University of California
Los Angeles

Multigrid methods for solids simulation

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

by

Yongning Zhu

2010

The dissertation of Yongning Zhu is approved.

Achi Brandt

Stanley Osher

Demetri Terzopoulos

Joseph M. Teran, Committee Chair

University of California, Los Angeles

2010

For my family, who offered me unconditional love and support throughout the course of this thesis

Table of Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgments

Foremost, I would like to express my sincere gratitude to my advisors Achi Brandt and Joseph Teran. Professor Achi Brandt taught me a lot about mathematics methodology, especially multigrid methods. His research attitude has affected me greatly. Professor Joseph Teran has continuously supported me. I owe him many thanks for his dedication to my research. His talented ideas have inspired me in many difficult situations. I would like to especially thank Dr. Eftychios Sifakis for his most insightful instruction and for sharing his experience without reservation.

I am grateful to Professor Andrea Bertozzi for admitting me to UCLA, which offered me the great opportunity to work and study with excellent researchers. I am also grateful to Professor Stanley Osher, Professor Demetri Terzopoulos, Professor Christian Anderson and Professor Luminita Vese for their wonderful classes and insightful instructions. I would like to particularly thank Dr. Andrew Selly and Dr. Rasmus Tamstorf from Walt Disney Animation who game me a special opportunity to learn about animation production and software development in the animation industry. I would like to thank Maggie Albert and Martha Contreras for their assistance during my PhD program. I am also indebted to many of my colleagues for supporting me. Jefferey Hellrung has helped me a lot with using Boost. Alejandro Cantarero has shared many insightful thoughts with me. Numerous other collaborators offered critical support at the most difficult times in this process.

I would also like to express my sincere gratitude to my friends in the UCLA Mathematics Departments, including Yifei Lou, Yan Wang, Mi Youn Jung, Yanghong Huang, Bin Dong, Xiaoqun Zhang, Alex Chen, Ming Yan, Jinjun Xu, Yu Mao, Rongjie Lai and many other friends who have helped and supported me.

Chapter 3 to 6 is a version of Zhu, Y., Sifakis, E., Teran, J., and Brandt, A. (2010). An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Transactions on Graphics*, 29(2):118. I am grateful to the Intel Microprocessor Research Lab and, in paricular, to Pradeep Dubey and Victor Lee for their invaluable assistance in carrying out the parallel experiments in this work, as well as their substantial material and scientific support. Special thanks to Andrew Selle for his help with our audio-visual materials and his extensive constructive feedback. I and my collaborators Eftichios Sifakis, Joseph Teran were supported in part by DOE 09-LR-04-116741-BERA, NSF DMS- 0652427, NSF CCF-0830554, ONR N000140310071. I was also supported by an Intel Larrabee Research Grant.

Lastly, I offer my gratitude and best wishes to many others who supported me in all aspects during the completion of my PhD program.

Yongning Zhu

October 28, 2010

Vita

1981	Born, Beijing, P. R. China.
2003	B.S (Mathematics), Peking University.
2005	M.S (Computer Science), University of British Columbia.

Publications

Y. Zhu, E. Sifakis, J. Teran and A. Brandt. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Transactions on Graphics*, 28:4, 2009.

Y.Zhu, Y. Wang, J. Hellrung, E. Sifakis and J. Teran. A second order virtual node algorithm for nearly incompressible linear elasticity in irregular domains. In preparation.

A. McAdams, Y. Zhu, A. Selle, R. Tamstorf, J. Teran and E. Sifakis. Interactive volumetric skin simulation for articulated characters. In preparation.

Abstract of the Dissertation

Multigrid methods for solids simulation

by

Yongning Zhu

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 2010

Professor Joseph M. Teran, Chair

The simulation of deformable solids is a traditional topic in mechanics. Recently, deformable solids simulation has been widely used in computer animation, game development and virtual surgery to generate realistic deformations. Numerous methods have been developed to accelerate the simulation of realistic materials with complicated geometries, dynamic effects, as well as under user interface control and interaction with rigid bodies and fluids. Recently, there has been an increasing interest in the application of multigrid methods to practical problems, targeting interactive simulation with very high resolutions and with the help of parallel computer implementations.

This thesis presents a multigrid framework for the simulation of high resolution elastic deformable models. The framework incorporates several state-of-the-art techniques from multigrid theory, while adapting them to the specific requirements of graphics, animation and engineering applications, such as the ability to handle elaborate geometry and complex boundary conditions. The efficiency of the solver is practically independent of material parameters, even for near-incompressible materials. The solver also supports the simulation of co-rotational linear elasticity and dynamic systems. With optimization, the al-

gorithm achieves simulation rates as high as 6 frames per second on an 8-core SMP for test models with 256K vertices, and 1.6 frames per second on a 16-core SMP for a 2M vertex object.

To achieve higher-order accuracy solutions, we also present a cut cell method for enforcing Dirichlet and Neumann boundary conditions with nearly incompressible linear elastic materials in irregular domains. Virtual nodes on cut uniform grid cells are used to provide geometric flexibility in the domain boundary shape without sacrificing accuracy. We use a mixed formulation utilizing a MAC-type staggered grid with piecewise bilinear displacements centered at cell faces and piecewise constant pressures at cell centers. These discretization choices provide the necessary stability in the incompressible limit and the necessary accuracy in cut cells. Numerical experiments suggest second order accuracy in L^∞. A geometric multigrid method is developed for solving the discrete equations for displacements and pressures that achieves nearly optimal convergence rates independent of grid resolution.

Finally, we propose a soft constraint model for controllable deformable solids simulation and collision problems, and develop an efficient multigrid solver for the constraint system.

CHAPTER 1

Introduction

The simulation of solids is an important problem and tool in computer animation, physics, mechanical engineering and biomedical engineering. Deformable solids simulation has been widely used to automate the generation of realistic visual effects related to deformable solids, analyze mechanical systems, and facilitate researchers in imitating and investigating biomedical systems. With increasingly hardware performance, problems with higher resolutions are becoming increasingly important. There has always been a need for simulation at interactive rates with higher accuracy, which typically leads to a large numerical system where the efficiency of traditional numerical methods is largely inadequate. Finding an efficient numerical solver for high resolution solids simulation is an interesting and challenging problem.

The theme of this thesis is efficient numerical methods for the simulation of elastic solids. I present an efficient multigrid framework that supports the dynamic simulation of very high resolution deformable solids with a wide range of materials, and utilize it in several applications.

1.1 Solids simulation

A fundamental task of mechanical engineering is to understand the mechanical properties of different materials and to design and control mechanical systems.

Within computational mathematics, numerical simulation offers an important tool for predicting the performance of mechanical systems without real world experiments, which may be expensive, dangerous or even impractical. Solids simulation approximates the behavior of mechanical systems with computational models, and thus can be used to compare existing models with experimental results in order to develop more realistic models. Furthermore, solids simulation helps researchers to foresee problems in control and design. The more accurate the simulation, the better and more reliable are the conclusions that can be offered to engineers.

In particular, the simulation of elastic materials is interesting for its broad applicability in architecture, manufacture, geophysics, material science and biomedical engineering. In particular, elasticity simulation has recently been used to generate realistic visual effects in computer animation, for motion pictures and interactive games

1.1.1 Solids simulation in computer animation

Physics-based simulation of the governing partial differential equations (PDE) of nonrigid solids has been particularly useful in computer animation due to its reliable imitation of real world materials Terzopoulos et al. (1987); Terzopoulos and Fleischer (1988a); Müller et al. (2002). Traditional animation requires highly experienced artists to manually rig the deformation and interpolate keyframe deformations to generate the animation. The more keyframes that are inserted, the more accurately the deformation is controlled. However, the naturalness of the animation depends critically on the expertise of the artist. A lot of hands-on work is required and it is very difficult to repeat the process once a minor modification is made to the model. Although procedural deformation techniques

are continuously being developed to target the automation of these tasks, it still requires a lot of artistic skills and training to achieve natural dynamics.

Simulating virtual character flesh and tissues as an elastic material by solving the governing equations of physics automatically generates deformations that are consistent with the behavior of real world materials. Volumetric deformations such as bulging and stretching are naturally resolved with a minimum amount of manual effort. Also, when a very high resolution can be afforded, much more interesting and compelling secondary visual effects can be generated with fine-scale detail and complexity. The quest for visual realism has spawned an ever growing interest in simulation techniques capable of accommodating larger and more detailed models and generating visually realistic results.

Beyond simulation that exactly follows the physics rule of elastic materials, generating physically plausible deformations under artists' control is an equally important problem in computer animation as discussed in Barzel et al. (1996). In certain cases, special effects with physically unrealistic movements are required, such as squeeze, squash and special deformations designed by artists. Moreover, artists need to preview the deformation in order to better control the final results, hence accelerating the production of the animation. This requires a fast simulation method capable of synthesizing very high resolution, high quality deformations, as well as the interactive simulation of reasonably detailed systems with reliable and repeatable results.

Other than volumetric objects, the elastic simulation of lower dimensional geometries, including curves and surfaces, is also very useful in animation, such as cloth simulation Bridson et al. (2003); Choi and Ko (2005a); Goldenthal et al. (2007) and hair simulation Selle et al. (2008). Techniques have been developed to simulate facial expressions Terzopoulos and Waters (1990); Sumner and Popović

(2004); Sifakis et al. (2005). Solids simulation techniques have also been at the heart of new techniques in image processing Terzopoulos and McInerney (1996); Andersson et al. (2001), geoscience and material science.

1.1.2 Biomedical simulation and virtual surgery

With the development of increasingly practical and reliable solids simulation techniques in computer animation, the simulation of complicated realistic materials such as biomedical systems becomes more and more interesting Gourret et al. (1989); Bro-Nielsen et al. (1998a); Lee et al. (2009). Traditional surgical training and design depends on hands-on experience, which can be dangerous and expensive. With the help of computer animation, a *virtual surgery* can be generated that demonstrates the deformation of flesh and tissues under cutting, stretching and suturing Wilhelms and Van Gelder (1997); Marescaux et al. (1998); Szekely et al. (1998); Liu et al. (2003); Raghupathi et al. (2004); Lapeer et al. (2010). When the generated animation is reliable enough, this technique can also be used to plan novel surgeries, such as cleft lip and palate surgery Cutting et al. (2002).

In both computer animation and virtual surgery applications, the simulation of materials that emulate the exact physical behavior of human tissue and flesh are of substantial importance Bischoff et al. (2000); Teran et al. (2005a); Guo et al. (2006). Furthermore, while simulating high accuracy and highly detailed geometric models, virtual surgery needs to be operated at an interactive speed; it needs to be combined with operations like cutting and suturing and to be robust under drastic surgical manipulations Bro-Nielsen et al. (1998b); Cotin et al. (2000); Berkley et al. (2004). These requirements make the numerical problem in virtual surgery exceedingly challenging.

1.2 Numerical difficulties

High discretization resolution is required to generate accurate solutions and detailed deformations. Figure 1.1 demonstrates two simulations of the same geometric model discretized under different resolutions. When a resolution of millions degrees of freedom can be afforded, much more interesting visual effects can be generated, such as fine scale wrinkling and intricate wave propagations.

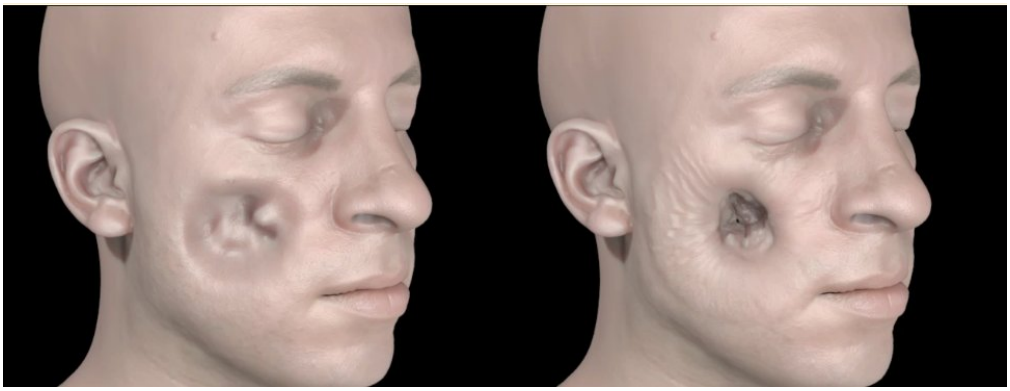


Figure 1.1: Low resolution vs high resolution. Left: a simulation with $33K$ degrees of freedom; right: the simulation of the same geometry model with $3M$ degrees of freedom.

1.2.1 Conjugate gradient method

Unlike kinematic methods, a dynamic simulation often requires the solution of a linear system. Traditional methods such as the Conjugate Gradient methods (CG) are frequently used in computer graphics and engineering, and can generate interactive simulation for problems whose number of degrees of freedom is in the low thousands. However, this is far from satisfactory for applications like virtual surgery.

Although the performance of computational hardware has been improved

drastically to support fast solution to higher resolution problems, the simulation rate does not achieve a comparable improvement. In fact, one of the key problems is that the efficiency of traditional numerical methods slows down substantially at such high resolutions. The efficiency of CG depends on the system condition number. As the resolution increases, the condition number of the linear system will also increase. The computational cost is typically $O(N^2)$, with N being the number of degrees of freedom. Notice that for 3D problems, N increases cubically with resolution. As a consequence, the number of iterations required for convergence increases to a prohibitively large number. Therefore, the efficiency of the linear algebra solver becomes the bottleneck.

1.2.2 Multigrid methods

Multigrid methods are a class of fast iterative methods that are based on a hierarchical discretization Brandt (1977a); Hackbusch and Trottenberg (1982). Geometric multigrid methods, based on a geometric multiresolution discretization, are particularly popular due to their simple implementation. In geometric multigrid methods, a numerical solver is defined on each discretization level, which is efficient in reducing error components within a particular frequency range. The composite multigrid iteration over multiple discretization levels thus reduces error components of a broad range of frequencies equally efficiently. Therefore, its efficiency is potentially independent of problem resolution. Typically, a carefully designed multigrid solver can achieve an asymptotic convergence with 80% reduction of the residual per iteration. Therefore, multigrid methods have a linearly increasing, i.e. $O(N)$, complexity under refinement. Moreover, solutions at different levels of details can be monitored during the solution process, thus offering richer information for design and control in animation production.

However, specific design is needed in order to achieve this theoretical efficiency for practical problems. The efficiency of a geometric multigrid solver depends on the compositive functioning of both an efficient single-level solver and the complete covering of error components by the discrete solutions across the multiple levels. The influence of the geometric discrepancy between different levels, the accuracy of geometric coarsening and the efficiency of a single level solver needs to be investigated for specific problems.

In particular, a typical difficulty arises when applying a general multigrid problem to arbitrary boundary conditions. An inefficient boundary smoother may reduce the convergence rate significantly. Although boundary regions are asymptotically zero, in practical applications, they can by no means be ignored. Moreover, for medium resolution problems, the extra effort on the boundary band may increase the computation cost per iteration such that multigrid method can be slower than traditional CG type methods.

Also, a multigrid method requires a single level solver that is efficient in reducing oscillatory errors with frequency commensurate with the grid resolution, thus the solver is also called a *smoother*. For elliptic equations, most iterative solvers are good smoothers. However, for non-elliptic equations or problems that are nearly non-elliptic, such as the near-incompressible elasticity problem, it is tricky to find an efficient smoother. And the multigrid method fails to be efficient.

Unlike geometric multigrid methods, the algebraic multigrid methods, generalize the multilevel idea to general sparse linear algebra systems. Algebraic multigrid methods can be conveniently defined on unstructured grids and irregular problems. However, the nontrivial multiresolution discretizations and the complicated relationship between different levels requires extra computation. This drastically increases the complexity of each iteration, especially, when the

linear system needs to be frequently modified. Also, without grid regularity, significant memory is required to represent the mesh topology.

Multigrid methods have been generalized to nonlinear problems. In continuum mechanics simulations, nonlinear constitutive models account for important effects. An intrinsically rotation invariant model has to be nonlinear, and the force response under large deformations presents obvious nonlinear behavior.

An important feature of multigrid methods is the high parallelizability. Parallelizing a numerical algorithms using multiple processors typically could increase the efficiency of the existing numerical methods. Although a parallel implementation for general large scale sparse matrices have been well explored in Dongarra and Kontoghiorghes (2001); Bolz et al. (2003); Hughes et al. (2007), the improvement in efficiency from parallel implementation can be limited by the speed of data transfer, the usage efficiency of the cached data and the complexity in developing a parallel instructions. The components of a multigrid method are typically localized and simple operations. Thus, the implementation of a multigrid method can easily be parallelized using a domain decomposition method.

1.3 Thesis structures

This thesis focuses on efficient multigrid methods for solids simulation that are characterized by :

1. their efficiency for the dynamic simulation of linear and co-rotational elasticity materials, for the entire range from compressible to highly incompressible materials,

2. the accommodation of complicated geometry and boundary conditions,

3. the demonstrated uniform high efficiency for problem-sets with up to millions of degrees of freedom and the demonstrated favorable scalability,

4. implementability on multi-threaded SMP platforms,

We also introduce a second-order boundary discretization based on the mixed finite element method. The discretized system can be solved efficiently using multigrid methods. Our method can also efficiently resolve elasticity problems with soft constraints Zhu et al. (2010).

This thesis is structured as follows. The next chapter presents a brief survey of related works emphasizing solids simulation applications in computer graphics and virtual surgery as well as existing numerical methods for acceleration. We also discuss recent applications of multigrid methods and mixed finite element methods and their advantages and disadvantages. Subsequently, we restate some mathematical background about hyperelasticity, in particular linear elasticity, as well as some existing discretization methods and linear algebra solvers. The multigrid correction scheme is also briefly explained. In Chapter 4, an augmented linear elasticity formulation is introduced with a finite difference discretization. A distributive relaxation together with a full multigrid solver is designed based on this discretization. In Chapter 5, we specifically discuss a new finite difference boundary discretization that generates an efficient multigrid solver for practical applications. We also investigate the accuracy of our method and discuss parallel implementation and results. Following that, we introduce a new augmented formulation for co-rotational elasticity and dynamic problems. In Chapter 7, a mixed finite element method on a staggered grid is developed for the linear elasticity problem, achieving a second-order accuracy for arbitrary geometric domains even under high incompressibility. Finally, a soft constraint is incorporated to the elasticity system, and a multigrid method is developed allowing us to efficiently

solve the constraint system and use that to resolve collision between deformable solids.

CHAPTER 2

Related works

The deformation of elastic solids is an interesting topic in computer animation, science and engineering. In this chapter, I will briefly introduce some previous works on deformable solids, particularly in computer graphics, I will also introduce some related work on the frequently used discretization methods for solids simulation, existing challenges and difficulties and the acceleration methods. For more complete reviews, please refer to Gibson and Mirtich (1997); Nealen et al. (2006). In particular, I will introduce recent applications of multigrid methods particularly in computer graphics. Finally, I will introduce some related works on mixed finite element methods with emphasis on their advantages and major challenges, as well as their applications in elasticity simulation.

2.1 Deformation models

In computer graphics, geometric deformation and physics-based simulation are two important classes of deformation problems. In geometric deformation, the solids or geometry mesh is deformed based on interpolations or according to geometric features. Such deformation is not necessarily consistent to the deformation of a practical volumetric material. Physics-based simulation, on the other hand, solves for the deformation that fits real world materials, generating more realistic effects.

Geometric deformation methods were developed earlier than physics-based approaches. With a high controllability and a fast performance, geometric deformation is still useful for many applications. Also, since its development benefitted physics-based simulation, we will first review several geometric deformation works that are related to physics-based simulation as well.

Geometric deformation: Free form deformation is a class of deformations that can be represented by an interpolation among some control points using Bernstein polynomials Barr (1981); Sederberg and Parry (1986); Singh and Fiume (1998). Due to the fact that the free form deformation connects the spatial deformation with a set of control points, which can be resolved with minimum computational work, the free form deformation can be easily controlled by artists, and are favorable in computer animation. Moreover, techniques have been developed to improve the performance and expand its applications. Hsu et al. (1992) developed a method to optimize the control points directly from a deformed mesh. And Rappoport et. al. propose a volume-preserving free-form deformation method in Rappoport et al. (1995). A space-warping method is introduced Barr (1984); Milliron et al. (2002) in which more complex deformation can be represented as a composition of rigid motion, tapering and bending.

Mass-spring models: Mass-spring models are a kind of physics-based methods for simulating deformable solids Platt (1992); Baraff and Witkin (1998); Vassilev and Spanlang (2002). The motion of each mass point is driven by connected spring forces. This model is useful in cloth simulation Baraff and Witkin (1998), hair simulation Selle et al. (2008), facial modelling Terzopoulos and Waters (1990), and muscle simulation Nedel and Thalmann (1998). The implementation of a mass-spring model on the GPU is also investigated in Georgii and Westermann (2005).

The mass-spring model, as an approximation to the elasticity of materials that try to preserve their original shape, achieves visually realistic deformation with fast simulation rate. However, mass-spring models suffer from instability issues at high stiffness. Also, incompressible effects are not naturally resolved in mass-spring systems. Techniques such as introducing volume preservation forces are investigated to compensate, as was done for face simulation in Lee et al. (1995). Although mass-spring model is generally not a continuum model, techniques are developed to generate consistent solutions under refinement Hutchinson et al. (1996); Debunne et al. (2000); Wu et al. (2001). Material mass and spring stiffness needs to be adjusted and even that may not be sufficient to avoid artifacts under refinement.

In general, the elasticity coefficients of the spring system are generated in an intuitive way according to practical experiences.

Lagrangian mechanics: The simulation of deformable solids was originally introduced to computer graphics by Terzopoulos et al. (1987); Terzopoulos and Fleischer (1988a,b), where time dependent equations are developed based on Lagrangian mechanics. Dynamic effects are naturally resolved by solving the derived equations, and integrating the forces and velocities over time. With the help of an implicit time integrator, the discretized system is stable even under large time steps. A method based on Gauss' principle of least constraint was also introduced in Baraff and Witkin (1992), albeit by introducing a more complicated system which is much less popular.

A feature of Lagrangian mechanics is its convenient combination with different geometric models. The free-form deformation requires minimum computational effort to generate natural dynamics Faloutsos et al. (1997). The Lagrangian mechanics was also applied by NURBS geometry Terzopoulos and Qin (1994),

quadratic functions Baraff and Witkin (1992) and finite elements Capell et al. (2002a).

Continuum mechanics constitutive models:

In continuum mechanics, the deformation is defined as a function of the material coordinates. A strain is defined to measure the distortion of the deformation, and the elasticity stress is dependent on the strain. As a consequence, the elasticity force composed from the elasticity stress has an effect to reduce the strain. The elasticity force generates the material acceleration, which formulates a partial differential equation. The equation can be discretized to a finite dimensional system, whose solution converges to the continuum solution under refinement. Therefore, the continuous constitutive models are natural for multiresolution methods. Also, in continuum mechanics, the material parameters can be estimated directly from its physics properties. Therefore, they simulate the real world materials more accurately than mass-spring models and are also more meaningful in engineering. For example, in Koch et al. (1996), the facial geometry and material parameters are estimated from CT data.

The deformation distortion is evaluated by a metric of the Green strain tensor. Picinbono et al. (2005); Barbič and James (2005), which is a nonlinear function, leading to nonlinear equations to solve. The solution to these nonlinear equations can be approximated by the solution of the linearized system at the current solution Terzopoulos et al. (1987); Debunne et al. (2001). More accurate solutions can be computed by solving a sequence of linearized systems using e.g. Newton-Raphson method Teran et al. (2005b).

When the deformation is small, a linear strain is employed as a good approximation to simplify the numerical problems. In a simplest constitutive model, i.e. the linear elasticity model, the stress is a linear function of the linear strain,

and the solution well approximates small deformations James and Pai (1999); Cotin et al. (1996). However, this approximation is not valid under large deformation. As a consequence, the solutions suffer from obvious artifacts. Moreover, linear elasticity does not keep an arbitrary constant rotation solution, which is unacceptable in computer animation.

More accurate models for large deformations, are characterized by nonlinear constitutive models (see Bonet and Wood (1997); Barbič and James (2005)) that are hard to solve Metaxas and Terzopoulos (1992); Terzopoulos and Witkin (1988). Fortunately, efforts have been made to avoid these artifacts while making use of the simplicity of linear elasticity. In Baraff and Witkin (1992), the authors eliminate global rotation and compute linear elastic stress in the rotated local coordinates, which leads to a simplified nonlinear model that avoids linear elasticity artifacts. This idea is generalized to deformations with spatially varying rotations, given prior information is available in Capell et al. (2002b). In particular, the authors designed a tool for users to paint the weights denoting the influence of each kinematic bone on all control vertices. A rotated linear elastic stress on each rotated coordinate is thus computed. The final results are blended from resolved deformations under different rotational coordinates. Non-trivial prior information is required, which will be more complicated for large models and deformations with more details. Co-rotational elasticity, introduced in Müller et al. (2002), further generalizes this idea by defining the continuous rotation field from the rotation component of the deformation gradient via a polar decomposition. It is a simple non-linear model that avoids linear elasticity artifacts and maintains the constant rotation solution. No prior information is required, but a nonlinear problem needs to be solved. Although co-rotational elasticity is not a realistic model, it generates much better visual results without extensive increase in computational effort, and the constant rotation solution is maintained, which

is desirable for computer animation Hauth and Strasser (2004); Müller and Gross (2004); Georgii and Westermann (2008, 2006); Thomaszewski et al. (2006).

Furthermore, significant efforts have been made on the simulation of more accurate hyperelasticity models, often characterized by their nonlinearity and anisotropy Fung (1993); O'Brien and Hodgins (1999a); Cotin et al. (2002); Teran et al. (2003); Müller et al. (2005b); Nealen et al. (2006). And Irving et al. (2004) modifies the Neo-Hookean model by extending its definition to inverted elements, and generates reasonable response even under extreme deformations with inverted elements.

Incompressible materials are known to suffer from locking phenomena Hansbo and Larson (2002). A volume preserving method is proposed by applying an artificial gradient vector field in Irving et al. (2007). Also, incompatible modes are introduced in Moita and Crisfield (1996) to avoid locking phenomena. And discrete Galerkin method circumvents locking Kaufmann et al. (2009a), but at the cost of at least a duplicated number of independent variables.

2.2 Discretizations

Finite difference methods: A differential operator in a PDE can be approximated with a finite difference scheme. Finite difference methods were used in the pioneering work of Terzopoulos et al. (1987) to discretize the elastic energy. Although the generation of higher-order finite difference schemes and finite difference schemes on unstructured grid is nontrivial, the finite difference method is favorable for structured grids and simpler geometric models due to the uniform formulation and the saving in memory bandwidth.

