Nicolas Moës, and Ted Belytschko. "Non-planar 3D crack growth by the extended finite element and level sets-Part II: Level set update." *International Journal for Numerical Methods in Engineering*, **53**(11):2569–2586, 2002.

[GPH99] R. Glowinski, T.-W. Pan, T.I. Hesla, and D.D. Joseph. "A distributed Lagrange multiplier/fictitious domain method for particulate flows." *International Journal of Multiphase Flow*, **25**(5):755 – 794, 1999.

[GPH01] R. Glowinski, T. W. Pan, T. I. Helsa, D. D. Joseph, and J. Périaux. "A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flow." *J. Comput. Phys.*, **169**(2):363–426, May 2001.

[GPP94a] Roland Glowinski, Tsorng-Whay Pan, and Jacques Periaux. "A fictitious domain method for Dirichlet problem and applications." *Comput. Methods Appl. Mech. Engrg.*, **111**:283–303, 1994.

[GPP94b] Roland Glowinski, Tsorng-Whay Pan, and Jacques Periaux. "A fictitious domain method for external incompressible viscous flow modeled by Navier-Stokes equations." *Computer Methods in Applied Mechanics and Engineering*, **112**(14):133 – 148, 1994.

[GR07] Sven Groí and Arnold Reusken. "An extended pressure finite element space for two-phase incompressible flows with surface tension." *J. Comput. Phys.*, **224**:40–58, May 2007.

[GW08] Axel Gerstenberger and Wolfgang A. Wall. "An eXtended Finite Element Method/Lagrange multiplier based approach for fluidstructure interaction." *Computer Methods in Applied Mechanics and Engineering*, **197**(1920):1699 – 1714, 2008. Computational Methods in FluidStructure Interaction.

[Hau90] Z. Haung. "A multi-grid algorithm for mixed problems with penalty." *Numer. Math.*, **57**(3):227–247, May 1990.

[HB09] D. B. P. Huynh and T. Belytschko. "The extended finite element method for fracture in composite materials." *International Journal for Numerical Methods in Engineering*, **77**(2):214–239, January 2009.

[HD10] Isaac Harari and John Dolbow. "Analysis of an efficient finite element method for embedded interface problems." *Computational Mechanics*, **46**:205–211, 2010.

[HH99] Ralf Hiptmair and Ronald H.W. Hoppe. "Multilevel methods for mixed finite elements in three dimensions." *Numerische Mathematik*, **82**:253–279, 1999.

[HH02] Anita Hansbo and Peter Hansbo. "An unfitted finite element method, based on Nitsche's method, for elliptic interface problems." *Comput. Methods Appl. Mech. Engrg.*, **191**:5537–5552, November 2002.

[HH04] Anita Hansbo and Peter Hansbo. "A finite element method for the simulation of strong and weak discontinuities in solid mechanics." *Comput. Methods Appl. Mech. Engrg.*, **193**:3523–3540, August 2004.

[HL05] Songming Hou and Xu-Dong Liu. "A numerical method for solving variable coefficient elliptic equation with interfaces." *J. Comput. Phys.*, **202**:411–445, January 2005.

[HMM04] J. J. Heys, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge. "First-order system least squares (FOSLS) for coupled fluid-elastic problems." *J. Comput. Phys.*, **195**(2):560–575, April 2004.

[HNS08] David J. Holdych, David R. Noble, and Robert B. Secor. "Quadrature rules for triangular and tetrahedral elements with generalized functions." *International Journal for Numerical Methods in Engineering*, **73**(9):13101327, 2008.

[HPO10] D. J. Hill, D. Pullin, M. Ortiz, and D. Meiron. "An Eulerian hybrid WENO centered-difference solver for elastic-plastic solids." *J. Comput. Phys.*, **229**(24):9053–9072, December 2010.

[HSS09] Jeffrey Hellrung, Andrew Selle, Arthur Shek, Eftychios Sifakis, and Joseph Teran. "Geometric fracture modeling in Bolt." In *SIGGRAPH 2009: Talks*, SIGGRAPH '09, pp. 7:1–7:1, New York, NY, USA, 2009. ACM.

[HW65] F. H. Harlow and J. E. Welch. "Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface." *Physics of Fluids*, **8**(12):2182–2189, 1965.

[HW98] Houde Han and Xiaonan Wu. "A New Mixed Finite Element Formulation and the MAC Method for the Stokes Equations." *SIAM J. Numer. Anal.*, **35**(2):560–571, April 1998.

[HWS12] Jeffrey Lee Hellrung, Jr., Luming Wang, Eftychios Sifakis, and Joseph M. Teran. "A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions." *J. Comput. Phys.*, **231**(4):2015–2048, February 2012.

[HWW10] Songming Hou, Wei Wang, and Liqun Wang. "Numerical method for solving matrix coefficient elliptic equation with sharp-edged interfaces." *J. Comput. Phys.*, **229**:7162–7179, September 2010.

[Hym52] Morton Hyman. "Non-iterative numerical solution of boundary-value problems." *Applied Scientific Research, Section B*, **2**:325–351, 1952. 10.1007/BF02919780.

[HZ01] Jianguo Huang and Jun Zou. "A mortar element method for elliptic problems with discontinuous coefficients." *IMA J Numer Anal*, **22**:549–576, July 2001.

[ITF04] G. Irving, J. Teran, and R. Fedkiw. "Invertible finite elements for robust simulation of large deformation." In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pp. 131–140, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[JC98] Hans Johansen and Phillip Colella. "A Cartesian grid embedded boundary method for Poisson's equation on irregular domains." *J. Comput. Phys.*, **147**:60–85, November 1998.

[JCD02] H. Ji, D. Chopp, and J. E. Dolbow. "A hybrid extended finite element/level set method for modeling phase transformations." *International Journal for Numerical Methods in Engineering*, **54**(8):1209–1233, July 2002.

[JD04] H. Ji and J. E. Dolbow. "On strategies for enforcing interfacial constraints and evaluating jump conditions with extended finite element method." *Int. J. Numer. Meth. Engng*, **61**:2508–2535, December 2004.

[JM05] Z. Jomaa and C. Macaskill. "The embedded finite difference method for the Poisson equation in a domain with an irregular boundary and Dirichlet boundary conditions." *J. Comput. Phys.*, **202**:488–506, January 2005.

[JM10] Z. Jomaa and C. Macaskill. "The Shortley-Weller embedded finite-difference method for the 3D Poisson equation with mixed boundary conditions." *J. Comput. Phys.*, **229**:3675–3690, May 2010.

[JSC06] B. L. Vaughn Jr., B. G. Smith, and D. L. Chopp. "A comparison of the extended finite element method with the immersed interface method for elliptic equations with discontinuous coefficients and singular sources." *Comm. App. Math. and Comp. Sci.*, **1**(1):207–228, 2006.

[Kla95] Axel Klawonn. "An Optimal Preconditioner for a Class of Saddle Point Problems with a Penalty Term." *SIAM J. Sci. Comput*, **19**:540–552, 1995.

[Kla98] Axel Klawonn. "Block-Triangular Preconditioners for Saddle Point Problems with a Penalty Term." *SIAM J. Sci. Comput.*, **19**(1):172–184, January 1998.

[KM87] Michal Kočvara and Jan Mandel. "A multigrid method for three-dimensional elasticity and algebraic convergence estimates." *Appl. Math. Comput.*, **23**(2):121–135, August 1987.

[KP98] Axel Klawonn and Luca F. Pavarino. "Overlapping Schwarz methods for mixed linear elasticity and Stokes problems." *Computer Methods in Applied Mechanics and Engineering*, **165**(14):233 – 245, 1998.

[KPB08] Ashok V. Kumar, Sanjeev Padmanabhan, and Ravi Burla. "Implicit boundary method for finite element analysis using non-conforming mesh or grid." *Int. J. Numer. Meth. Engng*, **74**:1421–1447, May 2008.

[KWC10] Do Y. Kwak, Kye T. Wee, and Kwang S. Chang. "An Analysis of a Broken P_1-Nonconforming Finite Element Method for Interface Problems." *SIAM Journal on Numerical Analysis*, **48**(6):2117–2134, 2010.