Finite element methods: Finite element methods are one of the stan-

dard discretization methods in scientific computing and mechanical engineering, in which the weak solutions are resolved in a finite dimensional solution space. A good reference for finite element methods for the solution of continuum mechanical systems may be found in Bonet and Wood (1997). Finite elements are typically mesh based methods, and tetrahedral meshes are among the most popular meshes for solids simulation Molino et al. (2003). The discrete system can be computed by first integrating the weak formula on a element basis to form an elemental system and then composing the elemental stiffness matrix to assemble a global stiffness matrix. The derived system is naturally symmetric, thus it can be solved using Conjugate Gradient methods. Other than using conformal meshes, a fictitious domain method Quarteroni and Valli (1999); Börgers and Widlund (1990) was used in Capell et al. (2002a), in which the geometric domain is embedded in a finite element mesh to simplify the boundary discretizations.

Discontinuous Galerkin finite elements are recently attracting attention Kaufmann et al. (2009a), where the deformation is defined to be discontinuous. As a consequence, it naturally resolves discontinuous solutions, and avoids locking phenomena in near-incompressible cases. Although in discontinuous Galerkin method, a larger linear system needs to be resolved, since the system is sparser then regular finite elements, the numerical solver will also be more efficient. However, when rendering the solutions, a continuous surface needs to be generated using e.g. moving-least-square methods.

Finite volume methods: Finite volume methods provide an intuitive and convenient way to discretize the force equation with an arbitrary constitutive model. Although in the case of using tetrahedron based piecewise linear solutions, the finite volume method generates an equivalent discretization to the finite element method Teran et al. (2003), it is generally different from finite

elements.

The previous three methods are typically mesh based methods. They generate sparse matrices with fixed sparsity. However, when the mesh topology changes such as with fractures, typically a remeshing is required Terzopoulos and Fleischer (1988a); O'Brien and Hodgins (1999b). Efforts are made to handle cutting problems by duplicating the elements to multiple partial elements in Sifakis et al. (2007a). However, the cut mesh may generate an ill-conditioned system and increase the problem complexity. Also, when the material is melting, demonstrating features of a viscoelastic material, the deforming mesh undertaking a large deformation and distortion may lead to ill-conditioned numerical systems. Thus, the mesh based methods are suboptimal for such cases Platt and Fleischer (1989).

Mesh free methods: Mesh free methods such as particle based methods are advantageous in handling objects with frequently changing topology. However, it is challenge to derive particle interaction to accurately mimic the material constitutive models. Nontrivial design and specific choice of the particle interaction force and smoothing kernels are required.

Desbrun et. al. extended the Smoothed Particle Hydrodynamics (SPH) for the simulation of highly deformable materials using particle systems Desbrun and Gascuel (1996). Particle based methods for the solution of partial differential equations are also investigated in solids simulation in Pauly et al. (2002a); Müller et al. (2004); Pauly et al. (2005). Materials are modeled as particles taking a volume space described by a smoothed kernel function. With this discretization, the deformation gradients can be approximated using the moving least square formulation, so that the stress and strain can be approximated for every particle. The particle based geometry representation does not require a mesh topology, there-

fore, it naturally resolves topological changing. However, without a predefined mesh topology, it requires a frequent update of the local particles information in order to avoid an expensive particle-particle interaction. Also, the moving least square method requires an matrix inversion for all particles, which is expensive. Recently, Martin et al. (2010) developed a method, where by precomputing a generalized element elaston, the elastic energy of rods, thin shells and solids can be computed in a uniform formulation, leading to a uniform treatment of the simulation for all three kinds of objects and their combinations.

2.3 Acceleration methods

Shape matching: Müller et al. (2005a) developed a simple imitation of deformable solids dynamics by estimating a rigid, linear or quadratic target deformation from the current deformations, and apply a traction force on each particle towards the targeting positions. This method is improved to a linear-time fast implementation in Rivers and James (2007), generating nice visual effects at interactive speed.

Modal analysis: Modal analysis pre-computes the singular value components of a linear system. Instead of representing the deformations using local deformation variables, the deformation is represented equivalently by the modal coordinates, which keeps the amplitude of each component. The equations for the deformations derive to an equivalent diagonal equation for the modal coordinates, which is easy to invert.

One of the challenges in modal reduction lies on the difficulty of resolving constraints. Each constraint system has a different set of modal coordinates, which can be pre-computed if static. However, certain constraints such as contact

and collision requires a frequent update of the modal coordinates, which are expensive. Also, for complex materials with large deformations, the deformation is influenced substantially by the nonlinearity; therefore, a frequent linearization and modal analysis is required. Choi and Ko (2005b) extended linear modal analysis by tracking the local rotation tensor and applying modal analysis on the local deformations, and they achieved interactive simulations with comparable visual results as the co-rotational elasticity. Barbič and James (2005) shows that by applying one Newton-Raphson iteration, the dynamic solution is good enough; therefore, they applied modal analysis on the linearized system, and achieving fast and realistic simulations for fully nonlinear models and generates nice visual effects with large deformations on medium resolution problems.

Model reduction: Model reduction methods have been developed where a few number of principle components of the possible deformations are precomputed, and a dynamic system can thus be derived for deformations of these components only. With the help of a model reduction method, geometries with several thousands degrees of freedom may be simulated at interactive speeds Zhou et al. (1996); James and Fatahalian (2003); Barbič et al. (2009). The precomputation of the principle components analysis (PCA) requires an SVD, which is very expensive. Moreover, for very high resolution problems, this is not even applicable.

Hierarchical structure and adaptivity: Hierarchical structures have been investigated in order to accelerate simulation speeds. A multigrid method is developed in Terzopoulos and Fleischer (1988b) for deformable solids simulation. Finite elements with hierarchical structures are developed in Bank (1996); Krysl et al. (2003) by refining the basis functions. Based on that, Debunne et. al. developed a simple conforming, hierarchical, adaptive refinement methods to ob-

tain a reduction of up to 80% of the variables. This method is used for thin shell and solids simulations in Debunne et al. (1999).

Uniformly refined discretizations lead to very large linear systems under high resolution, while not all of them are necessary. Adaptively refined data structures reduces the system complexity significantly without substantial reduction in the simulation quality. With the development of multiresolution geometry modeling Gortler and Cohen (1995), multiresolution discretizations especially with adaptively refined resolutions are applied to accelerate high quality simulations. In Debunne et al. (2001), mutually independent volumetric meshes of multiple resolutions are generated for a single model, where only part of the fine meshes are activated according to the local quality of simulation. Grinspun et. al. investigate the adaptive refinement from a more fundamental approach, and introduce a basis refinement method that can be generalized to high order finite elements Grinspun et al. (2002); Capell et al. (2002a). Point-based geometry with multiple resolution modeling are also investigated in Pauly et al. (2002b). Otaduy et al. (2007) proposed a multiresolution method to resolve elasticity problem with collisions. Molino et al. (2003) developed a tetrahedron meshing with adaptive refined elements; this meshing algorithm is then used in facial simulation Sifakis et al. (2007b) with 85% of reduction of the mesh size.

Parallelization: With the development of parallel computer architectures, many numerical methods can be accelerated with parallel implementation Dongarra and Kontoghiorghes (2001); Bolz et al. (2003). The acceleration to solids simulation is particularly investigated in Hughes et al. (2007); Thomaszewski et al. (2007).

Recently, techniques for general purpose GPU implementation are rapidly developing, and interest in their application to scientific computing is arising.

The power of massively parallel computing poses the problem of how to develop numerical methods that make the best use of such architectures Sørensen and Mosegaard (2006). Parallel algorithms are investigated for real time simulation with mass-spring models Georgii et al. (2005); Tejada and Ertl (2005). Recently, the implementation of multigrid methods using a finite element discretization and co-rotational elasticity model is investigated in Dick et al. (2010).

2.4 Constraints and collisions

Controlling deformations is an important problem in computer graphics and engineering. Collisions and contacts introduce interesting effects for deformable solids, and they are often modeled as constraint problems. Efficient numerical methods for constraint systems are particularly interesting in computer graphics Platt (1992); Witkin and Welch (1990); Gissler et al. (2006).

Reduced coordinate methods: For simple constraints which enforce a set of degrees of freedom, these variables can be eliminated from the system. A discretization needs to be carefully defined so that the constraint condition and corresponding variable can be easily eliminated. This method is extremely efficient when the constraints are carefully designed as in Barbič and James (2005). However, in more general cases, this method is not applicable.

Penalty methods: In penalty methods, an repulsive force is applied to prevent violation of the constraints. Penalty methods are easy to implement and accommodate to scenarios with frequently varying constraints Heidelberger et al. (2004). Therefore, they are widely used in contact and collision problems. In Terzopoulos et al. (1987), an exponential potential wall is employed to compute collision forces near an object. Equivalently, a penalty force is applied to push

the solution away from non-admissible configurations. Although the solution may not satisfy the constraint exactly, visually appealing results are generated. However, when the object has a large velocity, the penalty force needs to be very stiff to prevent the violation of constraints. Thus, the penalty methods require a small discretization resolution in space and time to avoid unstable solutions, which presents numerical inefficiency and difficulty.

Lagrange multipliers method: In more general cases, the constraints are proposed as additional equations. Such constraints can be equally satisfied by applying generalized forces that do not change the system energy. By resolving the state vector together with these extra forces, an augmented system is generated with the additional force variables of typically the same number as the constraints and additional constraint equations. These forces are also known as Lagrange multipliers. The augmented system keeps the system symmetry and definiteness, thus, the numerical algorithms for the original problems are typically applicable to the constraint problems.Benzi et al. (2005). Lagrange multipliers are widely used in motion control, collision and rigging problems Witkin and Welch (1990); Metaxas and Terzopoulos (1992); Cohen (1992); Heidelberger et al. (2004).

Admissible subspaces: Many methods have been developed to resolve constraints accurately without directly incorporating constraint conditions in the system. Typically, the path of approximated solution towards the solutions is restricted or projected to the admissible subspaces either explicitly or implicitly Müller et al. (2007). Baraff et. al. proposed a method in Baraff and Witkin (1998) for cloth simulation, where by modifying the mass matrix, the change in velocity is restricted to admissible variations.

A combination of previous methods are required for more complicated cases Isaacs and Cohen (1987); Sifakis et al. (2005). In Baraff and Witkin (1992),

the collision force is directly computed at the time step when collision happens, after which during the less drastic contact stage, the deformations are solved in the admissible subspaces prescribed as a quadratic programming problem. A semi-implicit collision solver in coupling with a deformable material simulation is presented in Bridson et al. (2005); Irving et al. (2007), and combined constraint methods are developed to implement physics-based rigging in Capell et al. (2002b, 2007).

2.5 Mixed finite element

The term "mixed finite element" was originally used in the 1960's to describe systems, in which both stress and displacement are considered as primary variables. In recent decades, theoretical investigations on its convergence, approximation and stability have been established. Mixed finite element methods have also been applied to fluid mechanics problems with both velocity and pressure as the primary variables. We refer the reader to Arnold (1990) for an early review of different variational principles, the concepts of convergence, approximability, stability and their relations, and Brezzi and Fortin (1991) for a more detailed introduction of mixed finite element methods,

Mixed finite element methods have been broadly applied to elasticity simulation, geophysics and mechanics simulations, semiconductor simulation and computational fluid dynamics. They share many similar concepts and ideas, but in this thesis, we will mainly discuss mixed finite element methods for elasticity problems.

Traditional finite element methods directly solve for the displacement as the fundamental unknowns. However, under certain circumstances, it may be bene-

ficial to simultaneously solve for dependent variables including displacement and stress Arnold and Winther (2002) or pressure Stenberg and Suri (1996). Therefore, stress-displacement and pressure-displacement formulations are developed respectively. In both the mixed formulation and the primary formulation, a unique problem is proposed in equivalent ways. However, the discretized numerical problems may have different accuracy, numerical condition and stability. With certain complications, we may benefit from the introduction of auxiliary unknowns.

First, for some physics problems, stress and pressure may be more interesting. For example, in the fracture problem, accurate approximation of stress is of substantial importance Belhachmi and Tahir. Also, some of the interesting problems require the generation of higher-order finite element spaces or finite element spaces that are hard to construct. For elasticity problems, the construction of an incompressible deformation function space is not trivial. Some investigation of its approximation, such as using a divergence free function space, can be found in Ye and Hall (1997); Brenner et al. (2007); AlexanderLinke (2008); Wang et al. (2009). Special treatment is required to construct such functional spaces. The complication may also be introduced due to the loss of robustness in certain extreme situations. For the elasticity problem, at the incompressible limit, the stress will diverge. And for near incompressible problems, it will be very large, leading to a nearly singular problem that is unstable. Moreover, a locking phenomena is observed at incompressible limit, and there can be a significant decrease in the accuracy of the solution Babuška and Suri (1992). An investigation about the stability of a mixed finite element method for the near-incompressible elasticity can be found in Braess and Ming (2005). Similar issues are also observed in the simulation of flow in porous media Arbogast et al. (1996), cartilaginous tissues Kaasschieter et al. (2003) and semiconductor device modeling Chen et al. (1995);

Brezzi et al. (2005), and in geophysics Cai et al. (1997), where mixed formulations are beneficial.

However, several difficulties are associated with mixed finite element methods. First, linear systems with more degrees of freedom must be solved. Also, while energy minimization problems often lead to positive definite algebraic equations, a mixed finite element problem is often equivalent to a saddle point problem, which leads to an indefinite system. As a consequence, traditional methods such as Conjugate Gradient method and Gauss-Seidel method may fail to converge. Kui et al. (1985) investigates a method to eliminate the additional degrees of freedom while keeping the nice properties of the mixed formulation. Others have investigated preconditioned Krylov subspace methods Vassilevski and Lazarov (1996); Benzi and Golub (2005) and fast Uzawa algorithm Cao (2003); Brenner (2009).

When applying mixed finite element methods to Stokes and Navier-Stokes equations, an "inf-sup" condition or Ladyzhenskaya-Babuska-Brezzi condition needs to be satisfied in order to keep the equation stable Arnold et al. (1984). Consequently, when applying a low-order mixed finite element method, special treatment is required to avoid instability. A number of authors have addressed this issue in Boland and Nicolaides (1984); Girault and Raviart (1979); Nicolaides and Wu (1996); Bochev et al. (2007). Han et. al. introduced a new mixed finite element formulation for the Stokes equation Han and Wu (1998) and Navier-Stokes equation Han and Yan (2008). In their work, the velocity components and pressure are defined on a staggered grid as in the marker and cell method (MAC). The relationship between their method and MAC is also investigated. A stable mixed finite element method is developed for more general nonlinear hyperelasticity models in Klaas et al. (1999).

2.6 Multigrid in computer graphics

Multigrid methods Briggs et al. (2000); Trottenberg et al. (2001) have been developed as a class of fast numerical solvers for elliptic equations, with the pioneering works from Brandt (1977b); Hackbusch and Trottenberg (1982). Algebraic multigrid methods have been developed by generalizing the geometric multigrid methods to arbitrary sparse matrices Brandt (1986). And the nonlinear multigrid method is developed in Brandt (1973).

A multigrid method was originally used in Terzopoulos and Fleischer (1988b) for deformable solids simulation in physics-based computer animation. Multigrid methods were developed for deformable solids simulation Wu and Tendick (2004); Georgii and Westermann (2006); Dick et al. (2008, 2010), thin shell simulation Green et al. (2002), mesh edition Ni et al. (2004); Shi et al. (2006), image editing Kazhdan and Hoppe (2008). The mapping of simple multigrid methods on the GPU has also been investigated Goodnight et al. (2003); Bolz et al. (2003).

Multigrid methods for the solution of a mixed system are investigated in Hiptmair (1997); Arnold et al. (2000); Gopalakrishnan and Tan (2009). Distributive relaxation was first introduced for fluid simulations in Brandt and Dinar (1978a), and a theoretical analysis about convergence is established in Wittum (1989, 1990). A stable multigrid method for near-incompressible elasticity is introduced in Wieners (2000). An efficient distributive relaxation and a multigrid method for problems with dominating grad-div operators are developed and applied to Stokes and poroelasticity problems in Gaspar et al. (2008); Oosterlee and Gaspar (2008)

CHAPTER 3

Mathematics background

3.1 Linear elaticity

We represent the deformation of an elastic volumetric object using a deformation function $\boldsymbol{\phi}$ which maps any material point $\boldsymbol{X}$ of the undeformed configuration of the object, to its position $\boldsymbol{x}$ in the deformed configuration, i.e. $\boldsymbol{x} = \boldsymbol{\phi}(\boldsymbol{X})$. Deformation of an object gives rise to elastic forces Bonet and Wood (1997) which are analytically given (in divergence form) as $\boldsymbol{f} = \boldsymbol{\nabla}^T \mathbf{P}$ or, component-wise $f_i = \sum_j \partial_j P_{ij}$ where $\mathbf{P}$ is the first Piola-Kirchhoff stress tensor. The stress tensor $\mathbf{P}$ is computed from the deformation map $\boldsymbol{\phi}$. This analytic expression is known as the elastic constitutive equation. We will henceforth adopt the common conventions of using subscripts after a comma to denote partial derivatives, and omit certain summation symbols by implicitly summing over any right-hand side indices that do not appear on the left-hand side of a given equation. Consequently, the previous equation is compactly written $f_i = P_{ij,j}$. The constitutive equation of linear elasticity is

$$\mathbf{P} = 2\mu\boldsymbol{\epsilon} + \lambda \mathrm{tr}(\boldsymbol{\epsilon})\mathbf{I} \quad \text{or} \quad P_{ij} = 2\mu\epsilon_{ij} + \lambda\epsilon_{kk}\delta_{ij}. \tag{3.1}$$

In this equation, μ and λ are the Lamé parameters of the linear material, and are computed from Young's modulus E (a measure of material stiffness) and

Poisson's ratio ν (a measure of material incompressibility) as $\mu = E/(2+2\nu), \lambda = E\nu/((1+\nu)(1-2\nu))$. Also, δ_{ij} is the Kronecker delta, $\boldsymbol{\epsilon}$ is the small strain tensor

$$\boldsymbol{\epsilon} = \tfrac{1}{2}(\mathbf{F} + \mathbf{F}^T) - \mathbf{I} \quad \text{or} \quad \epsilon_{ij} = \tfrac{1}{2}(\phi_{i,j} + \phi_{j,i}) - \delta_{ij}, \tag{3.2}$$

and $\mathbf{F}$ is the deformation gradient tensor, defined as $F_{ij} = \phi_{i,j}$. Using (3.1,3.2) we derive the governing equations

$$f_i = \mu\phi_{i,jj} + (\mu + \lambda)\phi_{j,ij} = \mathcal{L}_{ij}\phi_j. \tag{3.3}$$

In this equation $\mathcal{L} = \mu\Delta\mathbf{I} + (\mu+\lambda)\boldsymbol{\nabla}\boldsymbol{\nabla}^T$ is the partial differential operator of linear elasticity. A static elasticity problem amounts to determining the deformation map $\boldsymbol{\phi}$ that leads to an equilibrium of the total forces, i.e. $\mathcal{L}\boldsymbol{\phi} + \boldsymbol{f}_{\text{ext}} = 0$, where $\boldsymbol{f}_{\text{ext}}$ are the external forces applied on the object. For simplicity, we redefine $\boldsymbol{f} = -\boldsymbol{f}_{\text{ext}}$ and the static elasticity problem becomes equivalent to the linear partial differential equation $\mathcal{L}\boldsymbol{\phi} = \boldsymbol{f}$.

3.1.1 Discretization

We discretize the elasticity problem on a regular Cartesian lattice. Our deformable model is embedded in this lattice, similar to the approach of Rivers and James (2007). Although an unstructured mesh provides more flexibility, we opted for a regular grid for economy of storage. For example, storing the topology of a tetrahedral lattice could easily require 4-5 times more than the storage required for the vertex positions, taking up valuable memory bandwidth. Additionally, the discrete PDE and transfer operators are uniform across regular grids, eliminating the need for explicit storage. Although not used in this thesis, adaptivity can also be combined with regular grids, see e.g. Brandt (1977a).

Finite element methods (FEM) are arguably the most common discretization methods for elasticity applications in graphics (see e.g. O'Brien and Hodgins (1999b)), and have been applied to hyperelastic simulations Irving et al. (2004). Finite element methods automatically generate symmetric positive definite systems that can be solved using iterative solvers such as CG method or Gauss-Seidel method. It is convenient to establish analysis for error estimation and proof of stability of FEM, and it is also convenient for developing higher-order discretizations. FEM can naturally resolve irregular geometry defined on unstructured grids, although with extra storage costs for the grid structure and indirect memory access.

Finite difference methods are another important discretization for the PDE of elasticity Terzopoulos et al. (1987). Although, it can be difficult to discretize boundary conditions and to achieve subgrid accuracy of the domain geometry. For some applications we opted for a finite difference discretization defined on uniform grid. A familiar practice in the field of computational fluid dynamics (e.g. Harlow et al. (1965)). Although far less widespread for the simulation of solids, this formulation was selected for reasons of efficiency and numerical stability.

Our method owes its good performance for highly incompressible materials to a mixed formulation of elasticity (see Chapter 4). Although it is possible to combine this formulation with finite elements (see e.g. Brezzi and Fortin (1991) and Chapter 7), it is much simpler to implement it using finite differences. For regular lattices, both finite elements and finite differences are second-order accurate discretizations away from the boundary, while both are susceptible to degrading to first order near the boundaries as discussed in section 6.3.1. In addition, our finite difference scheme leads to sparser stencils than finite elements: in our formulation of 3D linear elasticity, each equation has 15 nonzero entries, while 81

entries are required by a trilinear hexahedral finite element discretization, and 45 for BCC tetrahedral finite elements. This translates to a lower computational cost for a finite difference scheme. Finally, as part of our contribution we derive a specific finite difference scheme that guarantees the same symmetry and definiteness properties that are automatic with finite element methods.

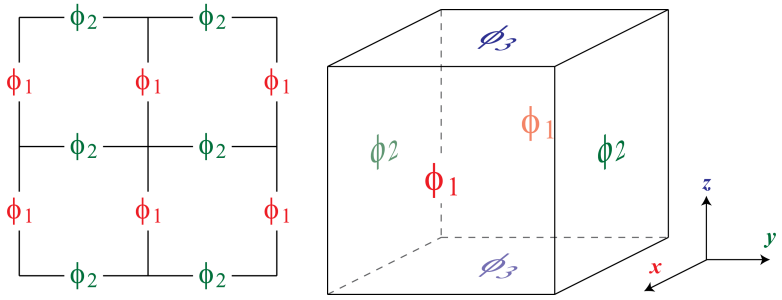


Figure 3.1: Staggering of variables in 2D(left) and 3D(right). Equations $\mathcal{L}_1$,$\mathcal{L}_2$,$\mathcal{L}_3$ are also stored on ϕ_1,ϕ_2,ϕ_3 locations respectively.

Use of staggered variables In a regular grid it would be most natural to specify all components of the vector-valued deformation map $\boldsymbol{\phi}$ at the same locations, for example at the nodes of the grid. However, for equation (3.3) doing so may result in grid-scale oscillations, especially for near-incompressible materials. This is qualitatively analogous to an artifact observed in the simulation of fluids with non-staggered grids, where spurious oscillations may be left over in the pressure field. For multigrid methods, such oscillatory modes are problematic, as they may not respect the fundamental property of elliptic PDEs that a low residual implies a smooth error, requiring more elaborate and expensive smoothers to compensate. We avoid this issue by adopting a staggered discretization (Figure 3.1), which is free of this oscillatory behavior. More specifically, ϕ_i variables are stored at the centers of grid faces perpendicular to the Cartesian axis vector $\boldsymbol{e}_i$. For example, ϕ_1 values are stored on grid faces perpendicular to $\boldsymbol{e}_1$, i.e. those parallel to the yz-plane. The same strategy is followed in 2D, where faces of grid

cells are now identified with grid edges, thus ϕ_1 values are stored at the center of y-oriented edges, and ϕ_2 values at the center of x-oriented edges. We define discrete first-order derivatives using central differences:

$$
\begin{aligned}
D_1 u[x,y,z] &= u[x+\tfrac{1}{2}h_x, y, z] - u[x-\tfrac{1}{2}h_x, y, z] \\
D_2 u[x,y,z] &= u[x, y+\tfrac{1}{2}h_y, z] - u[x, y-\tfrac{1}{2}h_y, z] \\
D_3 u[x,y,z] &= u[x, y, z+\tfrac{1}{2}h_z] - u[x, y, z-\tfrac{1}{2}h_z],
\end{aligned}
$$

where (h_x, h_y, h_z) are the dimensions of the background grid cells. Second-order derivative stencils are defined as the composition of two first-order stencils, i.e. $D_{ij} = D_i D_j$. An implication of these definitions is that the discrete first derivative of a certain quantity will not be collocated with it. For example, all derivatives of the form $D_i \phi_i$ are naturally defined at cell centers, while $D_1 \phi_2$ is located at centers of z-oriented edges in 3D, and at grid nodes in 2D. However, derivatives are centered at the appropriate locations for a convenient discretization of (3.3). In particular, all stencils involved in the discretization of equation $\mathcal{L}_i$ are naturally centered on the location of variable ϕ_i. Thus, the staggering of the unknown variables implies a natural staggering of the discretized differential equations. Figure 3.2 illustrates this fact in 2D, where the discrete stencils for the operators $\mathcal{L}_1$ and $\mathcal{L}_2$ from (3.3) are shown to be naturally centered at ϕ_1 and ϕ_2 variable locations, respectively.

3.2 Multigrid correction scheme

Multigrid methods are based on the concept of a smoother which is a procedure designed to reduce the residual $\boldsymbol{r}=\boldsymbol{f}-\boldsymbol{\mathcal{L}\phi}$ of the differential equation. For example, in a discretized system, Gauss-Seidel or Jacobi iteration are common

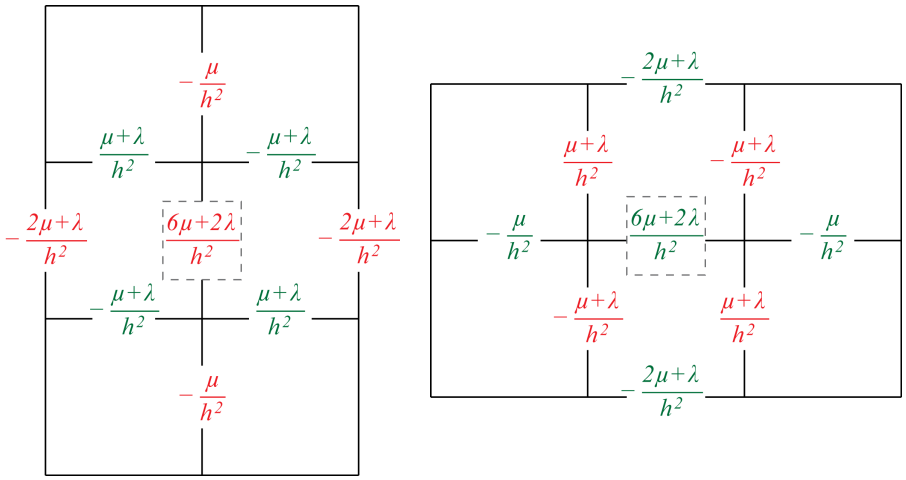


Figure 3.2: Discrete stencils for operators $\mathcal{L}_1$(left) and $\mathcal{L}_2$(right) of the PDE system (3.3). The red and green nodes of the stencil correspond to ϕ_1 and ϕ_2 values respectively. The dashed square indicates the center of the stencil, where the equation is evaluated.