[KX03] B.L. Karihaloo and Q.Z. Xiao. "Modelling of stationary and growing cracks in FE framework without remeshing: a state-of-the-art review." *Computers & Structures*, **81**(3):119 – 129, 2003.

[LB08] Adrián J. Lew and Gustavo C. Buscaglia. "A discontinuous-Galerkin-based immersed boundary method." *Int. J. Numer. Meth. Engng*, **76**:427–454, October 2008.

[LFK00] Xu-Dong Liu, Ronald P. Fedkiw, and Myungjoo Kang. "A boundary condition capturing method for Poisson's equation on irregular domains." *J. Comput. Phys.*, **160**:151–178, May 2000.

[Li98a] Zhilin Li. "A fast iterative algorithm for elliptic interface problems." *SIAM J. Numer. Anal.*, **35**:230–254, February 1998.

[Li98b] Zhilin Li. "The immersed interface method using a finite element formulation." *Appl. Numer. Math.*, **27**:253–267, July 1998.

[LI01] Zhilin Li and Kazufumi Ito. "Maximum principle preserving schemes for interface problems with discontinuous coefficients." *SIAM J. Sci. Comput.*, **23**:339–361, January 2001.

[LI06] Zhilin Li and Kazufumi Ito. *The immersed interface method: numerical solutions of PDEs involving interfaces and irregular domains*, volume 33 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, July 2006.

[LKP06] D. V. Le, B. C. Khoo, and J. Peraire. "An immersed interface method for viscous incompressible flows involving rigid and flexible boundaries." *J. Comput. Phys.*, **220**:109–138, December 2006.

[LL94] Randall J. Leveque and Zhilin Li. "The immersed interface method for elliptic equations with discontinuous coefficients and singular sources." *SIAM J. Numer. Anal.*, **31**:1019–1044, August 1994.

[LL97] Randall J. LeVeque and Zhilin Li. "Immersed interface methods for stokes flow with elastic boundaries or surface tension." *SIAM J. Sci. Comput.*, **18**:709–735, May 1997.

[LL01] Zhilin Li and Ming-Chih Lai. "The immersed interface method for the Navier-Stokes equations with singular forces." *J. Comput. Phys.*, **171**:822–842, August 2001.

[LL03] Long Lee and Randall J. LeVeque. "An immersed interface method for incompressible Navier-Stokes equations." *SIAM J. Sci. Comput.*, **25**:832–856, March 2003.

[LLW03] Zhilin Li, Tao Lin, and Xiaohui Wu. "New Cartesian grid methods for interface problems using the finite element formulation." *NUMERISCHE MATHEMATIK*, **96**(1):61–98, 2003.

[LPR05] Patrick Laborde, Julien Pommier, Yves Renard, and Michel Salaün. "High-order extended finite element method for cracked domains." *International Journal for Numerical Methods in Engineering*, **64**(3):354–381, 2005.

[LS03] Xu-Dong Liu and Thomas C. Sideris. "Convergence of the ghost fluid method for elliptic equations with interfaces." *Math. Comput.*, **72**:1731–1746, October 2003.

[LW04] B. P. Lamichhane and B. I. Wohlmuth. "Mortar finite elements for interface problems." *Computing*, **72**:333–348, May 2004.

[LWC09] Young-Ju Lee, Jinbiao Wu, and Jinru Chen. "Robust multigrid method for the planar linear elasticity problems." *Numer. Math.*, **113**(3):473–496, August 2009.

[LWI06] Zhilin Li, Xiaohai Wan, Kazufumi Ito, and Sharon R. Lubkin. "An augmented approach for the pressure boundary condition in a Stokes flow." *Commun. Comput. Phys.*, **1**:874–885, 2006.

[MB96] J.M. Melenk and I. Babuka. "The partition of unity finite element method: Basic theory and applications." *Computer Methods in Applied Mechanics and Engineering*, **139**(14):289 – 314, 1996.

[MBF05] Neil Molino, Zhaosheng Bao, and Ron Fedkiw. "A virtual node algorithm for changing mesh topology during simulation." In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[MBT03] Neil Molino, Robert Bridson, Joseph Teran, and Ronald Fedkiw. "A Crystalline, Red Green Strategy for Meshing Highly Deformable Objects with Tetrahedra." In *IMR*, pp. 103–114, 2003.

[MBT06] Nicolas Moës, Eric Béchet, and Matthieu Tourbier. "Imposing Dirichlet boundary conditions in the extended finite element method." *Int. J. Numer. Meth. Engng*, **67**:1641–1669, September 2006.

[MCC03] N. Moës, M. Cloirec, P. Cartraud, and J.F. Remacle. "A computational approach to handle complex microstructure geometries." *Comput. Methods Appl. Mech. Engrg.*, **192**:3163–3177, July 2003.

[MDB99] Nicolas Moës, John Dolbow, and Ted Belytschko. "A finite element method for crack growth without remeshing." *Int. J. Numer. Meth. Engng*, **46**:131–150, September 1999.

[MDH07] Hashem M. Mourad, John Dolbow, and Isaac Harari. "A bubble-stabilized finite element method for Dirichlet constraints on embedded interfaces." *Int. J. Numer. Meth. Engng*, **69**:772–793, January 2007.

[MG07] Chohong Min and Frédéric Gibou. "Geometric integration over irregular domains with application to level-set methods." *J. Comput. Phys.*, **226**:1432–1443, October 2007.

[MGB02] Nicolas Moës, A. Gravouil, and Ted Belytschko. "Non-planar 3D crack growth by the extended finite element and level sets-Part I: Mechanical model." *International Journal for Numerical Methods in Engineering*, **53**(11):2549–2568, April 2002.

[MGS11] S. E. Mousavi, E. Grinspun, and N. Sukumar. "Harmonic enrichment functions: A unified treatment of multiple, intersecting and branched cracks in the extended finite element method." *International Journal for Numerical Methods in Engineering*, **85**(10):1306–1322, March 2011.

[MKL05] S. Marella, S. Krishnan, H. Liu, and H. S. Udaykumar. "Sharp interface Cartesian grid method I: An easily implemented technique for 3D moving boundary computations." *J. Comput. Phys.*, **210**(1):1–31, November 2005.

[Mo02] Nicolas Mo˙ "Extended finite element method for cohesive crack growth." *Engineering Fracture Mechanics*, **69**(7):813 – 833, May 2002.

[MP03] Stefano Mariani and Umberto Perego. "Extended finite element method for quasi-brittle fracture." *International Journal for Numerical Methods in Engineering*, **58**(1):103–126, September 2003.

[MS10] S.E. Mousavi and N. Sukumar. "Generalized Gaussian quadrature rules for discontinuities and crack singularities in the extended finite element method." *Computer Methods in Applied Mechanics and Engineering*, **199**(4952):3237 – 3249, 2010.

[MST10] A. McAdams, E. Sifakis, and J. Teran. "A parallel multigrid Poisson solver for fluids simulation on large grids." In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pp. 65–74, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[NMG09a] Yen Ting Ng, Chohong Min, and Frédéric Gibou. "An efficient fluid-solid coupling algorithm for single-phase flows." *J. Comput. Phys.*, **228**:8807–8829, December 2009.

[NMG09b] Yen Ting Ng, Chohong Min, and Frédéric Gibou. "An efficient fluid-solid coupling algorithm for single-phase flows." *J. Comput. Phys.*, **228**(23):8807–8829, December 2009.

[OF04] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*, volume 57. Springer, 2004.

[OK06] M. Oevermann and R. Klein. "A Cartesian grid finite volume method for elliptic equations with variable coefficients and embedded interfaces." *J. Comput. Phys.*, **219**:749–769, December 2006.

[OS88] Stanley Osher and James A. Sethian. "Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations." *J. Comput. Phys.*, **79**(1):12–49, November 1988.

[OS01] Stanley J. Osher and Fadil Santosa. "Level Set Methods for Optimization Problems Involving Geometry and Constraints: I. Frequencies of a Two-Density Inhomogeneous Drum." *Journal of Computational Physics*, **171**(1):272 – 288, 2001.