Algorithm 1 Multigrid Correction Scheme – V-(1,1) Cycle

procedure MULTIGRID($\boldsymbol{\phi}, \boldsymbol{f}, L$) ▷ $\boldsymbol{\phi}$ is the current estimate
  $\boldsymbol{u}^h \leftarrow \boldsymbol{\phi}, \boldsymbol{b}^h \leftarrow \boldsymbol{f}$ ▷ total of $L+1$ levels
  for $l = 0$ **to** $L-1$ **do**
    Smooth($\mathbf{L}^{2^l h}$,$\boldsymbol{u}^{2^l h}$,$\boldsymbol{b}^{2^l h}$)
    $\boldsymbol{r}^{2^l h} \leftarrow \boldsymbol{b}^{2^l h} - \mathbf{L}^{2^l h}\boldsymbol{u}^{2^l h}$
    $\boldsymbol{b}^{2^{l+1} h} \leftarrow$Restrict($\boldsymbol{r}^{2^l h}$), $\boldsymbol{u}^{2^{l+1} h} \leftarrow 0$
  end for
  Solve $\boldsymbol{u}^{2^L h} \leftarrow (\mathbf{L}^{2^L h})^{-1}\boldsymbol{b}^{2^L h}$
  for $l = L-1$ **down to** 0 **do**
    $\boldsymbol{u}^{2^l h} \leftarrow \boldsymbol{u}^{2^l h}+$Prolongate($\boldsymbol{u}^{2^{l+1} h}$)
    Smooth($\mathbf{L}^{2^l h}$,$\boldsymbol{u}^{2^l h}$,$\boldsymbol{b}^{2^l h}$)
  end for
  $\boldsymbol{\phi} \leftarrow \boldsymbol{u}^h$
end procedure

smoothers. An inherent property of elliptic systems is that when the magnitude of the residual is small, the error $\boldsymbol{e} = \boldsymbol{\phi} - \boldsymbol{\phi}_{\text{exact}}$ is expected to be smooth Brandt (1986). Smoothers are typically simple, local and relatively inexpensive routines, which are efficient at reducing large values of the residual (and, as a consequence, eliminating high frequencies in the error). Nevertheless, once the high frequency component of the error has been eliminated, subsequent iterations are characterized by rapidly decelerated convergence. Multigrid methods seek to remediate this stagnation by using the smoother as a building block in a multi-level solver that achieves constant rate of convergence, regardless of the prevailing frequencies of the error. This is accomplished by observing that any persistent lower frequency error will appear to be higher frequency if the problem is resampled using a coarser discretization. By transitioning to ever coarser discretizations the smoother retains the ability to make progress towards convergence. The components of a multigrid solver are:

- Discretizations of the continuous operator $\mathcal{L}$ at a number of different resolutions, denoted as $\mathbf{L}^h, \mathbf{L}^{2h}, \mathbf{L}^{4h}$ etc. (where the superscripts indicate the mesh size for each resolution).
- Smoothing subroutine, defined at each resolution.
- Prolongation and Restriction subroutines. These implement an up-sampling and downsampling operation respectively, between two different levels of resolution.
- An exact solver, used for solving the discrete equations at the coarsest level. As the coarse grid is expected to be small, any reasonable solver would be an acceptable option.

Algorithm 1 gives the pseudocode for a V-(1,1) cycle of the Multigrid correction scheme, which is the method used in this thesis.

3.3 Multigrid methods for linear elasticity

We present an application of general multigrid methods to a linear elasticity problem. Consider a linear elasticity material with undeformed configuration being a cubic domain $[0,1]^3$. We discretize linear elasticity using finite element method defined on a BCC lattice generated from a regular grid of n^3 defined on the domain. The grid points are (ih, jh) with $i, j \in \{0, 1, ..., n-1\}$ and $h = 1/n$. Let us consider a problem with fixed boundary conditions on the left and right side of the cube, and free surface boundary conditions on the other four sides, i.e. $\mathbf{PN} = 0$, where $\mathbf{N}$ is the surface normal unit vector in undeformed configuration (see Figure 3.3). We discretize the system using a piecewise linear finite element method applied on the tetrahedral mesh. Each coarse grid problem is defined by the same algorithm but discretized on a grid of $(n/2)^3$. And we apply full weighting restriction and bi-linear interpolation for prolongation. For the relaxation, we choose a Gauss-Siedel relaxation in a lexicographical ordering, and we applied 10 boundary relaxations near the domain boundary.

With a low compressibility of Poisson's ratio being 0.2, we achieve a fast convergence with a rate of 0.2. But for high incompressible material with Poisson's ratio being 0.49, the multigrid efficiency is substantially decreased. The asymptotic convergence rate is greater than 0.9. For the purpose of a clear demonstration, we rendered the deformed object with a cutaway at a center cross section in Figure 3.4. The deformation is derived from the approximated solution after 5 multigrid V-(1,1) iterations with the same random initial guess and different

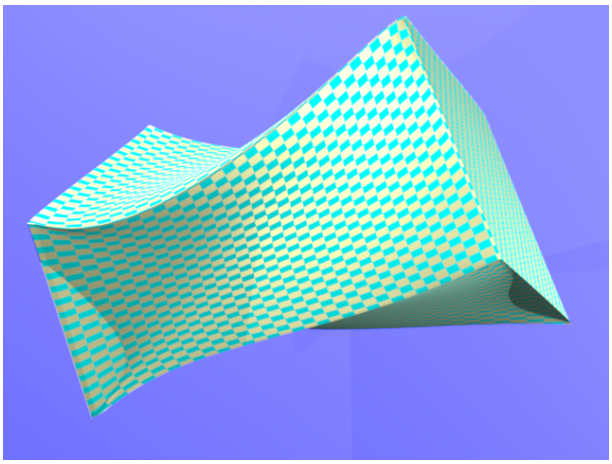


Figure 3.3: Linear elasticity example: a deformed cubic elastic material

Poisson's ratios.

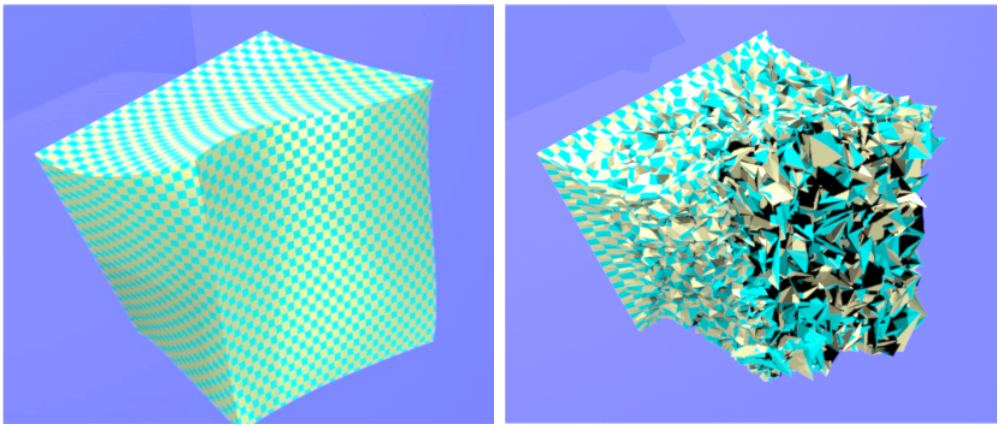


Figure 3.4: Comparison of multigrid convergence with different Poisson's ratios. We apply the same boundary condition and start with the same random initial guess and plot the deformed object after 5 multigrid V-(1,1) cycles. Left: Poisson's ratio is 0.2; right: Poisson's ratio is 0.49.

CHAPTER 4

Augmented linear elasticity

A majority of elastic materials of interest to computer graphics (e.g. the muscles and flesh of animated characters) and material science are ideally incompressible. For a multigrid solver, naive use of standard smoothers (e.g. Gauss-Seidel) in the presence of high incompressibility could lead to slow convergence or even loss of stability.

In this chapter, we introduced an augmented linear elasticity system discretized with a finite difference method and a distributive relaxation as a stable solver and efficient smoother for the derived linear system. The relaxation method is computationally inexpensive and leads to fast multigrid convergence independent of material parameters.

4.1 Finite difference discretization

When the Poisson's ratio approaches the incompressible limit $\nu \to 0.5$, the Lamé parameter λ becomes several orders of magnitude larger than μ. As a consequence, the dominant term of the elasticity operator $\mathcal{L} = \mu\Delta\mathbf{I}+(\mu+\lambda)\boldsymbol{\nabla}\boldsymbol{\nabla}^T$ is the rank deficient operator $(\mu+\lambda)\boldsymbol{\nabla}\boldsymbol{\nabla}^T$; thus $\mathcal{L}$ becomes near-singular. More

specifically, we see that any divergence-free field

$$\boldsymbol{\phi} = \begin{pmatrix} \cos kx \sin ky \\ -\sin kx \cos ky \end{pmatrix} \tag{4.1}$$

will be in the null-space of the dominant term, i.e. $\lambda \boldsymbol{\nabla}\boldsymbol{\nabla}^T\boldsymbol{\phi} = 0$. Thus, a solution to the elasticity PDE $\mathcal{L}\boldsymbol{\phi} = \boldsymbol{f}$ could be perturbed by a divergence-free displacement of substantial amplitude, without introducing a large residual for the differential equation. These perturbations can be arbitrarily oscillatory with large number of k, and lead to high-frequency errors that the multigrid method cannot smooth efficiently or correct using information from a coarser grid. Fortunately, this complication is not a result of inherently problematic material behavior, but rather an artifact of the form of the governing equations. Our solution is to reformulate the PDEs of elasticity into an equivalent system, which does not suffer from the near-singularity of the original. This stable differential description of near-incompressible elasticity is adapted from the theory of mixed variational formulations Brezzi and Fortin (1991). We introduce a new auxiliary variable p (which we call *pressure*) defined as $p = -(\lambda/\mu)\boldsymbol{\nabla}^T\boldsymbol{\phi} = -(\lambda/\mu)\mathbf{div}\boldsymbol{\phi}$. We can write

$$\begin{aligned} \mathcal{L}\boldsymbol{\phi} &= \mu(\Delta\mathbf{I} + \boldsymbol{\nabla}\boldsymbol{\nabla}^T)\boldsymbol{\phi} + \lambda\boldsymbol{\nabla}(\boldsymbol{\nabla}^T\boldsymbol{\phi}) \\ &= \mu(\Delta\mathbf{I} + \boldsymbol{\nabla}\boldsymbol{\nabla}^T)\boldsymbol{\phi} - \mu\boldsymbol{\nabla}p. \end{aligned} \tag{4.2}$$

Thus, the equilibrium equation $\mathcal{L}\boldsymbol{\phi} = \boldsymbol{f}$ is equivalently written as

$$\begin{pmatrix} \mu(\Delta\mathbf{I}+\boldsymbol{\nabla}\boldsymbol{\nabla}^T) & -\mu\boldsymbol{\nabla} \\ \mu\boldsymbol{\nabla}^T & \frac{\mu^2}{\lambda} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi} \\ p \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix}. \tag{4.3}$$

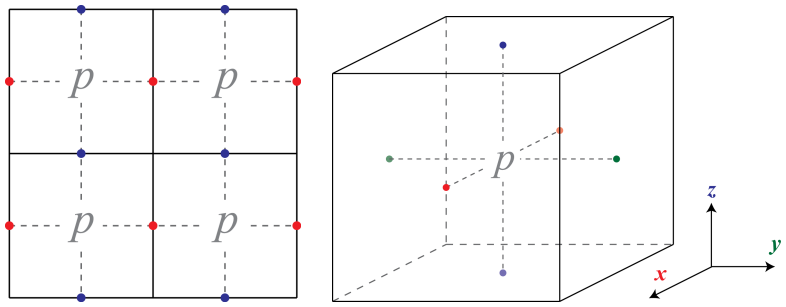


Figure 4.1: Placement of pressures in 2D (left) and 3D (right).

Let us define $\hat{\mathcal{L}}$ to denote the augmented differential operator, define $\hat{\boldsymbol{\phi}} = (\boldsymbol{\phi}, p)$ to denote the augmented unknowns and define $\hat{\boldsymbol{f}} = (\boldsymbol{f}, 0)$ to denote the right-hand side vector. Then the augmented system is equivalently written as $\hat{\mathcal{L}}\hat{\boldsymbol{\phi}} = \hat{\boldsymbol{f}}$. The top equation of system (4.3) follows directly from equation (4.2), while the bottom is the definition of pressure p. Conversely, the original differential equation (3.3) can be obtained from (4.3) by simply eliminating the pressure variable. Thus the augmented differential equation system of (4.3) is equivalent to the governing equations of linear elasticity. The important consequence of this manipulation is that this new discretization is stable, in the sense that the system can be smoothed with standard methods without leaving spurious oscillatory modes. This property can be rigorously proved via Fourier analysis; we can verify however that as λ tends to infinity, the term μ^2/λ vanishes, and the resulting limit system is now non-singular. In section 4.2 we describe a simple smoother, specifically tailored to equation (4.3). The newly introduced pressure variables are also discretized on an offset Cartesian lattice, with pressures stored in cell centers (see Figure 4.1). Pressure equations are also cell centered. As was the case with the non-augmented elasticity equations, the staggering of deformation ($\boldsymbol{\phi}$) and pressure (p) variables is such that all centered difference discrete operators are well defined where they are needed, and the componentwise stencil of each operators in $\hat{\mathcal{L}}$is demonstrated in 4.2.

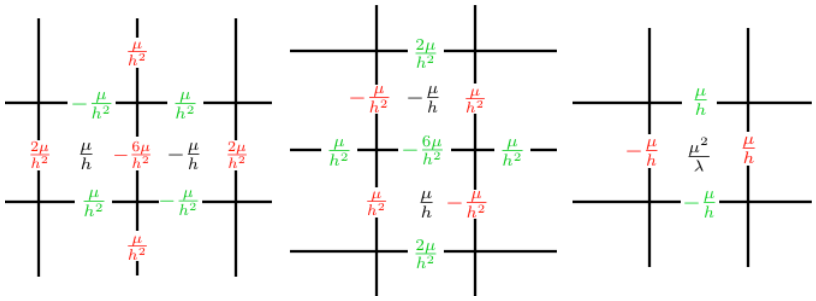


Figure 4.2: Discrete stencils for operators in $\hat{\mathcal{L}}$ for ϕ_1, ϕ_2 and p respectively in the left, middle and right figures.

4.2 Distributive relaxation

The discretization of system (4.3) yields a symmetric, yet indefinite matrix (discrete first order derivatives are skew-symmetric). Although this system has the stability to admit efficient local smoothing, the convergence of the solution cannot be accomplished with a standard Gauss-Seidel or Jacobi iteration. Additionally, for a differential equation such as (4.3) exhibiting nontrivial coupling between the variables ϕ_1, ϕ_2, ϕ_3 and p, a smoothing scheme which smoothes a given equation by updating several variables at once is often the optimal choice in terms of efficiency Trottenberg et al. (2001). The technique we use in our formulation is the distributive smoothing approach. This technique was applied to the Stokes equation in Brandt and Dinar (1978b) while Gaspar et al. (2008) discussed its application to linear elasticity. Consider the change of variables

$$\begin{pmatrix} \boldsymbol{\phi} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{I} & -\boldsymbol{\nabla} \\ \boldsymbol{\nabla}^T & -2\Delta \end{pmatrix} \begin{pmatrix} \boldsymbol{\psi} \\ q \end{pmatrix} \quad \text{or} \quad \hat{\boldsymbol{\phi}} = \hat{\mathcal{M}}\hat{\boldsymbol{\psi}}, \tag{4.4}$$

where $\hat{\boldsymbol{\psi}} = (\boldsymbol{\psi}, q)$ is the vector of auxiliary unknown variables, and $\hat{\mathcal{M}}$ is called the distribution matrix. Using the change of variables of equation (4.4), our augmented system $\hat{\mathcal{L}}\hat{\boldsymbol{\phi}} = \hat{\boldsymbol{f}}$ is equivalently written as $\hat{\mathcal{L}}\hat{\mathcal{M}}\hat{\boldsymbol{\psi}} = \hat{\boldsymbol{f}}$. Composing

the operators $\hat{\mathcal{L}}$ and $\hat{\mathcal{M}}$ yields

$$\hat{\mathcal{L}}\hat{\mathcal{M}} = \begin{pmatrix} \mu\Delta\mathbf{I} & 0 \\ \mu(1+\frac{\mu}{\lambda})\boldsymbol{\nabla}^T & -\mu(1+\frac{2\mu}{\lambda})\Delta \end{pmatrix}. \tag{4.5}$$

That is, the composed system is lower triangular, and its diagonal elements are simply Laplacian operators. This system can be smoothed with any scheme that works for the Poisson equation, including the Gauss-Seidel or Jacobi methods. In fact, the entire system can be smoothed with the efficiency of the Poisson equation, following a forward substitution approach, i.e. we smooth all ψ_1-centered equations across the domain first, followed by sweeps of ψ_2, ψ_3, and q-centered equations in sequence. After we solved ψ and q, we can resolve the solution by substituting into (4.4).

In implementation, we first discretize the auxiliary variable and the distribution matrix. In accordance with our staggered formulation, the components $\psi_1^h, \psi_2^h, \psi_3^h$ of the auxiliary vector $\boldsymbol{\psi^h}$ will be collocated with $\phi_1^h, \phi_2^h, \phi_3^h$ respectively, while q^h and p^h values are also collocated (see Figure 4.3 for 2D case). We further discretize the distribution matrix using a centered finite difference scheme. It is important to notice that while $\hat{\mathcal{L}}\hat{\mathcal{M}}$ is an lower triangular differential operator matrix, $\hat{\mathbf{L}}\hat{\mathbf{M}}$, i.e. the discrete matrix is also block lower triangular, hence can be solved using forward substitution.

More importantly, in practice, we do not necessarily build the auxiliary variables $(\boldsymbol{\psi}, q)$. Consider the Gauss-Seidel iteration for the discretized system $\hat{\mathbf{L}}\hat{\boldsymbol{\phi}} = \hat{\boldsymbol{f}}$: At every step, we calculate a point-wise correction to the variable $\hat{\phi}_i$, such that the residual of the collocated equation $\hat{\mathbf{L}}_i$ will vanish. That is, we replace

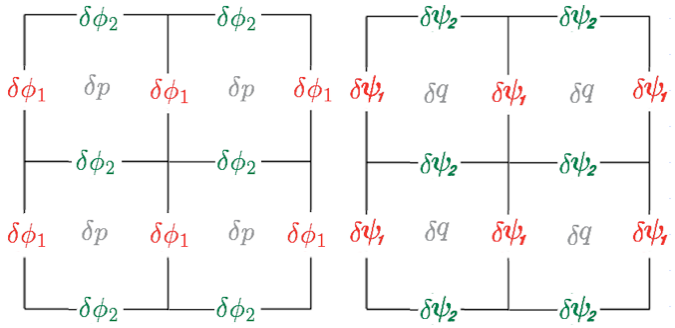


Figure 4.3: Location of auxiliary variables and the distribution stencils - 2D.

variable $\hat{\phi}_i$ with $\hat{u}_i + \delta$ (or $\hat{\boldsymbol{\phi}}$ with $\hat{\boldsymbol{\phi}} + \delta \boldsymbol{e}_i$) such that:

$$\boldsymbol{e}_i^T(\hat{\boldsymbol{f}} - \hat{\mathbf{L}}(\hat{\boldsymbol{\phi}} + \delta \boldsymbol{e}_i)) = 0 \Rightarrow (\boldsymbol{e}_i^T \hat{\mathbf{L}} \boldsymbol{e}_i)\delta = \boldsymbol{e}_i^T(\hat{\boldsymbol{f}} - \hat{\mathbf{L}}\hat{\boldsymbol{\phi}}).$$

The last equation is equivalent to $\hat{\mathbf{L}}_{ii}\delta = r_i^{\text{old}}$ or $\delta = r_i^{\text{old}}/\hat{\mathbf{L}}_{ii}$, where $\hat{\mathbf{L}}_{ii}$ is the i-th diagonal element of the discrete operator and r_i^{old} denotes the i-th component of the residual. In an analogous fashion, a Gauss-Seidel step on the distributed system $\hat{\mathbf{L}}\hat{\mathbf{M}}\hat{\boldsymbol{\psi}} = \hat{\boldsymbol{f}}$ amounts to changing $\hat{\psi}_i$ into $\hat{\psi}_i + \delta$ (or $\hat{\boldsymbol{\psi}}$ into $\hat{\boldsymbol{\psi}} + \delta \boldsymbol{e}_i$) such that the i-th residual of the distributed equation is annihilated

$$\begin{aligned} \boldsymbol{e}_i^T(\hat{\boldsymbol{f}} - \hat{\mathbf{L}}\hat{\mathbf{M}}(\hat{\boldsymbol{\psi}} + \delta \boldsymbol{e}_i)) = 0 \Rightarrow \boldsymbol{e}_i^T(\hat{\boldsymbol{f}} - \hat{\mathbf{L}}(\hat{\boldsymbol{\phi}} + \delta\hat{\mathbf{M}}\boldsymbol{e}_i)) = 0 \\ \Rightarrow (\boldsymbol{e}_i^T \hat{\mathbf{L}}\hat{\mathbf{M}} \boldsymbol{e}_i)\delta = \boldsymbol{e}_i^T(\hat{\boldsymbol{f}} - \hat{\mathbf{L}}\hat{\boldsymbol{\phi}}) \Rightarrow \delta = r_i^{\text{old}}/(\hat{\mathbf{L}}\hat{\mathbf{M}})_{ii}. \end{aligned}$$

In this derivation, the auxiliary vector $\hat{\boldsymbol{\psi}}$ is only used in the form $\hat{\mathbf{M}}\hat{\boldsymbol{\psi}}$ which is equal to the value of the original variable $\hat{\boldsymbol{\phi}}$. Thus, after the value of δ has been determined, $\hat{\boldsymbol{\phi}}$ is updated to $\hat{\boldsymbol{\phi}} + \delta\hat{\mathbf{M}}\boldsymbol{e}_i$ (see Figure 4.4 for a 2D example). The computational cost of distributive smoothing is comparable to that of simple Gauss-Seidel iteration, yet it allows efficient smoothing of the linear elasticity equations independent of Poisson's ratio. We summarize the distributive smoothing process in Algorithm 2.

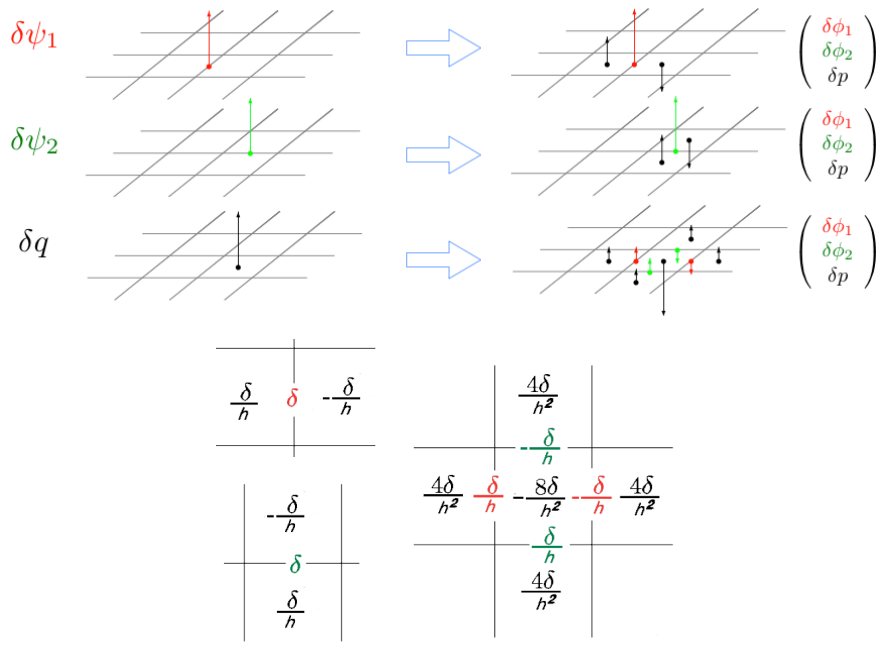


Figure 4.4: Stencils of distributions - 2D. Top: to the left is a δ-correction applied on the auxiliary variables $\hat{\boldsymbol{\psi}}$; to the right is an equivalent distributive correction applied on the original variables $\hat{\boldsymbol{\phi}}$. Bottom: distribution stencils for each correction applied on ψ_1(left top), ψ_2(left bottom) and q(right), i.e. the stencils of $m_i^T = e_i\hat{\mathbf{M}}$ in Algorithm 2 where the i-th variables are ψ_1, ψ_2 and q respectively.

Algorithm 2 Distributive Smoothing

procedure DISTRIBUTIVESMOOTHING($\hat{\mathbf{L}}$,$\hat{\mathbf{M}}$,$\hat{\boldsymbol{\phi}}$,$\hat{\boldsymbol{f}}$)
  for v **in** $\{\hat{\phi}_1, \hat{\phi}_2, \hat{\phi}_3, p\}$ **do** ▷ Must be in this exact order
    for i **in** Lattice[v] **do** ▷ i is an equation index
      $r \leftarrow \hat{f}_i - \hat{\mathbf{L}}_i \cdot \hat{\boldsymbol{\phi}}$ ▷ $\hat{\mathbf{L}}_i$ is the i-th row of $\hat{\mathbf{L}}$
      $\delta \leftarrow r/(\hat{\mathbf{L}}\hat{\mathbf{M}})_{ii}$
      $\hat{\boldsymbol{\phi}}$ += $\delta\, m_i^T$ ▷ m_i is the i-th row of $\hat{\mathbf{M}}$
    end for
  end for
end procedure

CHAPTER 5

Boundary system and geometric coarsening

The previous sections did not address the effect of boundaries, instead focusing on the treatment of the interior region. The efficiency of the interior smoother can be evaluated using a periodic domain. In fact, it is known Brandt (1994) that a boundary value problem can be solved at the same efficiency as a periodic problem, at the expense of more intensive smoothing at the boundary. In theoretical studies, the computational overhead of this additional boundary smoothing is often overlooked, as the cost of interior smoothing is *asymptotically* expected to dominate. Nevertheless, practical problem sizes may never reach the asymptotic regime and slow, generic boundary smoothers can pose a performance bottleneck. In this section, we develop a boundary discretization strategy, including a novel treatment of traction boundary conditions, that facilitates the design of efficient and inexpensive boundary smoothers.