[OSK09] M. Oevermann, C. Scharfenberg, and R. Klein. "A sharp interface finite volume method for elliptic equations on Cartesian grids." *J. Comput. Phys.*, **228**:5184–5206, August 2009.

[Par08] Lucia Parussini. "Fictitious Domain Approach Via Lagrange Multipliers with Least Squares Spectral Element Method." *J. Sci. Comput.*, **37**(3):316–335, December 2008.

[PCG07] B. Prabel, A. Combescure, A. Gravouil, and S. Marie. "Level set X-FEM non-matching meshes: application to dynamic crack propagation in elasticplastic media." *International Journal for Numerical Methods in Engineering*, **69**(8):1553–1569, February 2007.

[Pes72] Charles S. Peskin. "Flow patterns around heart valves: A numerical method." *Journal of Computational Physics*, **10**(2):252 – 271, 1972.

[PGR10] Joseph Papac, Frédéric Gibou, and Christian Ratsch. "Efficient symmetric discretization for the Poisson, heat and Stefan-type problems with Robin boundary conditions." *J. Comput. Phys.*, **229**:875–889, February 2010.

[Pit79] Juhani Pitkäranta. "Boundary subspaces for the finite element method with Lagrange multipliers." *Numerische Mathematik*, **33**:273–289, 1979.

[PLR95] S.D. Pieper, D.R. Jr. Laub, and J.M. Rosen. "A finite-element facial model for simulating plastic surgery." *Plastic and Reconstructive Surgery*, **96**(5):1100–1105, October 1995.

[PP09] Lucia Parussini and Valentino Pediroda. "Fictitious Domain approach with hp-finite element approximation for incompressible fluid flow." *J. Comput. Phys.*, **228**:3891–3910, June 2009.

[PPD09] Kyoungsoo Park, Jeronymo P. Pereira, C. Armando Duarte, and Glaucio H. Paulino. "Integration of singular enrichment functions in the generalized/extended finite element method for three-dimensional problems." *International Journal for Numerical Methods in Engineering*, **78**(10):1220–1257, June 2009.

[PTH10] Kejia Pan, Yongji Tan, and Hongling Hu. "An interpolation matched interface and boundary method for elliptic interface problems." *J. Comput. Appl. Math.*, **234**:73–94, May 2010.

[PWZ10] Luca F. Pavarino, Olof B. Widlund, and Stefano Zampini. "BDDC Preconditioners for Spectral Element Discretizations of Almost Incompressible Elasticity in Three Dimensions." *SIAM J. Sci. Comput.*, **32**(6):3604–3626, December 2010.

[RHS11] Casey L. Richardson, Jan Hegemann, Eftychios Sifakis, Jeffrey Hellrung, and Joseph M. Teran. "An XFEM method for modeling geometrically elaborate crack propagation in brittle materials." *Int. J. Numer. Meth. Engng*, **88**:1042–1065, December 2011.

[Rut08] V. Rutka. "A staggered grid-based explicit jump immersed interface method for two-dimensional Stokes flows." *International Journal for Numerical Methods in Fluids*, **57**(10):1527–1543, 2008.

[RW04] V. Rutka and A. Wiegmann. "A fast finite difference method for elliptic PDEs in domains with non-grid aligned boundaries with application to 3D linear elasticity." *MATHEMATICS IN INDUSTRY*, **5**:363–366, 2004.

[SAB06] Jeong-Hoon Song, Pedro M. A. Areias, and Ted Belytschko. "A method for dynamic crack and shear band propagation with phantom nodes." *Int. J. Numer. Meth. Engng*, **67**:868–893, August 2006.

[Sau63] V. K. Saul'ev. "On solution of some boundary value problems on high performance computers by fictitious domain method." *Dokl Akad Nauk SSSR 144 1962 497500 in Russian English translation in Soviet Math Dokl*, **4**:912–925, 1963.

[SB09a] Jeong-Hoon Song and Ted Belytschko. "Cracking node method for dynamic fracture with finite elements." *International Journal for Numerical Methods in Engineering*, **77**:360–385, 2009.