5.1 Domain description

Our geometrical description of the computational domain is based on a partitioning of the cells of the background grid (Figure 5.1). Initially, cells that have an overlap with the simulated deformable body are characterized as *interior* cells, otherwise they are designated *exterior* cells. Additionally, any cell can be user-specified to be a constrained (or *Dirichlet*) cell, overriding any interior/exterior

designation this cell may otherwise carry. Dirichlet cells practically correspond to kinematically constrained parts of the object. This classification into interior, exterior and Dirichlet cell types provides an intuitive way to specify the degrees of freedom of our problem, and define their associated equations. We categorize discrete variables and equations as follows:

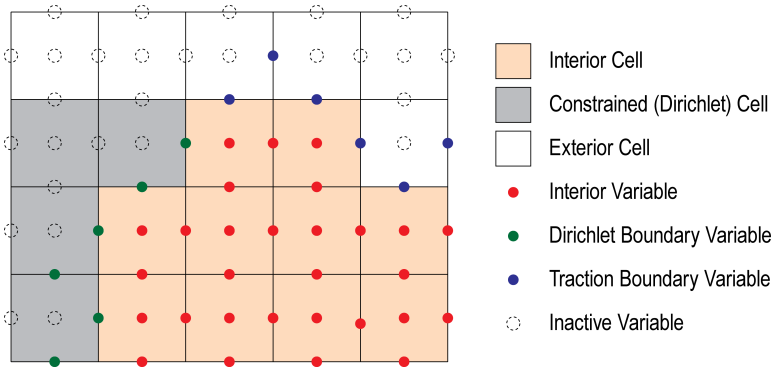


Figure 5.1: Classification of cells,variables and equations near the boundary.

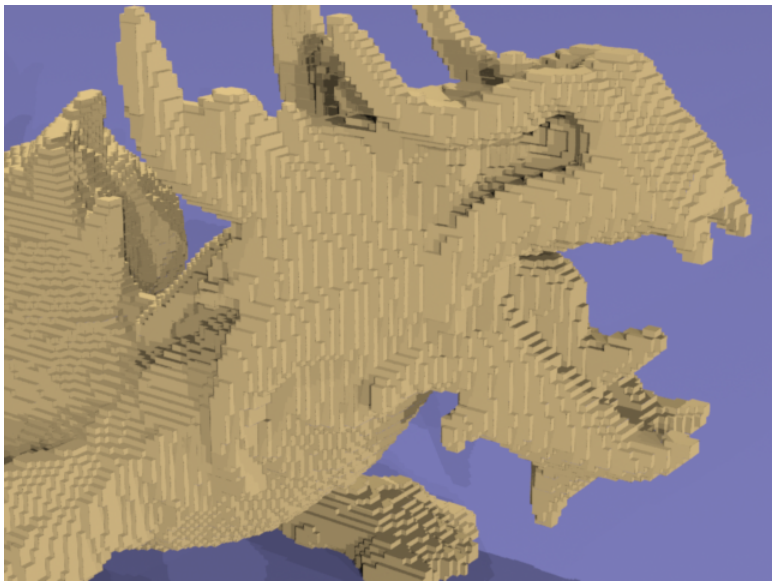


Figure 5.2: Active cells of a discretized dragon model with $402K$ cells.

- *Interior variables and equations.* Any of the variables ϕ_1, ϕ_2, ϕ_3 or p located strictly inside the interior region (i.e. either on an interior cell center, or

on the face between two interior cells) is designated an interior variable. For every interior variable, we label its collocated equation from (4.3) as an *interior equation* and we include this equation as part of our discrete system. Locations of interior variables and equations are depicted as red dots in figure 5.1.

- *Boundary variables and equations.* Certain interior equations (near the boundary) have a discrete stencil that extends onto variables that are not interior variables themselves. We label these as *boundary variables*. More specifically, a boundary variable is designated a Dirichlet boundary variable if it touches a Dirichlet cell (either inside or on the boundary of one), otherwise it is designated a traction boundary variable. Dirichlet and traction boundary variables are depicted as green and blue dots respectively, in figure 5.1. Similar to interior variables, for every boundary variable we add a boundary equation (or *boundary condition*) to our discrete system, in order to have the same number of equations as unknowns. Dirichlet variables are matched with a boundary condition of the form $\boldsymbol{\phi}(\boldsymbol{X}) = \boldsymbol{c}$, while traction variables are associated with a condition of the form $\mathbf{P}(\boldsymbol{X})\boldsymbol{N} = \boldsymbol{t}$, where $\boldsymbol{N}$ is the surface normal ($\boldsymbol{t} = 0$ corresponds to a free boundary).

- Variables that have not been designated interior or boundary are labeled *inactive* and can be ignored (depicted as dashed circles in figure 5.1). No equation is added to our system for these inactive grid locations.

5.2 A general-purpose box smoother

Although a well-posed system can be constructed as described, the distributive smoothing scheme is not valid near the boundary, as the distribution extends

outside the domain. In such situations a box smoother Brandt and Dinar (1978b) is a broadly applicable solution. This process amounts to collectively solving a number of equations in a subdomain, simultaneously adjusting the values of all variables within. Our complete smoothing procedure starts with a boundary box smoothing sweep, proceeds with interior distributive smoothing and finishes with a last boundary pass. An interior equation is smoothed distributively if the stencil of its respective equation in the composed system $(\hat{\mathcal{L}}\hat{\mathcal{M}})\hat{\boldsymbol{\psi}} = \hat{\boldsymbol{f}}$ only includes interior variables, as illustrated in Figure 5.3 (left). For the boundary, we use overlapping boxes that are two grid cells wide, and centered at the outermost layer of interior cells, as seen in Figure 5.3 (right). In our experiments the box smoother performed very well, generally achieving near-optimal efficiency for the entire multigrid scheme. In practice, however, this good convergence behavior came at the cost of an enormous computational overhead. This added cost stems from the need to solve a coupled linear system within each box. The computational effort spent on boundary smoothing was often two orders of magnitude more than the cost of interior smoothing on models with tens of thousands of vertices; although the interior cost scales with volume and the boundary cost scales with surface area, even with million-vertex models the cost of the boundary smoother would still dominate by a factor of 10. We address this issue in the next section by designing an effective, yet simple and inexpensive boundary smoother.

5.3 A fast symmetric Gauss-Seidel smoother

We propose a novel formulation that enables equation-by-equation smoothing that is both efficient and inexpensive. The main obstacle to efficient equation-by-equation boundary smoothing schemes, is lack of symmetry, definiteness or diagonal dominance. Additionally, discretizations of the boundary conditions

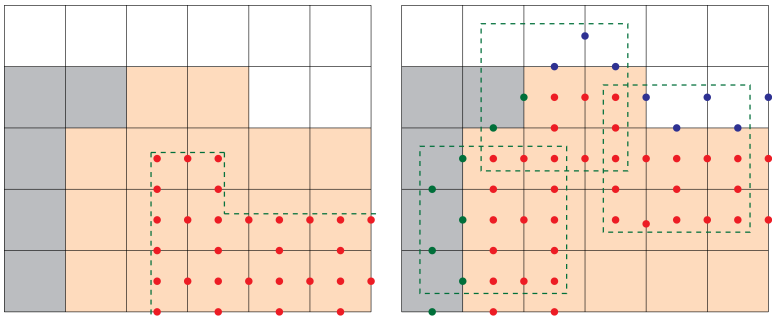


Figure 5.3: Left: Extent of distributive smoothing (interior region), Right: Boundary region with some boxes used by the box smoother.

(especially traction) can easily result in loss of symmetry. An alternative local smoother is the Kaczmarz method Trottenberg et al. (2001), which does not require symmetry or definiteness; we have nevertheless found it to converge extremely slowly, requiring a very large number of iterations. Our solution stems from a new perspective of the constitutive equations and the boundary conditions.

We will show that it is possible to construct a symmetric negative definite discretization that uses only interior variables (as defined in section 5.1). First, we revisit the constitutive equation of linear elasticity (3.1). The scalar coefficient $\mathrm{tr}(\boldsymbol{\epsilon})$ appearing in equation (3.2) is equivalently written as $\mathrm{tr}(\boldsymbol{\epsilon})=\sum_i \epsilon_{ii}=\sum_i \phi_{i,i}-d$, where $d=\mathrm{tr}(\mathbf{I})$ equals the number of spatial dimensions. Similarly, the last equation of system (4.3) is equivalent to $-(\mu/\lambda)p = \boldsymbol{\nabla}^T\boldsymbol{\phi} = \sum_i \phi_{i,i}$. Thus, we have $\mathrm{tr}(\boldsymbol{\epsilon}) = -(\mu/\lambda)p - d$, and equation (3.1) becomes

$$\mathbf{P} = \mu(\mathbf{F} + \mathbf{F}^T) - \mu p\mathbf{I} - (2\mu + d\lambda)\mathbf{I}. \tag{5.1}$$

The difference between equations (3.1) and (5.1) is that the original definition of stress is physically valid for any given deformation field $\boldsymbol{\phi}$ while the formulation of equation (5.1) will correspond to the real value of stress only when the augmented system (4.3) is solved exactly. In detail, the diagonal and off-diagonal components

of the stress tensor $\mathbf{P}$ are given as:

$$\begin{aligned} P_{ii} &= 2\mu\phi_{i,i} - \mu p - (2\mu + d\lambda) \\ P_{ij} &= \mu(\phi_{i,j} + \phi_{j,i}). \qquad (i \neq j). \end{aligned}$$

The finite difference approximations of these stress values are:

$$\begin{aligned} P_{ii}(\mathbf{X}) &= 2\mu \frac{\phi_i(\mathbf{X}+\frac{h_i}{2}\boldsymbol{e}_i) - \phi_i(\mathbf{X}-\frac{h_i}{2}\boldsymbol{e}_i)}{h_i} \\ &\quad -\mu p(\mathbf{X}) - (2\mu + d\lambda) \end{aligned} \tag{5.2}$$

$$\begin{aligned} P_{ij}(\mathbf{X}) &= \mu \left[\frac{\phi_i(\mathbf{X}+\frac{h_j}{2}\boldsymbol{e}_j) - \phi_i(\mathbf{X}-\frac{h_j}{2}\boldsymbol{e}_j)}{h_j} \right. \\ &\quad \left. + \frac{\phi_j(\mathbf{X}+\frac{h_i}{2}\boldsymbol{e}_i) - \phi_j(\mathbf{X}-\frac{h_i}{2}\boldsymbol{e}_i)}{h_i} \right]. \end{aligned} \tag{5.3}$$

The staggering of the position and pressure variables implies a natural placement of the stress values, in accordance with equations (5.2) and (5.3) (see Figure 5.5 for 2D case). Diagonal stress components are always located at cell centers, while off diagonal components $P_{ij}(i \neq j)$ are node-centered in 2D and edge-centered in 3D (see Figure 5.4, right). In a fashion similar to our classification of variables and equations, we label stresses as *interior* if their discrete stencils (defined in equations (5.2) and (5.3)) contain only interior or Dirichlet variables, while *boundary* stresses include at least one traction boundary variable in their stencil. Interior and boundary stresses are depicted in green and red, respectively, in figure 5.6 (left).

We can verify that the *position* equations $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ of system (4.3) are equivalent to the divergence form $\hat{\mathcal{L}}_i\hat{\boldsymbol{\phi}}=\partial_j P_{ij}$, where $\mathbf{P}$ is now given by the new definition

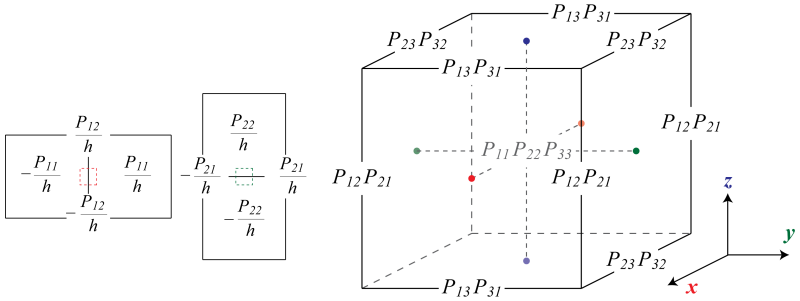


Figure 5.4: Left: Equations $\mathcal{L}_1$,$\mathcal{L}_2$ expressed as divergence stencils. Right: Placement of the components of stress tensor $\mathbf{P}$ in 3D.

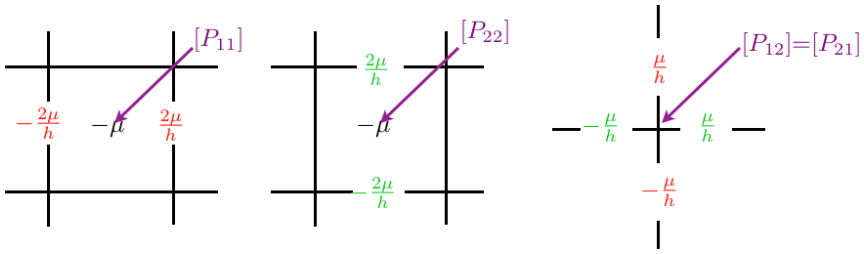


Figure 5.5: Discrete stencils for linear augmented stress components. Left: P_{11}; middle: P_{22}; right: $P_{12} = P_{21}$.

of equation (5.1). The discrete stencils for these equations can be constructed as a two-step process. First, we construct a finite difference discretization for each equation $f_i = \partial_j P_{ij}$, treating every value P_{ij} appearing in this stencil as a separate variable:

$$f_i(\mathbf{X}) = \sum_{j=1}^{d} \frac{P_{ij}(\mathbf{X}+\frac{h_j}{2}\boldsymbol{e}_j) - P_{ij}(\mathbf{X}-\frac{h_j}{2}\boldsymbol{e}_j)}{h_j}, \tag{5.4}$$

see figure 5.4 (left) for a visual illustration of this divergence stencil in 2D. As a second step, we replace the stress values P_{ij} in equation (5.4) using either a finite difference approximation, or a boundary condition. Each of the stress values P_{ij} (4 stresses in 2D, 6 in 3D) can either be an interior or a boundary stress. For all interior stresses, we simply substitute the appropriate finite difference

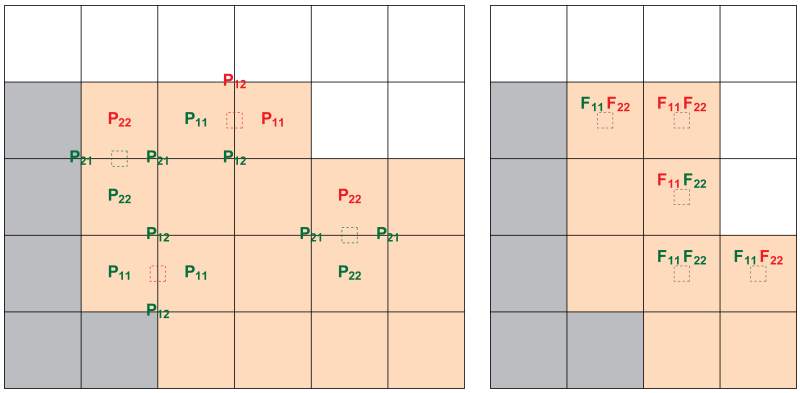


Figure 5.6: Left: Stress variables used in the divergence form of certain interior equations. Boundary stress variables are colored red, interior stresses are green. All boundary stresses can be set to a specific value using a traction condition from a nearby boundary. Right: Interior and boundary gradients used in pressure equations.

stencil, from equation (5.2) or (5.3). For boundary stresses, instead of computing them using a finite difference, we assume that their value is *known* by virtue of a traction boundary condition, thus this value can be simply substituted in equation (5.4). The assumption that every boundary stress is determined by a traction boundary condition is justified as follows:

- Stress variables of the form $\mathbf{P}_{ij} (i \neq j)$ are centered on grid edges in 3D (see Figure 5.4, right) and on grid nodes in 2D. This stress variable appears in the finite difference approximation of the term $\partial_j P_{ij}$ in the interior equation $\hat{\boldsymbol{\mathcal{L}}}_i$. Let $\boldsymbol{X}^*$ be the location where equation $\hat{\mathbf{L}}_i$ is centered. The stress variable P_{ij} is located one half of a grid cell away from $\boldsymbol{X}^*$, *along the direction* $\boldsymbol{e}_j$. Without loss of generality, assume P_{ij} is located at $\boldsymbol{X}^* + \frac{h_j}{2}\boldsymbol{e}_j$. P_{ij} neighbors exactly four cells; out of those, the two centered at $\boldsymbol{X}^* \pm \frac{h_i}{2}\boldsymbol{e}_i$ are *interior* cells, since we assumed that $\hat{\mathbf{L}}_i$ was an interior equation. The two other neighbor cells of P_{ij} are centered at $\boldsymbol{X}^* \pm \frac{h_i}{2}\boldsymbol{e}_i + h_j \boldsymbol{e}_j$. We can verify that if those two cells were interior or Dirichlet, P_{ij} would have been

an *interior* stress. Thus, P_{ij} is a boundary stress and one of the cells centered at $\boldsymbol{X}^* \pm \frac{h_i}{2}\boldsymbol{e}_i + h_j\boldsymbol{e}_j$ must be exterior. This means that P_{ij} is incident on a traction boundary face perpendicular to the direction $\boldsymbol{e}_j$, and there exists a traction condition $\mathbf{P}\boldsymbol{e}_j = \boldsymbol{t}$ that specifies a value $P_{ij} = t_i$ for this component of the stress. For a free boundary we simply have $P_{ij} = 0$.

- Stress variables of the form $\mathbf{P}_{ii}$ are located at cell centers, and appear in the finite difference approximation of $\partial_i P_{ii}$ in the interior equation $\hat{\mathbf{L}}_i$. Similar to the previous case, P_{ii} is located one half grid away from the location $\boldsymbol{X}^*$ of $\hat{\mathbf{L}}_i$ *along the direction* $\boldsymbol{e}_i$. Without loss of generality, assume P_{ii} is located at $\boldsymbol{X}^*+\frac{h_i}{2}\boldsymbol{e}_i$. From (5.2) we see that the stencil for P_{ii} contains the variables $\phi_i(\boldsymbol{X}^*), p(\boldsymbol{X}^*+\frac{h_i}{2}\boldsymbol{e}_i)$ which are both *interior* (since $\hat{\mathbf{L}}_i$ is interior) and the one additional variable $\phi_i(\boldsymbol{X}^*+h_i\boldsymbol{e}_i)$. P_{ii} would be a boundary stress only if $\phi_i(\boldsymbol{X}^*+h_i\boldsymbol{e}_i)$ was an exterior variable; in this case P_{ii} would have been "near" (specifically half a cell away from) a traction boundary face normal to $\boldsymbol{e}_i$. Once again, we will use the traction condition associated with this boundary to set $P_{ii} = t_i$ (or $P_{ii} = 0$ for a free boundary). The subtlety of this formulation is that the stress variable P_{ii} is not located exactly on the boundary; nevertheless the discrete stencil for P_{ii} is still a valid *first-order* approximation of the P_{ii} *at* the boundary.

In summary, we have justified that all *boundary* stress variables can be eliminated (and replaced with known constants) from the divergence form of interior position equations. Notably, for equations that are far enough from the traction boundary (specifically, those that do not require any boundary stresses in equation (5.1)), this process yields exactly the same results as the direct discretization of system (4.3). A similar treatment is performed on the discretization of the pressure equation $\hat{\mathcal{L}}_p = \mu \sum_i F_{ii} + \frac{\mu^2}{\lambda} p$. Similar to stresses, the deformation gradients

F_{ii} are also characterized as interior or boundary, based on whether they touch traction boundary variables. Since $P_{ii} = 2\mu F_{ii} - \mu p - (2\lambda + d\mu)$, we observe that F_{ii} is boundary if and only if the stress P_{ii} is boundary (see figure 5.6, right). For such boundary gradients or stresses we can use the traction condition $P_{ii} = t_i$ to eliminate F_{ii} from the pressure equation. This is accomplished by replacing $\hat{\mathbf{L}}_p \leftarrow \hat{\mathbf{L}}_p - \frac{1}{2}(P_{ii} - t_i)$ for every boundary gradient F_{ii}.

Our manipulations effectively remove all traction boundary variables from the discretization of the interior equations. For every Dirichlet boundary variable, we assume a Dirichlet condition of the form $\phi_i = c_i$ is provided. Thus, we can substitute a given value for every Dirichlet variable in the stencil of every interior equation that uses it. As a result, our overall discrete system can be written as $\hat{\mathbf{L}}^* \hat{\boldsymbol{\phi}}^* = \hat{\boldsymbol{f}} - \hat{\boldsymbol{f}}^D = \hat{\boldsymbol{f}}^*$, where $\hat{\boldsymbol{\phi}}^*$ only contains interior variables, and $\hat{\boldsymbol{f}}^D$ results from moving the Dirichlet conditions to the right-hand side. The discrete system matrix $\hat{\mathbf{L}}^*$ has as many rows and columns as interior variables, and will differ from $\hat{\mathbf{L}}$ near the boundaries, as it incorporates the effect of the boundary conditions. An analysis of our formulation can verify that $\hat{\mathbf{L}}^*$ has the form

$$\hat{\mathbf{L}}^* = \begin{pmatrix} \mathbf{L}_\phi & \mathbf{G} \\ \mathbf{G}^T & \mathbf{D}_p \end{pmatrix}. \tag{5.5}$$

In this formulation $\mathbf{L}_\phi$ is symmetric, negative definite, and $\mathbf{D}_p$ is a diagonal matrix with positive diagonal elements. As a final step, we define the *substitution* matrix $\hat{\mathbf{U}}$

$$\hat{\mathbf{U}} = \begin{pmatrix} \mathbf{I} & -\mathbf{G}\mathbf{D}_p^{-1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix},$$

and use it to pre-multiply our equation as

$$\hat{\mathbf{U}}\hat{\mathbf{L}}^{*}\hat{\boldsymbol{\phi}}^{*} = \begin{pmatrix} \mathbf{L}_{\phi} - \mathbf{G}\mathbf{D}_{p}^{-1}\mathbf{G}^{T} & \mathbf{0} \\ \mathbf{G}^{T} & \mathbf{D}_{p} \end{pmatrix} \boldsymbol{u}^{*} = \hat{\mathbf{U}}\hat{\boldsymbol{f}}^{*}. \tag{5.6}$$

The derived equation is lower triangular, therefore, we can first relax the equation $\mathbf{L}^{\text{unaug}}\boldsymbol{u} = \boldsymbol{f}^{\text{unaug}}$, where $\mathbf{L}^{\text{unaug}} \triangleq \mathbf{L}_{\phi} - \mathbf{G}\mathbf{D}_{p}^{-1}\mathbf{G}^{T}$, and then substitute in to the second equation in (5.6) to resolve p. $\mathbf{L}^{\text{unaug}}$ is essentially a symmetric and negative definite discretization of our non-augmented system (3.3) and can be smoothed via Gauss-Seidel iteration. The boundary and interior regions are smoothed in separate sweeps; during the sweep of the boundary smoother, all interior variables not being smoothed are effectively treated as Dirichlet values. The boundary smoother is confined in a narrow region between boundary conditions (variables of this narrow band are depicted in red in Figure 5.3, right). This narrow support of the boundary smoother has a strong stabilizing effect, and compensates for any difficulties encountered with near-incompressible materials. In practice, we found that 2 Gauss-Seidel boundary sweeps for every sweep of the distributive interior smoother are sufficient for Poisson's ratio up to $\nu = .45$, while 3-4 Gauss-Seidel sweeps suffice for values as high as $\nu = .495$. Finally, we note that Gauss-Seidel is not the only option for smoothing the discrete system derived in this section; in fact it is even possible to use a distributive smoother as in Algorithm 2, taking care to restrict the distribution stencil to active variables.

After completing the smoothing process, we need to update the values of the pressure and traction boundary variables that were previously annihilated. Since the lower right block of equation (5.6) is diagonal, all pressure equations can be satisfied exactly via a simple Gauss-Seidel sweep. Similarly, the boundary traction variables can be updated using the traction conditions $P_{ij} = t_i$ in a

simple back-substitution step, first updating variables on faces between interior and exterior cells, and then variables located at half-cell distance away from the interior region. Notably, at the end of the process all boundary conditions are satisfied *exactly* (i.e. they will have zero residuals), which simplifies our inter-grid transfers discussed next.

5.4 Restriction and prolongation on staggered grid

We designed the Restriction ($\mathcal{R}$) and Prolongation ($\mathcal{P}$) operators employed by algorithm 1 aiming to keep implementation as inexpensive as possible, while conforming to the textbook accuracy requirements for full multigrid efficiency (see Trottenberg et al. (2001)). We define the following 1D averaging operators

$$\mathcal{B}^1 u[x] = \tfrac{1}{2}u[x-\tfrac{h}{2}] + \tfrac{1}{2}u[x+\tfrac{h}{2}]$$
$$\mathcal{B}^2 u[x] = \tfrac{1}{4}u[x-h] + \tfrac{1}{2}u[x] + \tfrac{1}{4}u[x+h]$$
$$\mathcal{B}^3 u[x] = \tfrac{1}{8}u[x-\tfrac{3h}{2}] + \tfrac{3}{8}u[x+\tfrac{h}{2}] + \tfrac{3}{8}u[x-\tfrac{h}{2}] + \tfrac{1}{8}u[x+\tfrac{3h}{2}].$$

The restriction and prolongation operators will be defined as tensor product stencils of the preceding 1D operators as

$$\mathcal{R}_1 = \mathcal{B}^2 \otimes \mathcal{B}^1 \otimes \mathcal{B}^1 \qquad \mathcal{P}_1^T = 8\ \mathcal{B}^2 \otimes \mathcal{B}^3 \otimes \mathcal{B}^3 \tag{5.7}$$
$$\mathcal{R}_2 = \mathcal{B}^1 \otimes \mathcal{B}^2 \otimes \mathcal{B}^1 \qquad \mathcal{P}_2^T = 8\ \mathcal{B}^3 \otimes \mathcal{B}^2 \otimes \mathcal{B}^3 \tag{5.8}$$
$$\mathcal{R}_3 = \mathcal{B}^1 \otimes \mathcal{B}^1 \otimes \mathcal{B}^2 \qquad \mathcal{P}_3^T = 8\ \mathcal{B}^3 \otimes \mathcal{B}^3 \otimes \mathcal{B}^2 \tag{5.9}$$
$$\mathcal{R}_p = \mathcal{B}^1 \otimes \mathcal{B}^1 \otimes \mathcal{B}^1 \qquad \mathcal{P}_p^T = 8\ \mathcal{B}^1 \otimes \mathcal{B}^1 \otimes \mathcal{B}^1, \tag{5.10}$$

where $\mathcal{R}_i$,$\mathcal{P}_i$ are the restriction and prolongation operators used for variable $\boldsymbol{u}_i$, respectively, and $\mathcal{R}_p$,$\mathcal{P}_p$ are the operators used for the pressure variables. Our

restriction and prolongation are not the transpose of one another (as commonly done in other methods) but this practice is not unusual or problematic, see e.g. Brandt (1977a). Our domain description for the finest grid was based on a partitioning of the cells into interior, exterior and Dirichlet. The coarse grid is derived from the natural 8-to-1 coarsening of the Cartesian background lattice. Furthermore, a coarse cell is designated a Dirichlet cell if *any* of its eight fine sub-cells is Dirichlet. If any of the fine sub-cells are interior and none is Dirichlet, the coarse cell will be considered interior. Otherwise, the coarse cell is exterior. Thus, the coarse active domain is geometrically a superset of the fine, while its Dirichlet parts are extended (see Figure 5.7). Despite this geometrical discrepancy, which is no larger than the grid cell size, we were still able to obtain a highly efficient multigrid scheme as described next.

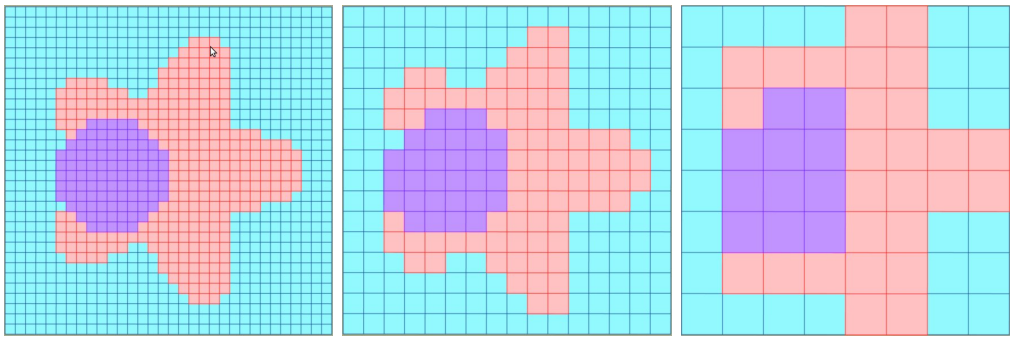


Figure 5.7: Grid coarsening.

In general, if a smoother leaves a residual on the boundary conditions, this residual has to be restricted to a coarse grid boundary equation. In our treatment of boundary conditions, we effectively forced all boundary conditions to be satisfied *exactly* after every application of the smoother, thus all coarse boundary conditions will be *homogeneous*; for Dirichlet equations they will have the form $u_i^{2h} = 0$ (i.e. the coarse grid incurs no correction), while traction equations will be of the form $\hat{P}_{ij}^{2h} = 0$, where $\hat{\mathbf{P}} = \mu(\mathbf{F} + \mathbf{F}^T) - \mu p \mathbf{I}$ is the homogeneous

part of **P**. We also note that, due to the possible geometrical change of the Dirichlet region, certain coarse Dirichlet equations will be centered on locations that were interior in the fine grid (shown as red dots in Figure 5.8, right). The fine grid interior equations (red dots in Figure 5.8, left) that would restrict their residuals onto these (now Dirichlet) coarse locations, will not have their residuals well represented on the coarse grid. We compensate for this inaccuracy by performing an extra 2-3 sweeps of our boundary Gauss-Seidel smoother over these equations, driving their residuals very close to zero, just prior to restriction. A similar inaccuracy may affect the prolongation of the correction: as we previously mentioned, the active region may have extended more in the coarse grid, compared to the fine. This discrepancy may introduce inaccuracies in the coarse grid solution near such relocated boundaries. Again, we compensate by performing an additional 2-3 Gauss-Seidel smoother sweeps on the locations of the fine grid that prolongate corrections from such relocated boundary variables (depicted as green circles in Figure 5.8, left). This simple treatment proved quite effective to guarantee a good coarse correction despite the small geometrical discrepancies of the two domains.

We apply this method on the same cubic domain example in section 3.3, and achieved a consistent convergence rate of smaller than 0.28 for Poisson's ratio from 0.2 to 0.49 (see Figure 5.9).

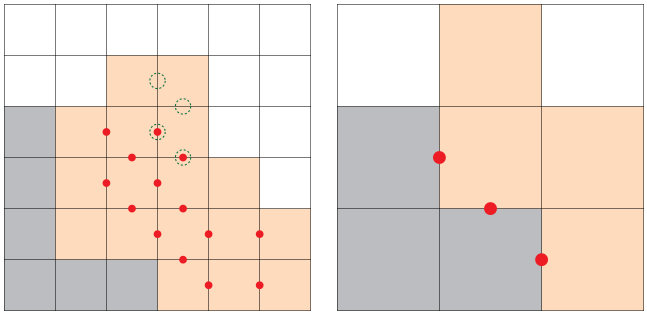


Figure 5.8: Boundary discrepancies in the fine (left) and coarse (right) domains. On the right, red dots indicate locations containing Dirichlet conditions on the coarse grid, but interior equations on the fine grid. On the left, red dots indicate interior equations that would restrict residuals on one or more locations occupied by Dirichlet conditions on the coarse grid; those restricted residuals will be replaced with zero. Green circles indicate fine interior variables that prolongate their correction from boundary coarse variables.

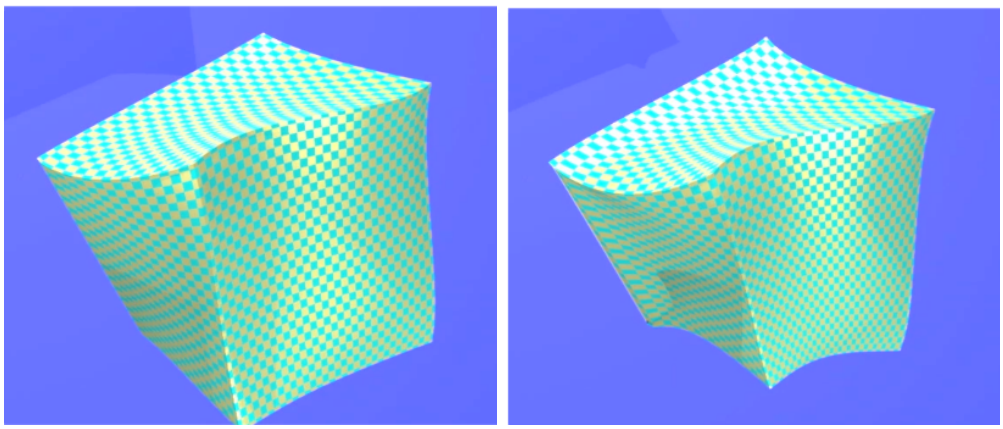


Figure 5.9: Comparison of multigrid convergence with different Poisson's ratios using distributive relaxation. We apply the same boundary condition and start with the same random initial guess and plot the deformed object after 5 multigrid V-(1,1) cycles. Left: Poisson's ratio is 0.2; right: Poisson's ratio is 0.49.

CHAPTER 6

Extended models and results

The linear elasticity model is appropriate for small deformation problems, however, it is not rotation invariant. Under large deformation, obvious artifacts are observed. In this chapter, a co-rotational elasticity model which is frequently used in graphics applications is considered. This is a non-linear model. We linearize the problem, and define a similar augmentation and distributive relaxation. An efficient multigrid solver is developed for the linearized problem, and extended to dynamic problem as well.

6.1 Co-rotational linear elasticity

In the large deformation regime, and in the presence of large rotational deformations, the linear elasticity model develops artifacts such as volumetric distortions in parts of the domain with large rotations. We provide an extension to the co-rotational linear elasticity model, which has been used in slightly different forms by a number of authors in computer graphics Müller et al. (2002); Hauth and Strasser (2004); Müller and Gross (2004), and has also used with finite elements and multigrid by Georgii and Westermann (2006, 2008). The co-rotational formulation extracts the rotational component of the local deformation at a specific part of the domain by computing the polar decomposition of the deformation gradient tensor $\mathbf{F} = \mathbf{RS}$ into the rotation $\mathbf{R}$ and the symmetric tensor $\mathbf{S}$. The

stress is then computed as $\mathbf{P} = \mathbf{R}\mathbf{P}_L(\mathbf{S})$, where $\mathbf{P}_L$ denotes the stress of a *linear* material, as described in equation (3.1). Thus, the co-rotational formulation computes stresses by applying the constitutive equation of linear elasticity in a frame of reference that is rotated with the material deformation as follows:

$$\begin{aligned}
\mathbf{P} &= \mathbf{R}\mathbf{P}_L(\mathbf{S}) = \mathbf{R}\left[2\mu(\mathbf{R}^T\mathbf{F}-\mathbf{I}) + \lambda\mathrm{tr}(\mathbf{R}^T\mathbf{F}-\mathbf{I})\mathbf{I}\right] \\
&= 2\mu(\mathbf{F}-\mathbf{R}) + \lambda\mathrm{tr}(\mathbf{R}^T\mathbf{F}-\mathbf{I})\mathbf{R} \\
&= 2\mu\mathbf{F} + \lambda\mathrm{tr}(\mathbf{R}^T\mathbf{F})\mathbf{R} - (2\mu + d\lambda)\mathbf{R} \\
&= 2\mu\mathbf{F} - \mu p\mathbf{R} - (2\mu + d\lambda)\mathbf{R}, \qquad (6.1)
\end{aligned}$$

where the last form of the stress in equation (6.1) results from introducing an auxiliary pressure variable $p{=}-(\lambda/\mu)\mathrm{tr}(\mathbf{R}^T\mathbf{F})$ similar to the augmentation used for linear elasticity in section 4.1. As before, the augmented position equations are defined as $\partial_j P_{ij}{=}f_i$. Combining with the pressure equations and rearranging we get

$$\begin{pmatrix} 2\mu\Delta\mathbf{I} & -\mu(\boldsymbol{\nabla}^T\mathbf{R}^T)^T \\ \mu(\mathbf{R}\boldsymbol{\nabla})^T & \frac{\mu^2}{\lambda} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi} \\ p \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}-(2\mu+d\lambda)\boldsymbol{\nabla}\cdot\mathbf{R} \\ 0 \end{pmatrix}. \qquad (6.2)$$

The notation for the off-diagonal blocks of the matrix in equation (6.2) was used to indicate whether the operators $\boldsymbol{\nabla}, \boldsymbol{\nabla}^T$ operate or not on the rotation matrix $\mathbf{R}$. In index form, these operators equal $[\mu(\boldsymbol{\nabla}^T\mathbf{R}^T)^T]_i = \mu\partial_j R_{ij}$, and $[\mu(\mathbf{R}\boldsymbol{\nabla})^T]_i = \mu R_{ij}\partial_j$ respectively.

6.1.1 Nonlinear iteration

In contrast with the equations of linear elasticity, equation (6.2) is a nonlinear PDE, since both the operator matrix and the right hand side vector contain the rotation matrix $\mathbf{R}$ which depends on the current deformation $\boldsymbol{\phi}$ itself. We

highlight this fact by writing this system as $\hat{\mathcal{L}}[\hat{\boldsymbol{\phi}}]\hat{\boldsymbol{\phi}}=\hat{\boldsymbol{f}}[\hat{\boldsymbol{\phi}}]$. Nevertheless, for the purposes of a multigrid scheme it is possible to treat system (6.2) as a linear equation, by freezing the values of $\hat{\mathcal{L}}$ and $\hat{\boldsymbol{f}}$ for the duration of a V-cycle, and updating them after a better solution to this frozen coefficient system has been obtained. In an iterative fashion, we obtain the $(k{+}1)$-th approximation to the solution of the linear system by executing one V-cycle on the constant coefficient system $\hat{\mathcal{L}}[\hat{\boldsymbol{\phi}}^k]\hat{\boldsymbol{\phi}}^{k+1}=\hat{\boldsymbol{f}}[\hat{\boldsymbol{\phi}}^k]$ (or quasi-linear form) of equation (6.2) in this context.

6.1.2 Distributive relaxation

We generalize the distributive smoothing approach to the quasi-linear equation (6.2). In this case, the distribution matrix is

$$\hat{\mathcal{M}} = \begin{pmatrix} \mathbf{I} & -(\boldsymbol{\nabla}^T\mathbf{R}^T)^T \\ \mathbf{0} & -2\Delta \end{pmatrix}. \tag{6.3}$$

Then, the distributed operator $\hat{\mathcal{L}}\hat{\mathcal{M}}$ becomes

$$\hat{\mathcal{L}}\hat{\mathcal{M}} = \begin{pmatrix} 2\mu\Delta\mathbf{I} & 2\mu\left[(\boldsymbol{\nabla}^T\mathbf{R}^T)^T\Delta - \Delta(\boldsymbol{\nabla}^T\mathbf{R}^T)^T\right] \\ \mu(\mathbf{R}\boldsymbol{\nabla})^T & -\mu(1+\frac{2\mu}{\lambda})\Delta \end{pmatrix}. \tag{6.4}$$

The top right block of $\hat{\mathcal{L}}\hat{\mathcal{M}}$ would be equal to zero if $\mathbf{R}$ is a spatially constant rotation, but not in the general case. However, near a solution where the rotations are expected to be smooth, this value is effectively zero, and $\hat{\mathcal{L}}\hat{\mathcal{M}}$ becomes a triangular matrix, similar to the linear case. Effectively, even if the distributed system is near-triangular, a Gauss-Seidel algorithm will still be an acceptable smoother. In practice we found distributive Gauss-Seidel to be a good smoother for the quasi-linear problem at all times, although the convergence rates were

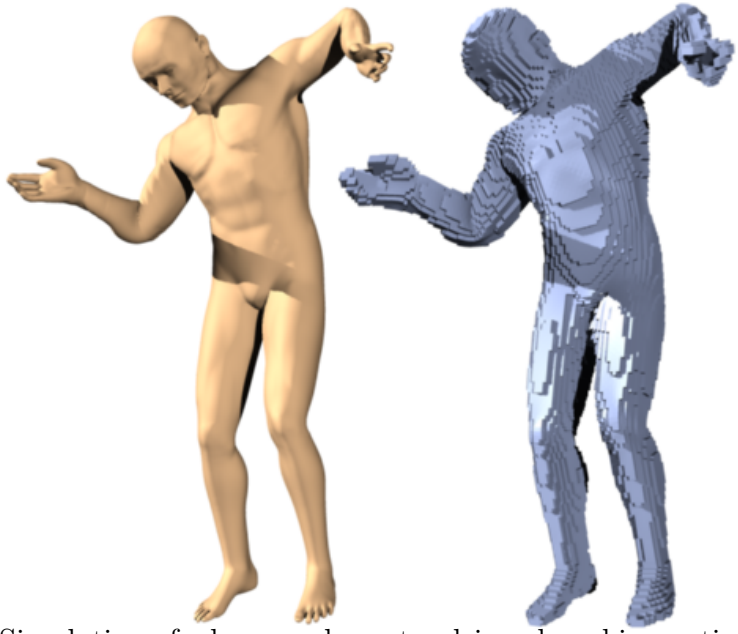


Figure 6.1: Simulation of a human character driven by a kinematic skeleton. The high-resolution rendering surface is seen in the left, while the simulation lattice is depicted on the right (resolution: 142K nodes, grid spacing 9mm).

slightly lower away from the solution.

6.1.3 Interior discretization

Discretization complications are introduced due to the fact that pressures are now multiplied by the non-diagonal matrix $\mathbf{R}$ in the augmented stress definition (6.1).

First of all, the deformation gradient matrix $\mathbf{F}$ is estimated at each cell center. Notice that the diagonal components of $\mathbf{F}$ are naturally defined with a centered difference scheme, while the off diagonal components are naturally defined on grid cell corners in 2D and edge centers in 3D. Thus, we take an average of the 4 off diagonal components to approximate $\mathbf{F}$ at cell centers. A polar decomposition is thus applied on this matrix to generate the frozen coefficients $\mathbf{R}$ at each cell

center.

Thus, the diagonal stresses P_{ii} can be discretized using a centered difference scheme on ϕ, and the coefficient for p uses the current approximation of $\mathbf{R}$ (see Figure 6.2 for 2D case). However, off diagonal stresses P_{ij} $(i \neq j)$ require an edge centered pressure value (or corner centered value in 2D case). We compute the needed pressure value by averaging the four neighboring pressures.

Finally, pressure equations are written as $\mu R_{ij}\phi_{i,j} + (\mu^2/\lambda)p = 0$, indicating that all gradient values $\phi_{i,j}$ are needed at a cell center. The stencil for diagonal gradients are again naturally defined at cell centers. The stencil for off-diagonal gradients will be averaged from the four neighboring edge centers, where they are naturally defined (see Figure 6.3 for 2D case).

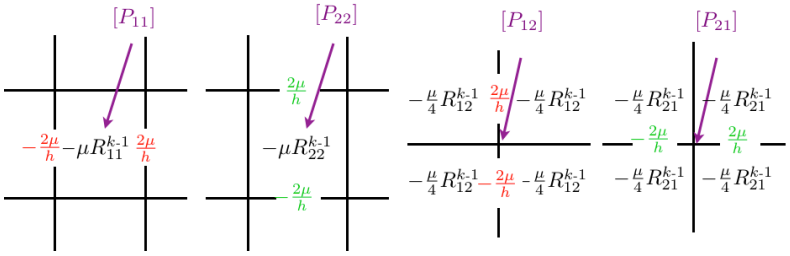


Figure 6.2: Discrete stencils for linearized augmented stress components. From left to right are the stencils for P_{11}, P_{22}, P_{12} and P_{21}.

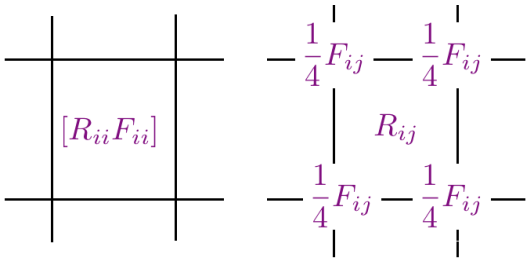


Figure 6.3: Discrete stencils for each $R_{ij}F_{ij}$ from pressure equations. Left: diagonal components $R_{ii}F_{ii}$; right: off diagonal components $R_{ij}F_{ij}(i \neq j)$.

6.1.4 Boundary discretization

In co-rotational elasticity case, we discretize our domain in exactly the same way as the linear elasticity case by rasterizing the domain into voxellized cells.

We first update $\mathbf{F}$ at all active cell centers. When the cell is near domain boundary, we compute the deformation gradient $\mathbf{F}$ by take an average among available $\mathbf{F}$ from a wider range of neighboring cells. Thus, $\mathbf{R}$ can be computed from $\mathbf{F}$ at each active cell center.

For the purposes of boundary smoothing, we again derive a symmetric definite discretization where Gauss-Seidel can be used. Following section 9.1, in the discrete form of stress divergence equation $\hat{\mathcal{L}}_i\hat{\boldsymbol{\phi}}=\partial_j P_{ij}$ introduced in (5.4), any exterior stress components are eliminated from the divergence stencil using an appropriate traction equation. If the stress component to be eliminated is a diagonal component, then the same boundary variable is eliminated as in linear elasticity case. If the stress component to be eliminated is an off diagonal component P_{ij}, that implies at least one among the 4 neighboring cells to the edge where P_{ij} locates is empty. Fortunately, in this case, P_{ji} is either also going to be eliminated, when the edge is an actual edge of the active domain, or not involved in any interior equations, when the edge is not an actual edge of the active domain, but rather the domain boundary is flat when passing the edge. Therefore, the coefficient R_{ij} is never going to be required in an empty cell. Also, $\mathbf{R}$ in Dirichlet region can be precomputed according to the fixed deformation defined in Dirichlet cells.

Near the boundary, the pressure equations may also require external variables. We designed a discretization that leads to a symmetric discretization of $\hat{\mathcal{L}}$. When computing $R_{ij}F_{ij}$, if this stencil locates at an edge center, where the collocated P_{ij} is to be eliminated, then we will freeze the value $R_{ij}F_{ij}$ to the cur-

rent approximation that is already computed when updating $\mathbf{R}$. Thus, this value will be moved to the right hand side and will not involve in the linear system. We verified that the derived system is symmetric with the form of (5.5). Thus, the same technique of unaugmentation is applied. Finally, we relax (5.6) in the boundary band.

6.1.5 Distribution discretization

By comparing the distribution matrix (6.3) and the augmented system (6.2), we discretize the distribution matrix by

$$\hat{\mathbf{M}} = \begin{pmatrix} \mathbf{I} & \frac{1}{\mu}\mathbf{G} \\ \mathbf{0} & -2\mathbf{M}_p \end{pmatrix}, \tag{6.5}$$

where $\mathbf{M}_p$ is the 5 point stencil of Laplacian operator. As we discussed, this discretization derives to an *approximate* lower triangular matrix $\hat{\mathbf{L}}\hat{\mathbf{M}}$. And the efficiency of the derived distributive relaxation is verified by numerical tests.

6.1.6 Coarsening

The coarse grid correction satisfy equation $\hat{\mathcal{L}}\delta\hat{\boldsymbol{\phi}} = \hat{\boldsymbol{f}} - \hat{\mathcal{L}}\hat{\boldsymbol{\phi}}^{\text{current}}$. This equation may be discretized in the same way on a coarser grid. However, we need the estimated coefficients $\mathbf{R}$ defined on coarse grid cell centers. We simple average the rotation vectors from the active cells among the 4 subcells by doing spherical linear interpolation between each pair of rotations. We use the same restriction and prolongation as in linear case, and achieve similar convergence rates, which will be discussed in section 6.3.

6.2 Dynamic system

The static formulation of elasticity disregards any dynamic effects. Our method, however, can easily accommodate the simulation of dynamic deformation; the effect of inertia actually improves the conditioning of the discrete equations.

The dynamic system is represented with the following time-dependent equation

$$\frac{\partial}{\partial t}\begin{pmatrix} \boldsymbol{\phi} \\ \boldsymbol{v} \end{pmatrix} = \begin{pmatrix} \boldsymbol{v} \\ \frac{1}{\rho}(\boldsymbol{f}_{\text{internal}}(\boldsymbol{\phi}, \boldsymbol{v}) + \boldsymbol{f}_{\text{external}}) \end{pmatrix},$$

where $\boldsymbol{f}_{\text{internal}}(\boldsymbol{\phi}, \boldsymbol{v})$ is the internal forces density. Here, we consider a damped elasticity force. And $\boldsymbol{f}_{\text{external}}$ the external force density like gravity.

6.2.1 Time integral

Implicit time integral presents high stability. We use a Backward Euler method, which allows simulation with large time steps.

$$\frac{1}{\delta t}\begin{pmatrix} \boldsymbol{\phi}^{k+1} - \boldsymbol{\phi}^{k} \\ \boldsymbol{v}^{k+1} - \boldsymbol{v}^{k} \end{pmatrix} = \begin{pmatrix} \boldsymbol{v}^{k+1} \\ \frac{1}{\rho}(\boldsymbol{f}_{\text{internal}}(\boldsymbol{\phi}^{k+1}, \boldsymbol{v}^{k+1}) + \boldsymbol{f}^{k+1}_{\text{external}}) \end{pmatrix},$$

i.e.

$$\begin{cases} \boldsymbol{\phi}^{k+1} &= \boldsymbol{\phi}^{k} + \delta t \boldsymbol{v}^{k+1} \\ \boldsymbol{v}^{k+1} &= \boldsymbol{v}^{k} + \frac{\delta t}{\rho}(\boldsymbol{f}_{\text{internal}}(\boldsymbol{\phi}^{k+1}, \boldsymbol{v}^{k+1}) + \boldsymbol{f}_{\text{external}}) \end{cases}.$$

The internal force is composed of the elastic force and the damping force. Previously, we developed augmented discretizations for the linear elasticity and co-rotational elasticity forces. $\boldsymbol{f}^{h}_{elasticity}(\boldsymbol{u}^{h}) = \mathbf{L}^{h}\boldsymbol{u}^{h} - \boldsymbol{f}^{h}$. In linear case, $\mathbf{L}^{h}$ is a

constant linear operator, and in co-rotational case, at each nonlinear iteration, $\mathbf{L}^h$ is also a frozen coefficient linear operator. Since it is hard to control dynamic effects with purely numerical damping, we also considered explicit Rayleigh damping. In Rayleigh damping, the damping force depends linearly on the velocity and

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{v}} = \gamma \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\phi}},$$

i.e. $\boldsymbol{f}_{damping}(v) = \gamma \mathbf{L}\boldsymbol{v}$.

By applying the backward Euler method, we have,

$$\begin{cases} \boldsymbol{\phi}^{k+1} &= \boldsymbol{\phi}^k + \delta t \boldsymbol{v}^{k+1} \\ \boldsymbol{v}^{k+1} &= \boldsymbol{v}^k + \frac{\delta t}{\rho}(\mathbf{L}\boldsymbol{\phi}^{k+1} + \gamma \mathbf{L}\boldsymbol{v}^{k+1} - \boldsymbol{f}) \end{cases}.$$

By multiplying $(\mathbf{I} - \frac{\delta t \gamma}{\rho}\mathbf{L})$ on both sides of position equation, and rearranging v^{k+1} related terms, we have

$$\Rightarrow \begin{cases} (\mathbf{I} - \frac{\delta t \gamma}{\rho}\mathbf{L})\boldsymbol{\phi}^{k+1} &= (\mathbf{I} - \frac{\delta t \gamma}{\rho}\mathbf{L})\phi^k + (\mathbf{I} - \frac{\delta t \gamma}{\rho}\mathbf{L})\delta t \boldsymbol{v}^{k+1} \\ (\mathbf{I} - \frac{\delta t \gamma}{\rho}\mathbf{L})\boldsymbol{v}^{k+1} &= \boldsymbol{v}^k + \frac{\delta t}{\rho}(\mathbf{L}\boldsymbol{\phi}^{k+1} - \boldsymbol{f}) \end{cases}.$$

By eliminating $\boldsymbol{v}^{k+1}$ from the position equation, we have

$$\begin{aligned} (\mathbf{I} - \frac{\delta t \gamma}{\rho}\mathbf{L})\boldsymbol{\phi}^{k+1} &= (\mathbf{I} - \frac{\delta t \gamma}{\rho}\mathbf{L})\boldsymbol{\phi}^k + \delta t \boldsymbol{v}^k + \frac{\delta t^2}{\rho}(\mathbf{L}\boldsymbol{\phi}^{k+1} - \boldsymbol{f}) \\ \Rightarrow (\mathbf{I} - \frac{\delta t}{\rho}(\gamma + \delta t)\mathbf{L})\boldsymbol{\phi}^{k+1} &= (\mathbf{I} - \frac{\delta t \gamma}{\rho}\mathbf{L})\boldsymbol{\phi}^k + \delta t \boldsymbol{v}^k - \frac{\delta t^2}{\rho}\boldsymbol{f} \\ \Rightarrow (\mathbf{L} - \frac{\rho}{\delta t(\gamma + \delta t)}\mathbf{I})\boldsymbol{\phi}^{k+1} &= -\frac{\rho}{\delta t(\gamma + \delta t)}(\boldsymbol{\phi}^k + \delta t \boldsymbol{v}^k - \frac{\delta t \gamma}{\rho}\mathbf{L}\boldsymbol{\phi}^k - \frac{\delta t^2}{\rho}\boldsymbol{f}). \end{aligned}$$

In the special case of zero damping, the equation simplifies to

$$(\mathbf{L} - \frac{\rho}{\delta t^2})\boldsymbol{\phi}^{k+1} = -\frac{\rho}{\delta t^2}(\boldsymbol{\phi}^k + \delta t \boldsymbol{v}^k) + \boldsymbol{f}.$$

In either case, the equation for ϕ^{k+1} is $(\mathbf{L} - s\mathbf{I})\boldsymbol{\phi}^{k+1} = \boldsymbol{f}(\boldsymbol{\phi}^k)$, where $s = \frac{\rho}{\gamma\delta t + \delta t^2}$ is a constant scalar. Therefore, the system matrix will be modified by adding a scalar matrix. A modified augmentation is proposed to resolve both dynamic problem and quasi-static probem.