[SB09b] Jeong-Hoon Song and Ted Belytschko. "Dynamic Fracture of Shells Subjected to Impulsive Loads." *Journal of Applied Mechanics*, **76**(5):051301, September 2009.

[SBC00] T. Strouboulis, I. Babuka, and K. Copps. "The design and analysis of the Generalized Finite Element Method." *Computer Methods in Applied Mechanics and Engineering*, **181**(13):43 – 69, 2000.

[SBC03] F. L. Stazi, E. Budyn, J. Chessa, and T. Belytschko. "An extended finite element method with higher-order elements for curved cracks." *Computational Mechanics*, **31**:38–48, 2003.

[SBC06] Peter Schwartz, Michael Barad, Phillip Colella, and Terry Ligocki. "A Cartesian grid embedded boundary method for the heat equation and Poisson's equation in three dimensions." *J. Comput. Phys.*, **211**:531–550, January 2006.

[SBM08] N. Sukumar, E. Béchet, and N. Moës. "Three-Dimensional Non-Planar Crack Growth by a Coupled Extended Finite Element and Fast Marching Method." *International Journal for Numerical Methods in Engineering*, **76**(5):727–748, 2008.

[Sch99] Joachim Schöberl. "Multigrid methods for a parameter dependent problem in primal variables." *Numerische Mathematik*, **84**:97–119, 1999. 10.1007/s002110050465.

[SCM01] N. Sukumar, D.L. Chopp, N. Mos, and T. Belytschko. "Modeling holes and inclusions by level sets in the extended finite-element method." *Computer Methods in Applied Mechanics and Engineering*, **190**(4647):6183 – 6200, 2001.

[SDF07] Eftychios Sifakis, Kevin G. Der, and Ronald Fedkiw. "Arbitrary cutting of deformable tetrahedralized objects." In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pp. 73–80, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[SHT09] E. Sifakis, J. Hellrung, J. Teran, A. Oliker, and C. Cutting. "Local flaps: a real-time finite element based solution to the plastic surgery defect puzzle." *Studies in Health Technology and Informatics*, **142**:313–318, 2009.

[SL09] Yongxing Shen and Adrian Lew. "An optimally convergent discontinuous Galerkin-based extended finite element method for fracture mechanics." *International Journal for Numerical Methods in Engineering*, **82**(6):716–755, May 2009.

[SMM00] N. Sukumar, N. Moës, B. Moran, and T. Belytschko. "Extended finite element method for three-dimensional crack modelling." *International Journal for Numerical Methods in Engineering*, **48**(11):1549–1570, August 2000.

[SSI07] Eftychios Sifakis, Tamar Shinar, Geoffrey Irving, and Ronald Fedkiw. "Hybrid simulation of deformable solids." In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pp. 81–90, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[SW00] J.A. Sethian and Andreas Wiegmann. "Structural Boundary Design via Level Set and Immersed Interface Methods." *Journal of Computational Physics*, **163**(2):489 – 528, 2000.

[TLL08] Zhijun Tan, D. V. Le, Zhilin Li, K. M. Lim, and B. C. Khoo. "An immersed interface method for solving incompressible viscous flows with piecewise constant viscosity across a moving elastic membrane." *J. Comput. Phys.*, **227**:9955–9983, December 2008.

[TP09] J.M. Teran and C.S. Peskin. "Tether force constraints in Stokes flow by the immersed boundary method on a periodic domain." *SIAM Journal on Scientific Computing*, **31**(5):3404–3416, 2009.

[TSI05] Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. "Robust quasistatic finite elements and flesh simulation." In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, pp. 181–190, New York, NY, USA, 2005. ACM.

[Ven06] Giulio Ventura. "On the elimination of quadrature subcells for discontinuous functions in the eXtended Finite-Element Method." *International Journal for Numerical Methods in Engineering*, **66**(5):761–795, 2006.

[Ver84] R. Verfürth. "A Multilevel Algorithm for Mixed Problems." *SIAM J. on Numerical Analysis*, **21**(2):264–271, April 1984.

[VGB09] Giulio Ventura, Robert Gracie, and Ted Belytschko. "Fast integration and weight function blending in the extended finite element method." *International Journal for Numerical Methods in Engineering*, **77**(1):1–29, January 2009.