6.2.2 Augmentation and distribution

In previous quasi-static linear elasticity and co-rotational elasticity cases, the original problem $\mathcal{L}\boldsymbol{u} = \boldsymbol{f}$ is augmented to the following problem

$$\begin{pmatrix} \mathbf{L}_\phi & \mathbf{G} \\ \mathbf{G}^T & \mathbf{D}_p \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi} \\ \boldsymbol{p} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \mathbf{0} \end{pmatrix},$$

and a change of variable is introduced

$$\begin{pmatrix} \boldsymbol{\phi} \\ \boldsymbol{p} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \frac{1}{\mu}\mathbf{G} \\ \mathbf{0} & \mathbf{M}_p \end{pmatrix} \begin{pmatrix} \boldsymbol{\psi} \\ \boldsymbol{q} \end{pmatrix}$$

to obtain an exact or approximated triangular system:

$$\begin{pmatrix} \mathbf{L}_\phi & \mathbf{G} \\ \mathbf{G}^T & \mathbf{D}_p \end{pmatrix} \begin{pmatrix} \mathbf{I} & \frac{1}{\mu}\mathbf{G} \\ \mathbf{0} & \mathbf{M}_p \end{pmatrix} \begin{pmatrix} \boldsymbol{\psi} \\ \boldsymbol{q} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_\phi & \frac{1}{\mu}\mathbf{L}_\phi\mathbf{G} + \mathbf{G}\mathbf{M}_p \\ \mathbf{G}^T & \frac{1}{\mu}\mathbf{G}^T\mathbf{G} + \mathbf{D}_p\mathbf{M}_p \end{pmatrix} \begin{pmatrix} \boldsymbol{\psi} \\ \boldsymbol{q} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \mathbf{0} \end{pmatrix}.$$

In linear elasticity case, the top right block $\frac{1}{\mu}\mathbf{L}_\phi\mathbf{G} + \mathbf{G}\mathbf{M}_p = 0$, and in co-rotational elasticity case, $\frac{1}{\mu}\mathbf{L}_\phi\mathbf{G} + \mathbf{G}\mathbf{M}_p \approx 0$.

For the dynamic system, if we introduce the same pressure, then the aug-

mented system

$$\begin{pmatrix} \mathbf{L}_\phi - s\mathbf{I} & \mathbf{G} \\ \mathbf{G}^T & \mathbf{D}_p \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi} \\ \boldsymbol{p} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \mathbf{0} \end{pmatrix}$$

is again symmetric indefinite and can be triangularized with a modified change of variable

$$\begin{pmatrix} \boldsymbol{\phi} \\ \boldsymbol{p} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \frac{1}{\mu}\mathbf{G} \\ \mathbf{0} & \mathbf{M}_p + \frac{s}{\mu}\mathbf{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{\psi} \\ \boldsymbol{q} \end{pmatrix}.$$

In fact, the derived system

$$\begin{aligned} & \begin{pmatrix} \mathbf{L}_\phi - s\mathbf{I} & \mathbf{G} \\ \mathbf{G}^T & \mathbf{D}_p \end{pmatrix} \begin{pmatrix} \mathbf{I} & \frac{1}{\mu}\mathbf{G} \\ 0 & \mathbf{M}_p + \frac{s}{\mu}\mathbf{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{\psi} \\ \boldsymbol{q} \end{pmatrix} \\ = & \begin{pmatrix} \mathbf{L}_\phi - s\mathbf{I} & \frac{1}{\mu}\mathbf{L}_\phi\mathbf{G} + \mathbf{G}\mathbf{M}_p \\ \mathbf{G}^T & \frac{1}{\mu}\mathbf{G}^T\mathbf{G} + \mathbf{D}_p\mathbf{M}_p + \frac{s}{\mu}\mathbf{D}_p \end{pmatrix} \begin{pmatrix} \boldsymbol{\psi} \\ \boldsymbol{q} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \mathbf{0} \end{pmatrix}. \end{aligned}$$

has the same upper triangular block which is exactly or approximately zero. Notice that the derived system for q is an approximation to $-\mu(1 + \frac{2\mu}{)}\lambda\Delta + \kappa I$ where κ is a constant scalar and $\kappa > 0$. Therefore, a Gauss-Seidel relaxation is even more efficient on relaxing that system. Therefore, by modifying the distribution matrix for pressure variable with an additional constant on the diagonal element, we derive a valid distributive relaxation, which should have the same or better smoothing effect as the quasi-static case. At the limit of $\delta t \to \infty$, this formulation naturally merges to the quasi-static case.

6.2.3 Coarsening of the new system

Since the additional term is a scaled identity, and $\mathbf{RP} \approx \mathbf{I}$, the same scaling identity is a good approximation for its corresponding coarse term. Also, scalar matrix is a localized equation; it can be solved by substitution. So the smaller the time step is, the larger D is, hence the faster the convergence we could expect.

6.3 Evaluation of solver performance

We first compare the performance of our method with a Conjugate Gradients (CG) solver, as illustrated in Figure 6.5. The left figure plots the reduction of the residual for our synthetic test model: a rectangular elastic box under mixed boundary conditions (also depicted in figure 6.14, on the right). We use CG to solve the symmetric, definite system resulting from the discretization of the (non-augmented) PDE (3.3) using finite differences on staggered grids of sizes 32^3 and 64^3, for two different values of Poisson's ratio ν. We observe that, after some initial progress, the convergence of CG slows down significantly. This deterioration is more pronounced on cases with more degrees of freedom, and higher incompressibility (which are the focus points of our method). Replacing the finite difference discretization with trilinear, hexahedral finite elements (middle plot) still exhibits the same stagnation, particularly for the more incompressible case. Our method (right plot), exhibits a practically constant convergence rate all the way until the error is at the levels of the floating point round-off threshold.

We subsequently compare our method with other multigrid techniques (i.e. using different relaxation or discretization approaches), in Figure 6.4. As a general comment, all methods evaluated here were able to achieve convergence rates that are largely independent of the model resolution (in contrast with CG). As a

Discretization used for Multi-Grid solver and benchmark description	Poisson's ratio	Convergence rate (V-cycle)
3D Poisson equation (scalar), 64^3 periodic domain, lexicographical Gauss-Seidel smoothing (reference problem)	N/A	0.19
Non-augmented linear elasticity, staggered finite difference discretization, 32^3 elastic box, mixed BCs	0.2	0.41
	0.475	0.85
	0.49	0.9
Linear elasticity, tetrahedral finite element discretization, 32^3 elastic box (160K tetrahedra), mixed BCs	0.2	0.35
	0.475	0.8
	0.49	0.89
Our method: Distributive Gauss-Seidel on augmented PDEs 32^3 elastic box, mixed BCs	0.2	0.26
	0.49	0.28
Our method on the 302K vertex Armadillo model (using linear elasticity)	0.475	0.21-0.35 (typical)
		0.62 (asymptotic)
Our method on the 302K vertex Armadillo model (co-rotational linear elasticity)	0.475	~0.38 (linear residual)
		~0.43 (nonlinear residual)
Our method on the 1.1M vertex articulated Human model	0.475	0.24
Our method on the 43K car model (backward euler & dynamics)	0.4	0.08

Figure 6.4: Comparison with alternative multigrid techniques. Convergence rate is defined as the asymptotic residual reduction factor: $|r_{k+1}|/|r_k|$

.

point of reference we include the convergence rate of 0.19 of a periodic 3D Poisson problem with lexicographical Gauss-Seidel smoothing. We first experimented with a staggered finite difference discretization of equation (3.3) which did not however employ the augmentation of section 4.1. We observe that the convergence rate is deteriorating with higher incompressibility, to reach a value of 0.9 for a material with $\nu = 0.49$. A similar behavior is observed with a tetrahedral finite element discretization, used in place of finite differences. These results are compatible with the findings of Griebel et al. (2003) who observe similar problems with near-incompressibility even for AMG solvers. Our method exhibits convergence rates of 0.26-0.28 even for highly incompressible materials. Apart from the convergence experiments performed on our synthetic elastic box example, figures 6.4 and 6.5 include experiments performed on irregular geometries such as the armadillo model of figure 6.18 and the human character of figure 6.1. We further discuss the convergence rates and run times of these irregular models in section 6.3.2.

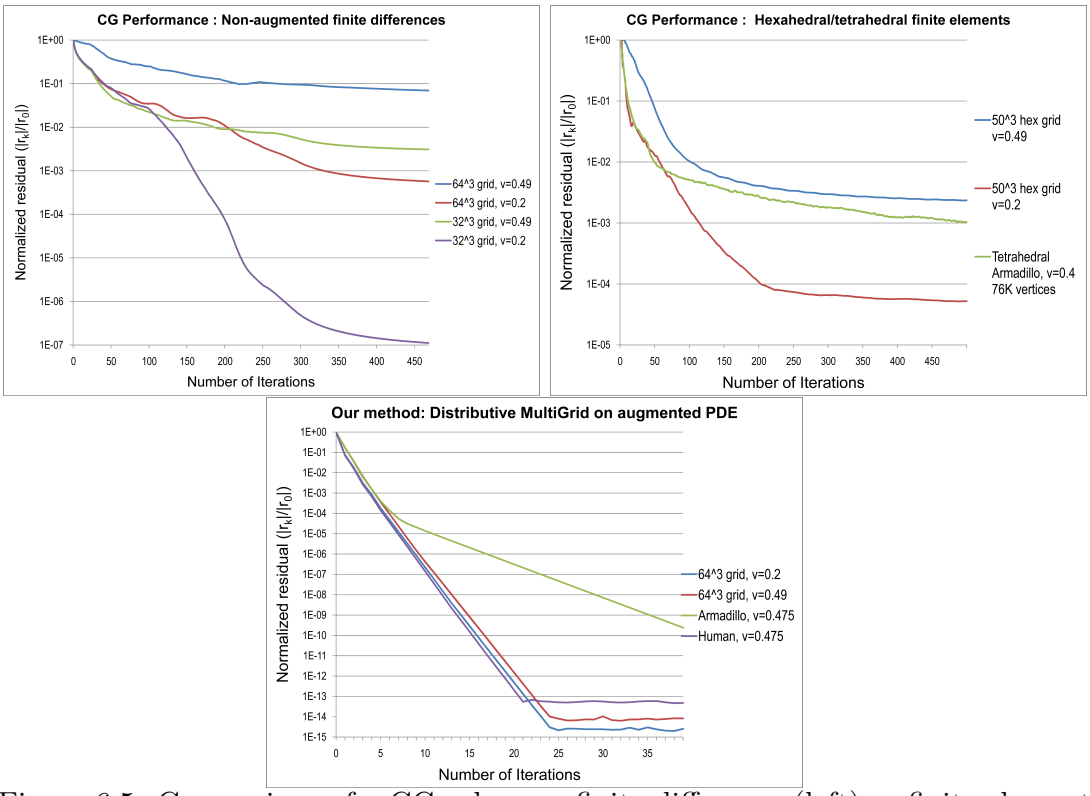


Figure 6.5: Comparison of a CG solver on finite difference (left) or finite element (middle) discretizations, with our proposed method (right)

6.3.1 Discretization accuracy analysis

Our method simulates objects of irregular shapes by embedding them in regular Cartesian lattices. Embedded simulation has been a popular method for physics-based animation, using either Cartesian lattices Müller et al. (2005b); Rivers and James (2007) for simplicity and efficiency, or tetrahedral embedding meshes for applications such as biomechanics Lee et al. (2009) and fracture modeling Molino et al. (2004). Although embedded models are computationally efficient and easy to generate, conforming meshes generally approximate the surface geometry of a model better than embedded models of comparable resolution. Several authors have proposed methods to compensate for this effect, for example by resolving collision and surface dynamics at a sub-element level Sifakis et al. (2007b) or

using an alternative interpolation method to generate the embedded surface for rendering Kaufmann et al. (2009b). In this section we evaluate the accuracy of our embedded method against a conforming discretization, and also compare our finite-difference method to an embedded discretization using finite elements.

For our accuracy analysis, we construct a 2D elasticity problem with an analytically known solution. Our testing model is the disc

$$D = \left\{(x, y) : (x - 0.5)^2 + (y - 0.5)^2 \leq 0.25^2\right\}$$

and is deformed according to the deformation function $\boldsymbol{\phi}(\boldsymbol{X})$=$(\phi_1, \phi_2)$ defined as

$$(\phi_1(x, y), \phi_2(x, y)) = \frac{2x}{\sqrt{\pi}} \left(\cos\left(\frac{\pi y}{2}\right), \sin\left(\frac{\pi y}{2}\right)\right). \tag{6.6}$$

We assume a linear elastic material. We substitute this analytic form of the deformation function $\boldsymbol{\phi}$ into the linear elasticity equations (3.3) to obtain the analytic expression of the elastic forces $\boldsymbol{f}$. We treat two quadrants of the outline of D (the thick shaded curves in figure 6.6) as Dirichlet boundary conditions, while the rest of the boundary (the two unshaded quadrants of the outline) are treated as traction boundaries. We analytically compute the traction value along

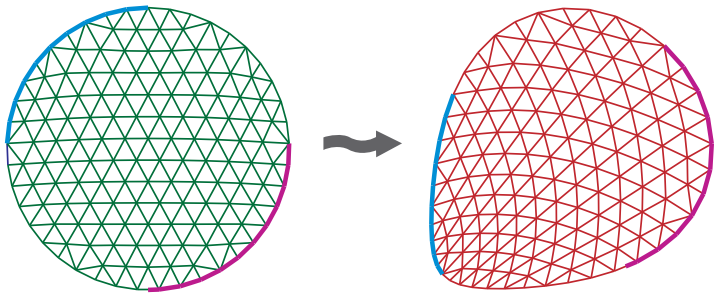


Figure 6.6: Illustration of the analytic deformation in our accuracy study. The thick shaded boundary sections indicate Dirichlet boundary conditions. The undeformed object is depicted on the left, the deformed state on the right.

the circular boundary of D as $\boldsymbol{t} = \mathbf{PN}$ where the stress $\mathbf{P}$ is computed from equation (3.1) and $\mathbf{N}$ is the outward pointing normal. Despite its geometric simplicity, this test problem highlights certain challenges related to embedding and discretization, especially for large values of Poisson's ratio, since the deformation prescribed in equation (6.6) incurs substantial change of volume in different parts of the domain (as seen in figure 6.6) giving rise to large elastic forces. We compare our embedded finite difference discretization with two finite element discretizations, one defined on a conforming tessellation of D, and one using an embedding triangular mesh, as illustrated in figure 6.7.

Since our test problem involves non-zero traction conditions on the embedded boundary (in contrast with our other examples in this thesis which use free boundaries, with zero traction) we need to treat this traction condition properly for the embedded finite element or finite difference grids. For these embedded discretizations, we start by approximating the circular boundary of D (the green curve in figure 6.7) with a polygonal curve. We subsequently compute a force for each segment of this polygonal curve that falls within the part of the boundary where traction conditions are given. This force is computed from the traction value as $\boldsymbol{f} = l \cdot \boldsymbol{t}$ where l is the length of the segment. We distribute half of this

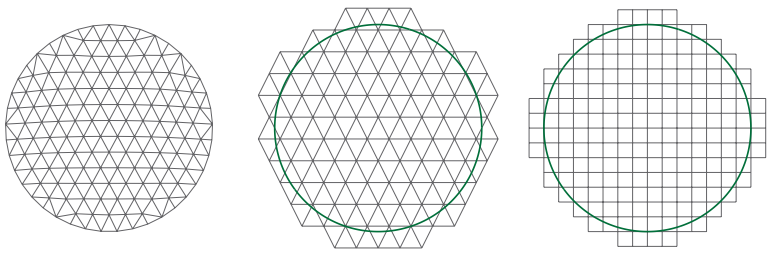


Figure 6.7: The three discretization methods in our comparative study. Left: A conforming tessellation, discretized with the finite element method. Middle: An embedded finite element discretization on a triangular mesh. Right: Our staggered finite difference method based on a Cartesian background lattice.

force to each endpoint of the segment; the traction condition is thus converted into individual forces on the vertices of the embedded boundary. Finally, we remap these forces from the polygonal boundary curve back to the degrees of freedom of the embedding simulation mesh. For a triangular embedding, this is accomplished by simply distributing the force from a vertex of the embedded boundary to the three vertices of its containing simulation triangle, weighted by the barycentric weights of the boundary location in the triangle (see figure 6.8, left). In our staggered, Cartesian embedding the x and y coordinates are embedded in two non-collocated lattices; thus, we distribute the x component of the force (denoted as f_1 in 6.8, right) to ϕ_1 grid locations, weighted by the bilinear embedding weights of the boundary location in this grid, while the y component of the force (denoted as f_2) will be similarly distributed to ϕ_2 grid locations. After this remapping, traction forces that have been mapped to locations of interior variables are scaled by $1/h^2$ (to remove the area weighting and convert them to force densities, as in the PDE form of elasticity) and then added to the right hand side of the discrete equation $\mathcal{L}\boldsymbol{\phi} = \boldsymbol{f}$, while forces mapped to boundary variable locations are converted back into traction conditions on the faces of the embedding grid as $t_i = f_i(N \cdot N')/h$, where N is the normal to the embedded boundary and N' is the normal to boundary face of the embedding grid. Notably, for free (zero-traction) boundaries, this treatment simply reduces to the method described in section 9.1.

Figure 6.10 illustrates the accuracy of the different discretization methods in our test, for different resolutions and degrees of incompressibility. Figure 6.9 plots the maximum error in the computed discrete solution under different levels of refinement. Since a discretization with order of accuracy equal to κ bounds the error as $|e| = O(h^\kappa)$ and $h \approx N^{-1/2}$ where N is the number of vertices in a uniform discretization, the asymptotic order of accuracy is approximated as

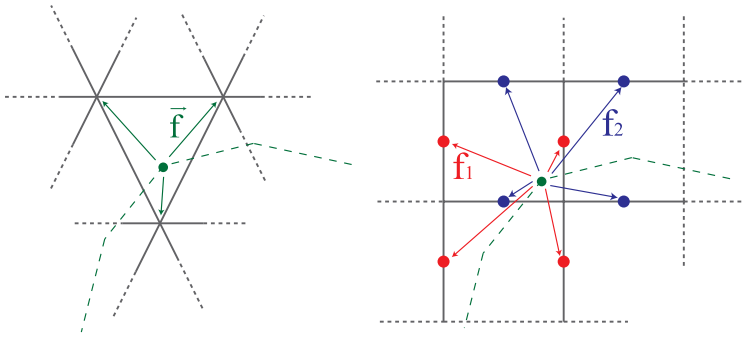


Figure 6.8: Left: Boundary traction forces are barycentrically distributed to the vertices of a triangular embedding mesh. Right: In our staggered discretization, each component of the traction force is bilinearly distributed to the grid locations of the respective (staggered) variable.

$\kappa \approx -2 \log |e| / \log N = -2m$, where m is the slope of the doubly-logarithmic plot of figure 6.9. We emphasize that the order of accuracy assessed in this section is completely independent of the convergence rate of the solver used for each discretization (see section 6.3). The discrete problems formulated in this section were solved to full convergence with an appropriate solver (multigrid or conjugate gradients). Our findings are summarized as follows:

- Although the different discretizations under consideration start with different levels of error for a base resolution, this error is asymptotically reduced at comparable rates under refinement. We estimated an asymptotic order of accuracy between $1.15-1.25$ from the tests plotted in figure 6.9. This approximate first-order accuracy is to be expected both from our finite difference scheme (due to the first-order treatment of the boundary) and the finite element discretizations (due to the use of first-order linear triangle elements, see e.g. Hughes (1987)).

- The conforming discretizations produced lower errors than both the finite difference, and finite element based embedded discretizations. For materi-

als with low Poisson's ratio, our proposed embedded method would require approximately $10-20$ times more degrees of freedom to match the accuracy of the conforming discretization. For near-incompressible materials this discrepancy is smaller, with our embedded method requiring approximately one extra grid refinement to reach the accuracy of the conforming method. Of course, a comparison of the degrees of freedom necessary for a given measure of accuracy does not necessarily reflect the computational cost of each approach. Our method typically leads to significantly reduced run times compared to conforming tetrahedral FEM models with the same number of degrees of freedom, due to the regularity of the discretization, convergence efficiency of the multigrid solver and improved numerical conditioning from our treatment of near-incompressibility. These performance benefits are less evident for low-resolution models (with up to a few thousand of degrees of freedom) where a conforming model, if available, may offer better accuracy per computation cost. For large models such as the ones demonstrated in our examples in section 6.3.2 our method can significantly outperform conforming tetrahedral meshes for the same degree of accuracy, even if our method requires a higher number of degrees of freedom to achieve the same accuracy. Finally, generating a good conforming tetrahedral model for detailed geometries such as those in figures 6.19 and 6.21 is a nontrivial meshing task which is not necessary in our embedded scheme.

- We also observe a tendency for the error on the embedded discretizations to be more oscillatory near the boundary, compared to the conforming case. These embedding artifacts are typically less pronounced with our method than with an embedded finite element approach on near-incompressible

materials (see figure 6.10), they are attenuated under refinement and can be significantly reduced in practice by slightly padding the embedding mesh outwards, typically by as little as one layer of cells (see figure 6.11).

- Our method matches or exceeds the accuracy of the embedded finite element discretization in our tests. The two embedded methods yield comparable accuracy for materials with low Poisson's ratio, especially in the asymptotic limit. For modest to high degrees of incompressibility, our method is noticeably more accurate and less prone to embedding artifacts than the embedded finite element discretization at the same resolution. Finally, our method performs similarly for materials of low and high incompressibility, although the embedded boundary surface tends to be slightly smoother for compressible materials.

Two important caveats should also be mentioned: The circular elastic body in our test had a smooth boundary surface which was well approximated by conforming tesselations even at low resolutions. Highly detailed models with intricate features (see e.g., figures 6.19 and 6.21) would incur significantly higher approximation errors for a conforming tesselation that does not descend to the resolution level necessary to resolve all the geometric detail. Secondly, in our embedded examples we used the analytic expression of the deformation field in equation (6.6) to specify Dirichlet boundary conditions directly on the vertices of the embedding meshes. This can be an acceptable practice for applications such as skeleton-driven characters, where kinematic constraints have a volumetric extent and can therefore be sampled at the locations of the simulation degrees of freedom. However, when Dirichlet conditions need to be specified at sub-grid locations and extrapolating these constraints to simulation vertices is not convenient, conforming meshes that resolve the constraint surface would be at an

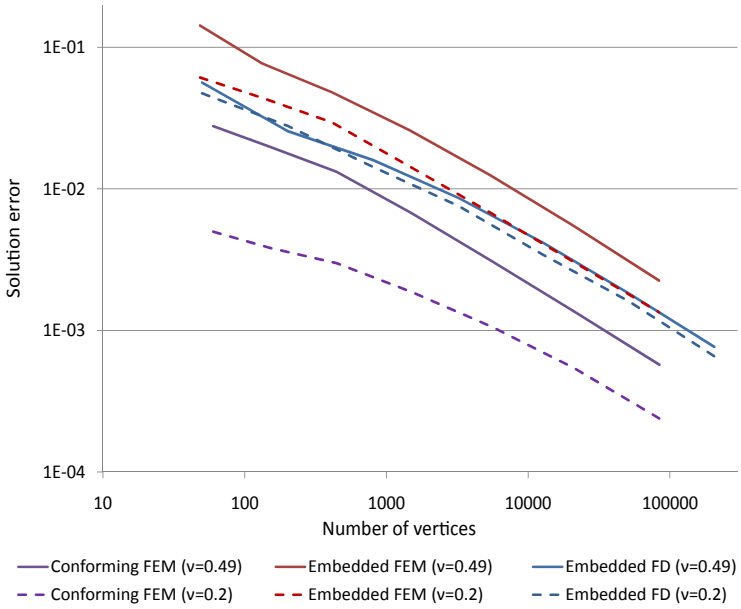


Figure 6.9: Plots of the maximum solution error for various discretizations. Solid lines indicate near-incompressible material parameters while dashed lines correspond to low Poisson's ratio. In this doubly-logarithmic plot, a slope of -0.5 indicates a first-order accurate method. The asymptotic order of accuracy observed from all six experiments ranges between $1.15-1.25$.

advantage. In future work we will investigate adding embedded soft-constraints in our framework (see e.g. Sifakis et al. (2007b)) to provide this additional flexibility. Finally, in our tests we considered discretizations of approximately uniform density (even when the mesh topology was irregular). It is also possible to use an adaptive discretization, either in the form of an adaptive conforming tessellation or an adaptive finite difference scheme (see e.g. Losasso et al. (2004)). In fact, there are well established multigrid methods that operate in conjunction with adaptive discretizations Brandt (1977a), and we believe the elasticity solver proposed in this thesis can be similarly applied to adaptive (e.g. octree) discretizations. We defer this extension to future work, along with a principled comparative evaluation of different adaptive discretization schemes for elasticity, especially in light of the nontrivial implications adaptivity may have on accuracy,

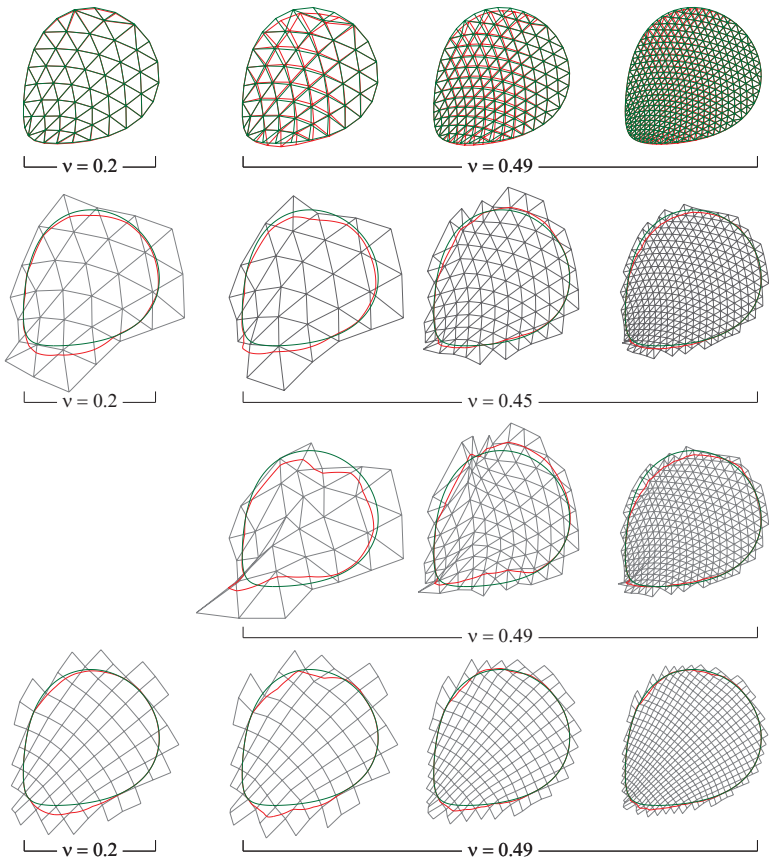


Figure 6.10: Convergence of different discretizations under refinement. The analytic solution is depicted in green, the discrete solution in red. The Poisson's ratio (ν) used for each experiment is given. Top row: A finite element discretization on a conforming triangle mesh. Compressible (ν=0.2) and near-incompressible (ν=0.49) cases are shown. Second and third row: Embedded finite element simulation on a triangle mesh. An additional case of moderate incompressibility (ν=0.45) is illustrated. Bottom row: Our embedded finite difference method. Note that both the embedded boundary and the background lattice are independently interpolated from the staggered deformation variables (not pictured). Also, the resolution in the rightmost column corresponds to approximately the same number of degrees of freedom for all discretizations.

numerical conditioning and potential for parallelization.

6.3.2 Animation tests

In addition to our comparative benchmarks, we tested our method on models with elaborate, irregular geometries. Figure 6.1 demonstrates the simulation of flesh of a human character with key-framed skeleton motion. The model was simulated at 2 resolutions yielding V-cycle times of 0.62sec for a 142K vertex model (pictured in figure 6.1), and 3.48sec for a larger resolution with 1.15M vertices (figure 6.11, right). The convergence rate for this example, as seen in Figures 6.5(right) and 6.4, was slightly better than our synthetic box examples at 0.24. We attribute this result to the extensive Dirichlet regions throughout the body induced by the kinematic skeleton, which stabilize the model and allow for highly efficient smoothing. In contrast, the armadillo model of figure 6.18 is very weakly constrained, with Dirichlet regions defined only over the hands and feet (see also Griebel et al. (2003) for a discussion of sub-optimal smoothing performance with dominating traction boundaries). In this model with extensive zero-traction boundary conditions, our method exhibited convergence rates between 0.21-0.35 for the first 7-8 V-cycles after a large perturbation; at that point the residual had been reduced by four orders of magnitude and the model had visually reached convergence after just the first few iterations. Subsequent V-cycles would ultimately settle at an asymptotic rate of 0.62 which could be improved by increasing the intensity of the boundary smoother, although this was not pursued since the model was already well converged and the extra smoothing cost would not be practically justified. With typical incremental motion of the boundary conditions, 1-2 V-cycles per frame would be enough to produce a visually converged animation.

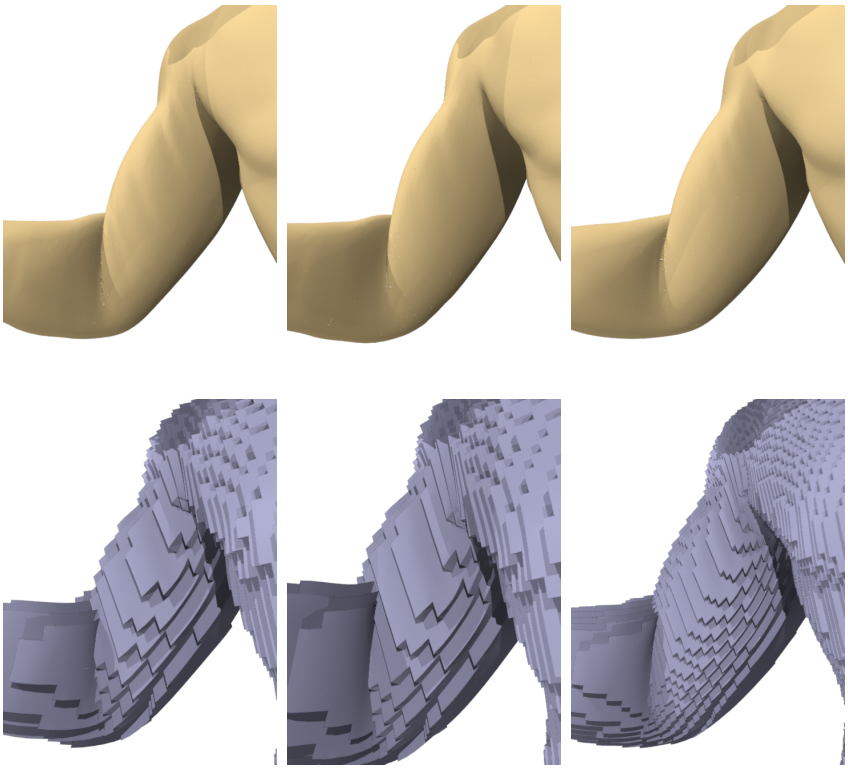


Figure 6.11: Closeup of the elbow joint from figure 6.1. Left: Grid spacing 9mm (142K vertices). Embedding artifacts are visible on the surface. Middle: Padding the embedding cage with one additional layer of cells visibly reduces the artifacts. Right: Surface artifacts are outright reduced using a higher resolution embedding cage (4.5mm spacing, 1.15M vertices).

Figure 6.4 also reports the convergence rates for the armadillo model of figure 6.18 simulated using co-rotational linear elasticity. Since the coefficients of the discrete system vary with the current configuration, the convergence rate is also variable. Additionally, the residual of the quasi-linearized system will differ from its actual non-linear counterpart; this discrepancy will also depend on whether the quasi-linearization process is close to convergence. The rates reported are typical of the animations shown, assuming 2 V-cycles per frame, and update of the quasi-linearization every 5 V-cycles. The average run time was 5.1sec per V-cycle, 10.2sec/frame (with 2 V-cycles). For comparison, we also simulated the

tetrahedral armadillo model of Teran et al. (2005b) using the quasistatic solver described in their paper. This tetrahedral model contains 380K tets and 76K vertices, thus contains approximately one quarter of the degrees of freedom of our embedded model in figure 6.18. For a Poisson's ratio of 0.4 each Newton-Raphson iteration (which includes a CG solve) required approximately 8.7sec while 5 Newton iterations per frame were required for acceptable convergence, leading to an approximate cost of 43.5sec/frame.

We also demonstrate examples of fully dynamic simulation. In figure 6.20, a 43K vertex car model is simulated using the static elasticity equations, as well as the dynamic scheme of section 6.2. As expected, the convergence rate for the Backward Euler system was significantly faster than our static problem (due to the addition of the identity term in the system matrix). Using a time step Δt equal to the frame time, our observed convergence rate was 0.08. Figure 6.21 illustrates the dynamic simulation of an elastic dragon figurine. The embedding grid has 402K cells/voxels (see Figure 5.2) and simulation cost is 8.2sec/frame. Figure 6.19 illustrates a high-velocity impact of a rigid body on a face model. The embedding grid contains 915K cells/voxels and simulation cost is 21sec/frame. We note that no explicit collision handling was performed for this example; instead, the degrees of freedom of the face that came in contact with the impacting object were kinematically prescribed to move with it for the duration of the impact. For these dynamic simulations just a single V-cycle per frame was sufficient, due to the better conditioning of the Backward Euler equations. Additionally, figure 6.16 provides a detailed breakdown of the execution cost of the individual subroutines on some of our benchmarks.

Finally, we note that usual trilinear interpolation would infrequently give rise to visual artifacts. Such artifacts would surface in simulations where the resolu-

Figure 6.12: Comparison of trilinear (left) and tricubic interpolation (right) on a coarse simulation. The embedding grid includes only 11K cells.

tion of the embedding grid was substantially coarser than that of the embedded surface (see, e.g. figure 6.12) and in conjunction with very extreme deformation. Since trilinear interpolation does not produce continuous derivatives, surface normals would exhibit visible discontinuities in these cases. We found that using tricubic interpolation as in Lekien and Marsden (2005) effectively eliminates this problem, as seen in figure 6.12. Notably, their method is based on evaluating higher-order derivatives at the nodes of the interpolation lattice, a process that is trivially implemented with finite differences in our regular discretization.

6.3.3 Parallelization

Our discretization of elasticity and the multigrid solver proposed in this thesis possess a number of characteristics that favor parallelism and scalable performance. The use of regular grids promotes locality, allows operations such as smoothing and transfer between grids to be implemented as streaming operations and allows for easy domain partitioning based on the background grid. Indirect

memory access is avoided since we do not use an explicit mesh to represent the simulated model. In addition, the regularity of the discrete equations enables a compact storage of the matrix in our linear system. For example, in the case of linear elasticity, all interior equations use the same stencil, thus there is no need to store a separate equation per grid location; this allows for a small memory footprint, even for large domains (see figure 6.17). In the case of co-rotational linear elasticity, only the rotation field needs to be stored and updated on the grid, while the system matrix can be built on-the-fly. We evaluated the potential of our algorithm for parallel performance by multi-threading a specific test problem on shared memory platforms. In this section we describe our parallelization methodology and present performance measurements.

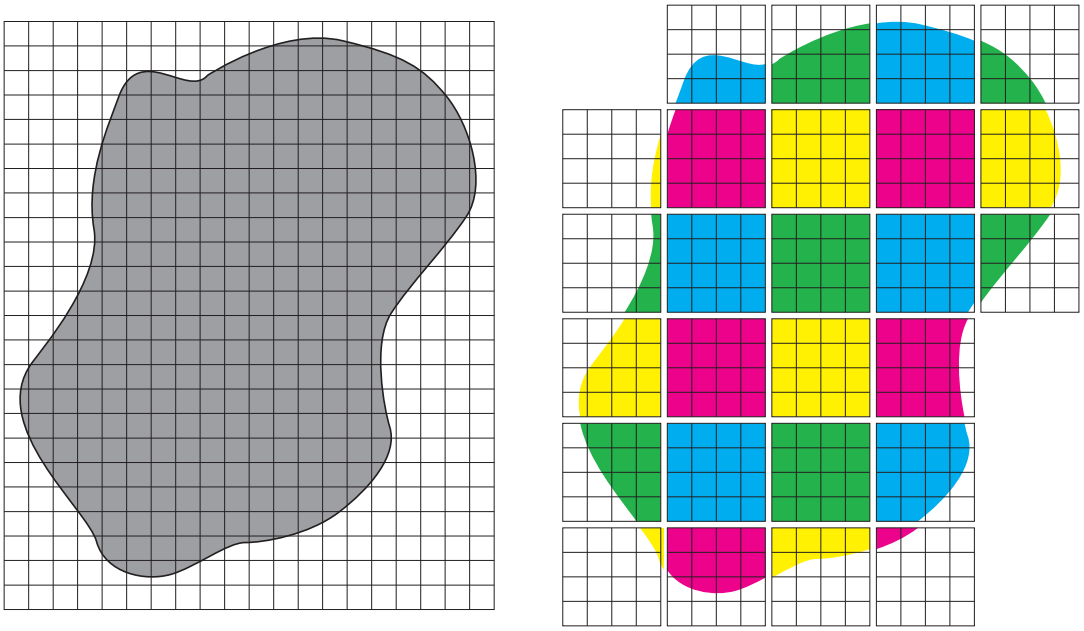


Figure 6.13: Volumetric partitioning using colored blocks in a 2D domain.

Parallelization of the components of the multigrid solver is based on an appropriate domain decomposition. Operations such as the interior (distributive) smoother, the computation and restriction of residuals, and prolongation of the

coarse-grid correction are performed throughout the volumetric extent of the simulated model. Consequently, these subroutines require a volumetric partitioning of the simulation domain. This is simply accomplished by partitioning the background cartesian grid into rectangular blocks, as depicted in the two-dimensional illustration of figure 6.13, which can then be processed by separate threads. However, operations such as the smoother incur data dependencies between neighboring blocks. In lieu of locking, we employ a coloring of the blocks (4 colors are used in the two-dimensional example of figure 6.13, 8 colors would be used in 3D) such that no two blocks of the same color are neighboring. All blocks of the same color can be processed in parallel without data dependencies, while different color groups are processed in sequence. The optimal block size depends on the desired number of blocks per color group (to allow simultaneous use of more processing threads) and the properties of the memory subsystem. For example, rectangular blocks of $16 \times 8 \times 8$ cells will align well with 512-bit cache lines ($16 \times$ 4-byte floats) while providing a few tens of blocks per color for problems in the order of 10^5 nodes.

Additionally, the rectangular shape of these blocks simplifies their traversal, as this can simply be performed with a triple loop over fixed index ranges. Such static loops yield improved cache and prefetching performance, and facilitate vectorization (either explicitly or as a compiler optimization). Care needs to be taken for blocks that include the domain boundaries, since some of their cells are outside the active simulation domain. In order to retain the benefits of traversing the block with a static triple loop, we perform the operation in question (i.e. smoothing, residual computation, inter-grid transfer) for all cells of a block, but only write the output of this operation conditionally on the value of a bitmap that indicates active/inactive grid locations. Finally, there are certain subroutines of our multigrid solver (e.g. boundary smoothing) that operate on a band around

the surface of the object. Since a volumetric partitioning could be inefficient and unbalanced for these operations, we perform a separate partitioning of the surface of the object. A chromatic grouping of these surface partitions is precomputed, as seen in figure 6.14, to allow all blocks within a color group to be processed in parallel without locking.

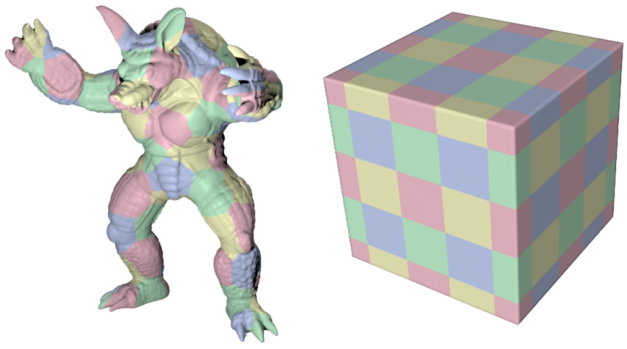


Figure 6.14: Surface partitioning of 3D models into colored surface patches.

CMP Scalability -- Problem size 64x64x64										
Kernel	Core 2	1c1t	1c4t	4c1t	4c4t	8c1t	8c4t	16c1t	32c1t	32c4t
Boundary smoothing	2.98	1.00	1.87	3.76	6.52	7.00	12.07	20.84	22.67	33.43
Distributive smoothing	2.96	1.00	1.84	3.90	6.86	7.55	13.44	13.79	28.37	24.51
Residual computation	3.79	1.00	1.85	3.95	7.24	7.82	14.11	14.75	29.61	26.58
Restriction	2.66	1.00	1.74	3.76	6.08	6.81	9.82	15.80	17.64	25.84
Prolongation	3.51	1.00	1.77	3.87	3.78	4.33	3.74	5.64	8.27	7.16
CMP Scalability -- Problem size 32x32x32										
Kernel	Core 2	1c1t	1c2t	1c4t	4c1t	4c4t	8c1t	16c1t		
Boundary smoothing	2.06	1.00	1.59	1.70	3.23	5.74	5.61	10.52		
Distributive smoothing	1.75	1.00	1.65	1.80	3.72	6.20	6.79	11.59		
Residual computation	2.42	1.00	1.63	1.76	3.66	6.31	7.08	12.55		

Figure 6.15: Parallel scaling on a Larrabee simulator for a number of different configurations. NcMt indicates a simulated platform with N cores and M threads/core. Prolongation/restriction were serialized on grids 32^3 and smaller. "Core 2" indicates the speedup of a single-threaded execution on an Intel Core 2 processor at the same clock frequency as the simulated platform.

We evaluated the potential of our algorithm for parallel performance using a

Single-core execution times of individual subroutines (in seconds)	32^3 Elastic Box (5 levels)	64^3 Elastic Box (6 levels)	128^3 Elastic Box (7 levels)	38K vertex Armadillo (4 levels)	302K vertex Armadillo (5 levels)	302K vertex Armadillo (corotational elasticity)	142K vertex Human (5 levels)	1.1M vertex Human (6 levels)
Boundary smoothing	0.052	0.197	0.711	0.217	0.890	1.281	0.275	1.074
Distributive (interior) smoothing	0.034	0.235	1.328	0.078	0.562	1.016	0.200	1.434
Residual computation	0.015	0.097	0.686	0.023	0.170	0.284	0.077	0.575
Residual restriction	0.001	0.012	0.088	0.002	0.015	0.014	0.010	0.079
Correction prolongation	0.007	0.040	0.240	0.011	0.096	0.099	0.036	0.294
Exact solver (at coarsest grid)	0.004	0.005	0.006	0.004	0.004	0.004	0.006	0.013
Approximate solution restriction	N/A	N/A	N/A	N/A	N/A	0.015	N/A	N/A
Update of nonlinear operator	N/A	N/A	N/A	N/A	N/A	2.37(*)	N/A	N/A
Other	0.001	0.001	0.002	0.001	0.001	0.006	0.012	0.014
Total	0.114	0.588	3.062	0.336	1.737	5.089	0.616	3.483

Figure 6.16: Single-core execution profiles. (*) The cost of the operator update was amortized based on 1 update every 5 V-cycles

multi-threaded version of our solver on the following shared-memory platforms: An 8-core SMP workstation with 3.0GHz Intel X5365 Xeon processors, a 16-core SMP server with 2.93GHz Intel X7350 CPUs, and a cycle-accurate performance simulator for the x86-based many-core Intel architecture codenamed Larrabee. Our benchmarks were based on our synthetic elastic box example, under high incompressibility ($\nu = .48$), with mixed boundary conditions and at resolutions ranging from 32^3 to 256^3 vertices. Figure 6.17 illustrates the speedup of our benchmarks, and associated working set sizes, on the 8- and 16-core SMPs. Figure 6.15 illustrates the speedup of individual subroutines on the Larrabee simulator, for two different problem sizes and at configurations up to 32 cores, with 4 threads/core. We note that the utilization of more than 1 thread per core does not increase the computational bandwidth (instructions are sequentially dispatched from different threads), but serves to hide instruction and memory latencies. Although we did not exploit the SIMD capacity of Larrabee in this experiment, the

memory utilization was at a low 0.5GB/Gcycles per core, demonstrating there is a substantial memory bandwidth margin to allow vectorization to further improve the performance of our solver.

Grid Size	32^3		64^3		128^3		256^3	
Threads	X5365	X7350	X5365	X7350	X5365	X7350	X5365	X7350
1	0.114	0.1172	0.588	0.624	3.06	3.12	18.27	18.61
2	0.0986	0.101	0.447	0.471	1.99	2.09	10.8	11.62
4	0.0793	0.0919	0.25	0.466	1.13	1.36	6.22	6.71
8	0.0709	0.0807	0.164	0.273	0.807	0.822	4.03	4.01
16		0.083		0.198		0.596		2.75
Working Set	26MB		45MB		180MB		1.2GB	

Figure 6.17: Scaling performance of the linear elasticity multigrid solver on multiprocessor systems (Intel X5365: 3.0Ghz, 8-core, 32GB RAM, Intel X7350: 3.0Ghz, 16-core, 32GB RAM). Measurements correspond to complete V-cycle times in seconds

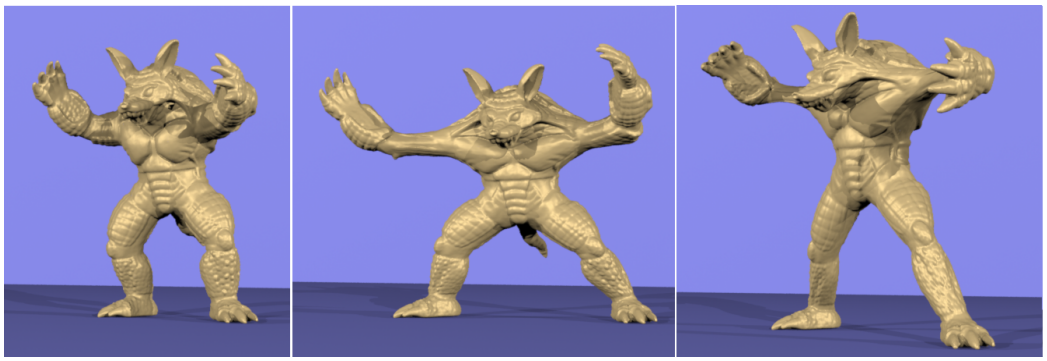


Figure 6.18: Quasistatic simulation of armadillo model with co-rotational linear elasticity. Resolution: 302K cells.

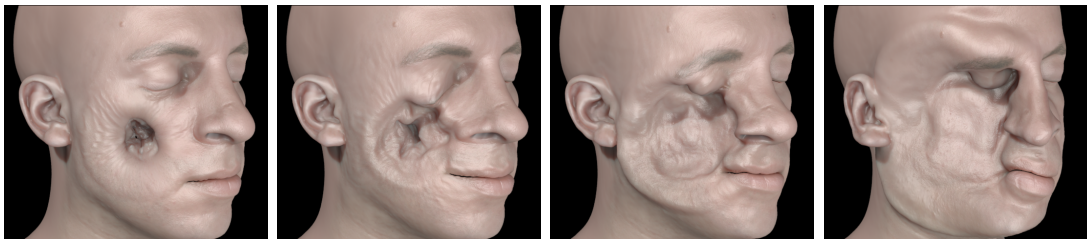


Figure 6.19: Dynamic simulation of an object impacting a face at high velocity. Modeled as a thick layer of flesh (no skull) using co-rotational linear elasticity. The embedding simulation cage contains 915K cells. The high-resolution embedded surface contains 10.1M triangles.

Figure 6.20: Dynamic simulation of a soft elastic car model deforming under kinematic constraints, using linear elasticity. Resolution: 43K cells.

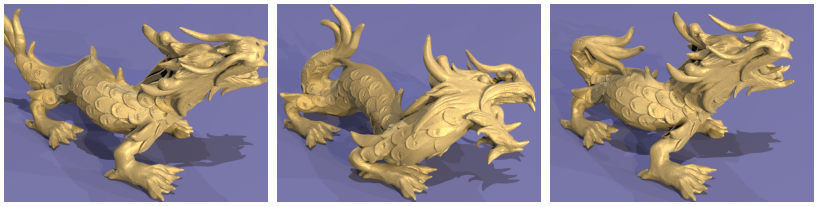


Figure 6.21: Embedded animation of a deformable dragon shaking his head, using co-rotational linear elasticity and simulation of dynamics. The embedded surface contains 7.2M triangles, while the simulated embedding cage has 402K cells (closeup pictured on the right).

CHAPTER 7

A second order mixed finite element method

In previous chapters, we developed a finite difference discretization and an efficient multigrid solver for linear elasticity problems with mixed boundary conditions. In the interior region, the discretization has a second order truncation error. Due to the inaccuracy in boundary discretizations, we achieve a first order accuracy in the solution. A first order accuracy in solution leads to a zeroth order accuracy in the stress. However, in cracking and fracture problems, an accurate stress estimation is of substantial importance. Also, a higher-order discretization leads to much more accurate solution without a significant increase in the problem set to resolve, hence leading to improvement of visual effects at a similar computation cost to afford.

While we can develope higher-order finite difference boundary discretization, it is tricky to achieve a symmetric discretization. Therefore, a lot of efficient linear algebra methods for symmetric systems do not apply. On the other hand, a finite element method naturally generates a symmetric system from a self-adjoint differential operator and maintains the operator's definiteness. Also, for coercive operators, it is easy to show that higher-order accuracy may naturally be achieved when discretized with higher-order finite elements. Moreover, the augmentation techniques introduced for stabilizing the near incompressible system is naturally derived using a mixed finite element method, which can be derived from a saddle point problem that is equivalent to the minimization of linear elasticity energy.

Therefore, we consider the finite element approach.

The staggered grid has been favorable in fluid simulation due to the sparsity and stability. The use of staggered grid in solids simulation is however not a standard. Low order finite element discretization of linear elasticity is known to be unstable for Stokes' equation. We choose a finite element space defined on a staggered grid, which is proved to be stable for Stokes' equation. We discretize the geometric domain by embedding it in a regular staggered grid. A Neumann boundary condition is naturally resolved. Also, a weak formulation for the Dirichlet boundary condition is developed.

We follow the idea of distributive relaxation and develop an efficient multigrid method. We solved a variety of problems, and verified the second order accuracy of the discretization.