[VGL09] L. Beirão da Veiga, V. Gyrya, K. Lipnikov, and G. Manzini. "Mimetic finite difference method for the Stokes problem on polygonal meshes." *J. Comput. Phys.*, **228**(19):7215–7232, October 2009.

[WB00] Andreas Wiegmann and Kenneth P. Bube. "The explicit-jump immersed interface method: finite difference methods for pdes with piecewise smooth solutions." *SIAM J. Numer. Anal.*, **37**:827–862, February 2000.

[Wie00] Christian Wieners. "Robust multigrid methods for nearly incompressible elasticity." *Computing*, **64**(4):289–306, July 2000.

[WK99] Barbara I. Wohlmuth and Rolf H. Krause. "Multigrid Methods Based On The Unconstrained Product Space Arising From Mortar Finite Element Discretizations." *SIAM J. NUMER. ANAL*, **39**:2001, 1999.

[WL04] Justin W. L. Wan and Xu-Dong Liu. "A Boundary Condition–Capturing Multigrid Approach to Irregular Boundary Problems." *SIAM J. Sci. Comput.*, **25**:1982–2003, June 2004.

[WML01] G.J. Wagner, N. Moës, W. K. Liu, and T. Belytschko. "The extended finite element method for rigid particles in Stokes flow." *International Journal for Numerical Methods in Engineering*, **51**(3):293–313, May 2001.

[WW08] Peng Wei and Michael Yu Wang. "A structural optimization method with XFEM and level set." *Structural Optimization*, (February):1–16, 2008.

[WX10] Haijun Wu and Yuanming Xiao. "An unfitted *hp*-interface penalty finite element method for elliptic interface problems." 2010.

[XW06] Sheng Xu and Z. Jane Wang. "An immersed interface method for simulating the interaction of a fluid with moving boundaries." *J. Comput. Phys.*, **216**:454–493, August 2006.

[XW08] Sheng Xu and Z. Jane Wang. "A 3D immersed interface method for fluid-solid interaction." *Computer Methods in Applied Mechanics and Engineering*, **197**(25-28):2068 – 2086, 2008. Immersed Boundary Method and Its Extensions.

[YMB90] David P. Young, Robin G. Melvin, Michael B. Bieterman, Forrester T. Johnson, Satish S. Samant, and John E. Bussoletti. "A locally refined rectangular grid finite element method: application to computational fluid dynamics and computational physics." *J. Comput. Phys.*, **92**:1–66, December 1990.

[YW07] Sining Yu and G. W. Wei. "Three-dimensional matched interface and boundary (MIB) method for treating geometric singularities." *J. Comput. Phys.*, **227**:602–632, November 2007.

[ZB03] Goangseup Zi and Ted Belytschko. "New crack-tip elements for XFEM and applications to cohesive cracks." *International Journal for Numerical Methods in Engineering*, **57**(15):2221–2240, August 2003.

[ZSB04] G. Zi, J. H. Song, É. Budyn, S. H. Lee, and T. Belytschko. "A method for growing multiple cracks without remeshing and its application to fatigue crack growth." *Modelling and Simulation in Materials Science and Engineering*, **12**(1):901–915, 2004.

[ZST10] Yongning Zhu, Eftychios Sifakis, Joseph Teran, and Achi Brandt. "An efficient multigrid method for the simulation of high-resolution elastic solids." *ACM Trans. Graph.*, **29**(2):16:1–16:18, April 2010.

[ZW06] Y. C. Zhou and G. W. Wei. "On the fictitious-domain and interpolation formulations of the matched interface and boundary (MIB) method." *J. Comput. Phys.*, **219**:228–246, November 2006.

[ZW09] Shan Zhao and G. W. Wei. "Matched interface and boundary (MIB) for the implementation of boundary conditions in high-order central finite differences." *Int. J. Numer. Meth. Engng*, **77**:1690–1730, March 2009.

[ZZF06] Y. C. Zhou, Shan Zhao, Michael Feig, and G. W. Wei. "High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources." *J. Comput. Phys.*, **213**:1–30, March 2006.