7.1 Variational formulation for linear elasticity

We consider the two-dimensional linear elasticity problem defined on an arbitrary domain Ω.

$$-\nabla \cdot \sigma(\boldsymbol{u}) = \boldsymbol{f} \qquad \text{in } \Omega \tag{7.1}$$

$$\boldsymbol{u}|_{\Gamma_d} = \boldsymbol{u}_0 \qquad \text{on } \Gamma_d \tag{7.2}$$

$$(\sigma(\boldsymbol{u}) \cdot \boldsymbol{n})|_{\Gamma_n} = \boldsymbol{g} \qquad \text{on } \Gamma_n, \tag{7.3}$$

where $\boldsymbol{u} = \boldsymbol{\phi} - \mathbf{I} : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^2$ is the displacement, σ is the Cauchy stress tensor. and $\boldsymbol{f}$ is the elasticity force or the negated external force. $\boldsymbol{u}_0$ is a prescribed Dirichlet boundary condition, and $\boldsymbol{g}$ is the vector valued surface traction. In

linear elasticity, the stress $\sigma(\boldsymbol{u})$ depends linearly on the Cauchy strain $\boldsymbol{\epsilon}(\boldsymbol{u})$:

$$\begin{aligned} \boldsymbol{\epsilon}(\boldsymbol{u}) &= \frac{\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T}{2} && (7.4) \\ \sigma(\boldsymbol{u}) &= 2\mu\, \epsilon(\boldsymbol{u}) + \lambda\, \mathbf{tr}\, \epsilon(\boldsymbol{u})\, \mathbf{I} && (7.5) \\ &= \mu\, (\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T) + \lambda\, (\nabla \cdot \boldsymbol{u})\, \mathbf{I}. && (7.6) \end{aligned}$$

Therefore, the displacement satisfies the same equation as the deformation function in linear elasticity case with a different right hand side, and the equations of linear elastic equilibrium can be equivalently written as

$$\begin{aligned} -(\mu \Delta \mathbf{I} + (\lambda + \mu) \nabla \nabla^T) \boldsymbol{u} &= \boldsymbol{f} \quad \text{in } \Omega && (7.7) \\ \boldsymbol{u}|_{\Gamma_d} &= \boldsymbol{u}_0 \quad \text{on } \Gamma_d && (7.8) \\ \mu(\boldsymbol{u}_n + \nabla(\boldsymbol{u} \cdot \boldsymbol{n})) + \lambda(\nabla \cdot \boldsymbol{u})\boldsymbol{n}|_{\Gamma_n} &= \boldsymbol{g} \quad \text{on } \Gamma_n. && (7.9) \end{aligned}$$

A weak form for this PDE can be derived by taking an inner product with an arbitrary vector valued function $\boldsymbol{v} \in V_0 = (H^1_{0,\Gamma_d}(\Omega))^2$

$$\int_\Omega (\nabla \cdot \sigma(\boldsymbol{u})) \cdot \boldsymbol{v}\, d\boldsymbol{x} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}\, d\boldsymbol{x}. \tag{7.10}$$

By applying the divergence theorem to the left hand side, we have

$$\begin{aligned} \int_\Omega (\nabla \cdot \sigma(\boldsymbol{u})) \cdot \boldsymbol{v}\, d\boldsymbol{x} &= \int_\Omega \sigma(\boldsymbol{u})_{ij,j} v_i\, d\boldsymbol{x} = \int_\Omega (\sigma(\boldsymbol{u})_{ij} v_i)_{,j} - \sigma(\boldsymbol{u})_{ij} v_{i,j}\, d\boldsymbol{x} \\ &= \int_{\partial\Omega} \sigma(\boldsymbol{u})_{ij} v_i n_j\, ds - \int_\Omega \sigma(\boldsymbol{u})_{ij} v_{i,j}\, d\boldsymbol{x} \\ &= \int_{\partial\Omega} (\sigma(\boldsymbol{u}) \cdot \boldsymbol{n}) \cdot \boldsymbol{v}\, ds - \int_\Omega \sigma(\boldsymbol{u}) : (\nabla \boldsymbol{v})^T\, d\boldsymbol{x} \\ &= \int_{\Gamma_n} \boldsymbol{g} \cdot \boldsymbol{v}\, ds - \int_\Omega \sigma(\boldsymbol{u}) : (\nabla \boldsymbol{v})^T\, d\boldsymbol{x}. \end{aligned} \tag{7.11}$$

By the definition and the symmetry of $\sigma(\boldsymbol{u})$,

$$\begin{aligned}
&\int_\Omega \sigma(\boldsymbol{u}) : (\nabla \boldsymbol{v})^T \, d\boldsymbol{x} \\
= &\int_\Omega \mu(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T) : (\nabla \boldsymbol{v})^T + \lambda(\nabla \cdot \boldsymbol{u})(\nabla \cdot \boldsymbol{v}) \, d\boldsymbol{x} \\
= &\int_\Omega \frac{\mu}{2}(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T) : (\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T) + \lambda(\nabla \cdot \boldsymbol{u}) \, (\nabla \cdot \boldsymbol{v}) \, d\boldsymbol{x} \\
= &\int_\Omega 2\mu\epsilon(\boldsymbol{u}) : \epsilon(\boldsymbol{v}) + \lambda(\nabla \cdot \boldsymbol{u}) \, (\nabla \cdot \boldsymbol{v}) \, d\boldsymbol{x}.
\end{aligned}$$

Substituting into (7.10) and (7.11), we have the following variational form:

$$\begin{aligned}
&\text{find } \boldsymbol{u} \in H^1(\Omega) \times H^1(\Omega), \boldsymbol{u}|_{\Gamma_d} = \boldsymbol{u}_0, \text{ such that} \\
&\int_\Omega 2\mu \left(\frac{\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T}{2} \right) : \left(\frac{\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T}{2} \right) + \lambda(\nabla \cdot \boldsymbol{u})(\nabla \cdot \boldsymbol{v}) \, d\boldsymbol{x} \\
&\quad = -\int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, d\boldsymbol{x} + \int_{\Gamma_n} \boldsymbol{g} \cdot \boldsymbol{v} \, ds \quad \forall \boldsymbol{v} \in H^1_{0,\Gamma_d}(\Omega) \times H^1_{0,\Gamma_d}(\Omega). \quad (7.12)
\end{aligned}$$

This problem is equivalent to the following energy minimization problem defined over the domain $V = \{\boldsymbol{u} \in (H^1(\Omega))^2 : \boldsymbol{u}|_{\Gamma_d} = \boldsymbol{u}_0\}$.

$$\begin{aligned}
\inf_{\substack{\boldsymbol{u} \in (H^1(\Omega))^2 \\ \boldsymbol{u}|_{\Gamma_d} = \boldsymbol{u}_0}} &\left\{ \int_\Omega \left(\mu \left| \frac{\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T}{2} \right|_F^2 + \frac{\lambda}{2}(\nabla \cdot \boldsymbol{u})^2 \right) d\boldsymbol{x} \right. \\
&\left. + \int_\Omega \boldsymbol{f} \cdot \boldsymbol{u} \, d\boldsymbol{x} - \int_{\Gamma_n} \boldsymbol{g} \cdot \boldsymbol{u} \, ds \right\}. \quad (7.13)
\end{aligned}$$

When supporting irregular domain geometries, a natural choice for the numerical approximation of these equations is the finite element method (FEM) with unstructured meshes that conform to the geometry of $\partial\Omega$. However, meshing complex geometries can prove difficult and time-consuming when the boundary frequently changes shape. This task is even more difficult when using the more

elaborate element types seen in mixed FEM formulations. Mixed formulations are typically necessary for stability with the nearly incompressible materials we consider here. In many applications, such as shape optimization for elastic materials, it is necessary to change the geometry of the domain at each iteration of a simulation. In such cases, frequent unstructured remeshing can be prohibitively costly (especially in 3D). Also, many numerical methods, such as finite differences, do not naturally apply to unstructured meshes. These concerns motivated the development of "embedded" (or, "immersed") methods that approximate solutions to (7.7) on Cartesian grids or structured meshes that do not conform to the boundary. Retention of higher-order accuracy in L^∞-norm with such embedding strategies is an ongoing area of research.

With these concerns in mind, we introduce a second order virtual node method for approximating the equations of linear elastic equilibrium with irregular embedded Neumann and Dirichlet boundaries on a uniform Cartesian grid. We use a regular grid because it simplifies the implementation, permits straightforward Lagrange multiplier spaces for Dirichlet constraints and naturally allows for higher-order accuracy in L^∞. The method is most suited for problems like level set-based shape optimization where the geometry of the domain is frequently changing and constant remeshing is a clearly inferior alternative to embedding. Also, we allow for nearly incompressible materials 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 pressures at cell centers. Our approach combines piecewise bilinear interpolation of displacement components with the addition of "virtual" nodes on cut cells that accurately account for the irregular shape of the geometry boundary. The variational nature of the method naturally enables symmetric numerical stencils at the boundary. We use Lagrange multipliers to

enforce embedded Dirichlet conditions weakly. In the general case, our choice of Lagrange multiplier space admits an efficient means for smoothing boundary equations in our geometric multigrid method for solving the discretized equations. Numerical experiments indicate second order accuracy in L^∞ independent of Poisson's ratio, domain geometry or boundary condition type.

7.2 Mixed finite element formulation

In augmented linear elasticity, we introduce a pressure variable to achieve a stable numerical system for the near-incompressible linear elasticity problem, and the weak formulation for augmented linear elasticity can be derived via many approaches. Duality formulations allow a natural extension to more general cases, and we will follow Brezzi and Fortin (1991) in this approach.

We start with the following equality.

$$\frac{\lambda}{2}\int_\Omega p^2\,d\boldsymbol{x} = \sup_{q\in L^2(\Omega)}\int_\Omega pq\,d\boldsymbol{x} - \frac{1}{2\lambda}\int_\Omega q^2\,d\boldsymbol{x},\ \forall p\in L^2(\Omega).$$

First, let us denote $p=-\nabla\cdot\boldsymbol{v}$, and apply change of variable to replace q with μp. Substituting into (7.13) for the term $\frac{\lambda}{2}\int_\Omega(\nabla\cdot\boldsymbol{v})^2\,d\boldsymbol{x}$, we achieve the following saddle point problem, which has exactly the same solution $\boldsymbol{u}$ as the linear elasticity energy minimization problem.

$$\inf_{\substack{\boldsymbol{u}\in(H^1(\Omega))^2\\ \boldsymbol{u}|_{\Gamma_d}=\boldsymbol{u}_0}}\ \sup_{p\in L^2(\Omega)}\ \{\int_\Omega\left(\mu\left|\frac{\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T}{2}\right|_F^2 - \mu p(\nabla\cdot\boldsymbol{u}) - \frac{\mu^2}{2\lambda}p^2\right)d\boldsymbol{x} + \int_\Omega \boldsymbol{f}\cdot\boldsymbol{u}\,d\boldsymbol{x} - \int_{\Gamma_n}\boldsymbol{g}\cdot\boldsymbol{u}\,ds\quad\}. \tag{7.14}$$

The equilibrium solution to this problem is also the solution of the following weak problem:

$$
\begin{aligned}
&\text{Find } (\boldsymbol{u}, p) \in H^1(\Omega) \times H^1(\Omega) \times L^2(\Omega), \boldsymbol{u}|_{\Gamma_d} = \boldsymbol{u}_0, \text{ such that} \\
&\int_\Omega 2\mu \left(\frac{\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T}{2} \right) : \left(\frac{\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T}{2} \right) - \mu p (\nabla \cdot \boldsymbol{v}) \, d\boldsymbol{x} \\
&\quad = - \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, d\boldsymbol{x} + \int_{\Gamma_n} \boldsymbol{g} \cdot \boldsymbol{v} \, ds \quad \forall \boldsymbol{v} \in H^1_{0,\Gamma_d}(\Omega) \times H^1_{0,\Gamma_d}(\Omega) \qquad (7.15) \\
&\int_\Omega \left(-\mu q \nabla \cdot \boldsymbol{u} - \frac{\mu^2}{\lambda} pq \right) d\boldsymbol{x} = 0 \quad \forall q \in L^2(\Omega). \qquad (7.16)
\end{aligned}
$$

Notice that this weak problem is consistent with the augmented linear elasticity. By introducing $p = -\lambda/\mu \nabla \cdot \boldsymbol{u}$, and replace the regular stress with an augmented stress $\sigma(\boldsymbol{u}) = \mu \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T \right) - \mu \, p \, \mathbf{I}$, which is equivalent to the linear elasticity stress when $p = -\lambda/\mu \nabla \cdot \boldsymbol{u}$, we derive the augmented linear elasticity problem.

$$
\begin{aligned}
\mu(\Delta \mathbf{I} + \nabla \nabla^T)\boldsymbol{u} - \mu \nabla p &= \boldsymbol{f} \quad \text{in } \Omega \qquad (7.17) \\
\mu \nabla \cdot \boldsymbol{u} + \frac{\mu^2}{\lambda} p &= 0 \quad \text{in } \Omega \qquad (7.18) \\
\boldsymbol{u}|_{\Gamma_d} &= \boldsymbol{u}_0 \quad \text{on } \Gamma_d \qquad (7.19) \\
\mu(\boldsymbol{u}_n + \nabla(\boldsymbol{u} \cdot \boldsymbol{n})) - \mu p \, \boldsymbol{n}|_{\Gamma_n} &= \boldsymbol{g} \quad \text{on } \Gamma_n. \qquad (7.20)
\end{aligned}
$$

By taking inner product of (7.17) with an arbitrary test function $\boldsymbol{v} \in H^2_{0,\Gamma_d}$ and integrating over domain Ω, we obtain the first equation in (7.14). By multiplying (7.18) with $q \in L^2(\Omega)$ and integrating over Ω, we obtain the second equation in (7.14).

7.2.1 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 Han and Wu (1998). We generalize this approach to the case of nearly incompressible linear elasticity in embedded domains. We approximate the Sobolev space V with a finite element subspace V^h, where each displacement component of a function in V^h is represented as a piecewise bilinear scalar function defined on a staggered quadrilateral grid (see Figure 7.1). To be more specific, consider the staggered grids:

We approximate the Sobolev space V with a finite element subspace V^h, where each component of V^h is approximated by a piecewise bi-linear finite element space defined on a staggered quadrilateral grid, as demonstrated in Figure 7.1. To be more specific, consider the staggered grids

$$\mathcal{G}_h^x = \{(ih, (j-1/2)h) : (i,j) \in I_x \subset Z^2\}$$

$$\mathcal{G}_h^y = \{((i-1/2)h, jh) : (i,j) \in I_y \subset Z^2\}.$$

Here, h is the discrete spacing between grid points. Furthermore, we use the following notation to denote quadralaterals 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 I_x and 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 quadralateral T_{ij}^x or T_{ij}^y respectively whose

intersection with the domain Ω is non-empty. In other words, I_x and I_y are the sets of vertices in the staggered lattices that are at most a distance of h away from Ω. Henceforth, we will use

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

and

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

to denote the collection of x and y grid quadralaterals that intersect (or embed) the domain Ω. We construct two subspaces of $H^1(\Omega)$ based on these quadrangulations respectively:

$$V_x^h = \{v_h \in C^{(0)}(\Omega) : v_h|_{T_{ij}^x} \in Q_1(T_{ij}^x) \quad \forall T_{ij}^x \in \mathcal{T}_h^x \text{ s.t. } T_{ij}^x \cap \Omega \neq \emptyset\}$$

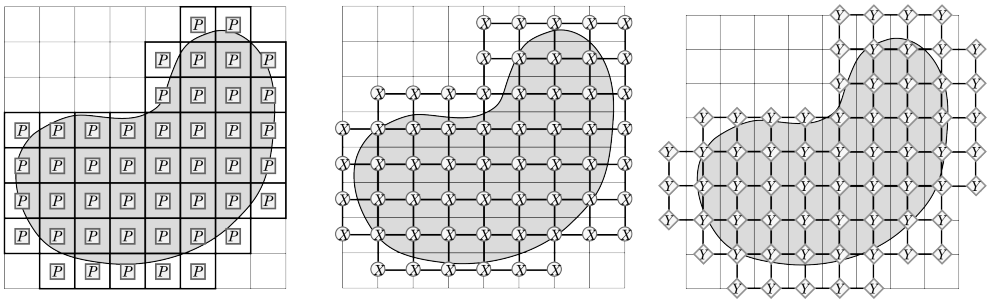
$$V_x^h = \{v_h \in C^{(0)}(\Omega) : v_h|_{T_{ij}^y} \in Q_1(T_{ij}^y) \quad \forall T_{ij}^y \in \mathcal{T}_h^y \text{ s.t. } T_{ij}^y \cap \Omega \neq \emptyset\}.$$

For simplicity of notation in subsequent equations we will also use mappings $\eta_1 : I_1 = \{1, 2, ..., N_x\} \rightarrow I_x$ and $\eta_2 : I_2 = \{1, 2, ..., N_y\} \rightarrow I_y$ to associate each x and y grid vertex with a unique integer between 1 and N_x and 1 and N_y respectively. With this convention, any approximated solution $\boldsymbol{u} \in V_x^h \times V_y^h$ can be expressed as

$$\boldsymbol{u}(\boldsymbol{x}) \ = \ \begin{pmatrix} \sum_{k_1 \in I_1} u_{1k_1} N_{1k_1}(\boldsymbol{x}) \\ \sum_{k_2 \in I_2} u_{2k_2} N_{2k_2}(\boldsymbol{x}) \end{pmatrix}, \tag{7.21}$$

where N_{1k_1} and N_{2k_2} are the commonly used piecewise bilinear interpolating 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 $\boldsymbol{u}$ can thus be seen to be over $N_x + N_y$ unknowns.



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

Figure 7.1: Staggered grid finite element quadrangulation and embedded domain boundary.

We additionally approximate the space for pressure $V_p = L^2(\Omega)$ 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) $\mathcal{G}_h^p$:

$$\mathcal{G}_h^p = \{((i + \frac{1}{2})h, (j + \frac{1}{2})h) : (i, j) \in I_p \subset Z^2\}$$

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

$$\mathcal{T}_h^p = \{T_{ij}^p : T_{ij}^p \cap \Omega \neq \emptyset\}$$

$$V_p^h = \{p_h \in L^2(\Omega) : p_h|_{T_{ij}^p} \in P_0(T_{ij}^p) \quad \forall T_{ij}^p \in \mathcal{T}_h^p \text{ s.t. } T_{ij}^p \cap \Omega \neq \emptyset\}$$

The grid $\mathcal{G}_h^p$ is a cell-centered grid (as opposed to node-centered grids $\mathcal{G}_h^x$ and $\mathcal{G}_h^y$). That is, we assume that pressure variables live at the cell centers of this grid. In other words, there is one pressure variable located in each $T_{ij}^p \in \mathcal{T}_h^p$. I_p is defined similarly to I_x and I_y, however here it refers to the collection of cell centered indices in the grid $\mathcal{G}_h^p$ whose associated quadralaterals T_{ij}^p have a non-zero intersection with Ω. For the sake of simplicity in subsequent equations,

we again use a mapping $\eta_3 : I_3 = \{1, 2, ..., N_p\} \to I_p$ to associate each cell in the pressure grid with a unique integer between 1 and N_p. Thus, any approximated solution of pressure has the following representation:

$$p(\boldsymbol{x}) = \sum_{k_3 \in I_3} p_{k_p} \chi_{\mathcal{T}^p_{k_p}}(\boldsymbol{x}) \tag{7.22}$$

where $\chi_{\mathcal{T}^p_{k_p}}(\boldsymbol{x})$ is the characteristic function for the quadralateral $T^p_{k_p}$. That is,

$$\chi_{\mathcal{T}^p_{k_p}}(x) = \begin{cases} 1, & \boldsymbol{x} \in T^p_{k_p} \\ 0, & \boldsymbol{x} \notin T^p_{k_p} \end{cases}$$

We choose test functions $\boldsymbol{v}(\boldsymbol{x}) = N_{mk_m}(\boldsymbol{x})\boldsymbol{e}_m, m = 1, 2$ and substitute the finite element discretization (7.21),(7.22) into each term in the mixed variational form (7.15),

$$\begin{aligned}
& 2\mu \int_\Omega \frac{\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T}{2} : \frac{\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T}{2}\, d\boldsymbol{x} \\
= \;& \mu \int_\Omega \nabla \boldsymbol{u} : (\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T)\, d\boldsymbol{x} \\
= \;& \mu \sum_{i,j \in \{1,2\}} \int_\Omega u_{i,j}(v_{i,j} + v_{j,i})\, d\boldsymbol{x} = \mu \sum_{i,j \in \{1,2\}} \int_\Omega u_{i,j}(N_{mk_m,j}(\boldsymbol{x})\delta_{mi} + N_{mk_m,i}(\boldsymbol{x})\delta_{mj})\, d\boldsymbol{x} \\
= \;& \mu \sum_{i \in \{1,2\}} \int_\Omega \left(\sum_{j \in \{1,2\}} u_{i,j} N_{mk_m,j}(\boldsymbol{x})\delta_{mi} + u_{i,m} N_{mk_m,i}(\boldsymbol{x}) \right) d\boldsymbol{x} \\
= \;& \mu \sum_{i \in \{1,2\}} \sum_{k_i \in I_i} \int_\Omega u_{ik_i} \left(\sum_{j \in \{1,2\}} N_{ik_i,j}(\boldsymbol{x}) N_{mk_m,j}(\boldsymbol{x})\delta_{mi} + N_{ik_i,m}(\boldsymbol{x}) N_{mk_m,i}(\boldsymbol{x}) \right) d\boldsymbol{x} \\
= \;& \mu \sum_{i \in \{1,2\}} \sum_{k_i \in I_i} u_{ik_i} \left(\delta_{mi} \int_\Omega \sum_{j \in \{1,2\}} N_{mk_i,j}(\boldsymbol{x}) N_{mk_m,j}(\boldsymbol{x})\, d\boldsymbol{x} + \int_\Omega N_{ik_i,m}(\boldsymbol{x}) N_{mk_m,i}(\boldsymbol{x})\, d\boldsymbol{x} \right)
\end{aligned}$$

$$
\begin{aligned}
-\mu \int_\Omega p \nabla \cdot \boldsymbol{v} d\boldsymbol{x} &= -\mu \sum_{k_p \in I_p} p_{k_p} \int_{T^p_{p_k} \cap \Omega} N_{mk_m,m}(\boldsymbol{x}) d\boldsymbol{x} \\
\int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}\, d\boldsymbol{x} &= \int_\Omega f_m N_{mk_m}(\boldsymbol{x})\, d\boldsymbol{x} \\
\int_{\Gamma_n} \boldsymbol{g} \cdot \boldsymbol{v}\, ds &= \int_{\Gamma_n} g_m N_{mk_m}(\boldsymbol{x})\, ds
\end{aligned}
$$

We can also choose $v = 0$ and $q(\boldsymbol{x}) = \chi_{\mathcal{T}^p_{k_p}}(\boldsymbol{x})$ to derive the pressure equations:

$$
-\mu \sum_{k_i \in I_i} u_{ik_i} \int_{T^p_{k_p} \cap \Omega} N_{ik_i,i}(\boldsymbol{x})\, d\boldsymbol{x} - \frac{\mu^2}{\lambda} \sum_{k_p \in I_p} p_{k_p} \int_{T^p_{k_p} \cap \Omega} 1\, d\boldsymbol{x} = 0
$$

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

$$
\begin{pmatrix} \mathbf{L}^h_u & \mathbf{G}^{h^T} \\ \mathbf{G}^h & \mathbf{D}^h_p \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^h \\ \boldsymbol{p}^h \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}^h \\ \mathbf{0} \end{pmatrix} \qquad \text{or} \qquad \hat{\mathbf{L}}^h \hat{\boldsymbol{u}}^h = \hat{\boldsymbol{f}}^h \tag{7.23}
$$

where $\hat{\boldsymbol{u}}^h = (\boldsymbol{u}^h, p^h)$ and $\hat{\boldsymbol{f}}^h = (\boldsymbol{f}^h, 0)$. Furthermore, our use of regular grids gives the discrete equations a finite difference interpretation. If we scale the system by $-\frac{1}{h^2}$, each block in the discrete system approximates the corresponding differential operator in (7.17), i.e. (7.23) discretizes the following equation:

$$
h^2 \begin{pmatrix} -\mu(\Delta + \nabla\nabla^T) & \mu\nabla \\ -\mu\nabla^T & -\frac{\mu^2}{\lambda} \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ p \end{pmatrix} = \begin{pmatrix} -\boldsymbol{f}h^2 \\ 0 \end{pmatrix} \tag{7.24}
$$

The linear system is the Hessian matrix of a saddle point problem, therefore

although the discretized system is symmetric but indefinite. In fact, the first diagonal block $\mathbf{L}_u^h$ is negative definite, and the other diagonal block is positive definite.

7.2.2 Implementation details

For ease of implementation, we perform the integrations involved in the discrete equations 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 quadralateral of the pressure grid with the embedded domain $T_{k_p}^p \cap \Omega$:

$$\int_\Omega N_{mk_i,j}(\boldsymbol{x}) N_{mk_m,j}(\boldsymbol{x})\, d\boldsymbol{x} = \sum_{k_p \in I_p} \int_{T_{k_p}^p \cap \Omega} N_{mk_i,j}(\boldsymbol{x}) N_{mk_m,j}(\boldsymbol{x})\, d\boldsymbol{x}$$

$$\int_\Omega N_{ik_i,m}(\boldsymbol{x}) N_{mk_m,i}(\boldsymbol{x})\, d\boldsymbol{x} = \sum_{k_p \in I_p} \int_{T_{k_p}^p \cap \Omega} N_{ik_i,m}(\boldsymbol{x}) N_{mk_m,i}(\boldsymbol{x})\, d\boldsymbol{x}$$

$$\int_\Omega N_{mk_m}(\boldsymbol{x})\, d\boldsymbol{x} = \sum_{k_p \in I_p} \int_{T_{k_p}^p \cap \Omega} N_{mk_m}(\boldsymbol{x})\, d\boldsymbol{x}$$

$$\int_{\Gamma_n} N_{mk_m}(\boldsymbol{x})\, ds = \sum_{k_p \in I_p} \int_{T_{k_p}^p \cap \Gamma_n} N_{mk_m}(\boldsymbol{x})\, ds.$$

In the interior, this simply amounts to evaluating the same integrals over each full quadralateral $T_{k_p}^p$. However, at the boundary, care must be taken to respect the material region alone when the intersection between the pressure cells and the embedded domain is non-trivial. In either of the boundary, or interior cases, there will be 13 degrees of freedom involved in the integration over such a pressure cell. This is because the staggering of variables leads to 13 interpolating functions supported over a given pressure cell (6 x-components, 6 y-components and one

pressure). In other words, we express the matrix (or stiffness matrix) in our discrete system as a sum of 13×13 element stiffness matrices $\mathbf{A}^{k_p}$. Furthermore, we break the integrals involved in a given element $T^p_{k_p}$ up into four subintegrals over the subquadrants $(\omega_1, \omega_2, \omega_3, \omega_4)$ of $T^p_{k_p}$ (see Figure 7.2. This is because the integrands are all smooth over these regions. Notably, they are quadratic and we simply preform these integrations analytically. The elemental stiffness matrix is accumulated from the following sub-stiffness-matrices $\mathbf{A}^{k_p} = \mathbf{A}^{k_p}_{\omega_1} + \mathbf{A}^{k_p}_{\omega_2} + \mathbf{A}^{k_p}_{\omega_3} + \mathbf{A}^{k_p}_{\omega_4}$. For example, $\mathbf{A}^{k_p}_{\omega_1}$ involves x_1, x_2, x_3, x_4, y_7, y_8, y_{10} and y_{11} as demonstrated in Figure 7.2 right), therefore, it only has non-zero values on rows and columns involving these degrees of freedom. If we order the 13 nodes with indices shown in Figure 7.2 left, then

$\mathbf{A}^{k_p}_{\omega_1} =$

i \ j	X_1	X_2	X_3	X_4	Y_7	Y_8	Y_{10}	Y_{11}	P_{13}
X_1	$\mu\int_{\omega_1} \nabla N_{xk_i}\cdot\nabla N_{xk_j} + N_{xk_i,x}N_{xk_j,x}\,dx$				$\mu\int_{\omega_1} N_{xk_i,y}N_{yk_j,x}dx$				$-\mu\int_{\omega_1} N_{xk_i,x}dx$
X_2									
X_3									
X_4									
Y_7	$\mu\int_{\omega_1} N_{yk_i,x}N_{xk_j,y}dx$				$\mu\int_{\omega_1} \nabla N_{yk_i}\cdot\nabla N_{yk_j} + N_{yk_i,y}N_{yk_j,y}\,dx$				$-\mu\int_{\omega_1} N_{yk_i,y}dx$
Y_8									
Y_{10}									
Y_{11}									
P_{13}	$-\mu\int_{\omega_1} N_{xk_j,x}dx$				$-\mu\int_{\omega_1} N_{yk_j,y}dx$				$-\mu^2/\lambda\int_{\omega_1} 1dx$

On the interior of the domain, where $T^p_{k_p} \cap \Omega = T^p_{k_p}$, the sum of these four

subintegrals is always the same:

$$\mathbf{A}^{k_p} = \begin{pmatrix} \mu \begin{pmatrix} \frac{1}{4} & 0 & 0 & -\frac{1}{4} & 0 & 0 & \frac{9}{64} & -\frac{3}{32} & -\frac{3}{64} & \frac{3}{64} & -\frac{1}{32} & -\frac{1}{64} \\ 0 & \frac{1}{4} & -\frac{1}{4} & 0 & 0 & 0 & \frac{3}{64} & \frac{3}{32} & -\frac{9}{64} & \frac{1}{64} & \frac{1}{32} & -\frac{3}{64} \\ 0 & -\frac{1}{4} & \frac{3}{2} & -1 & 0 & -\frac{1}{4} & -\frac{3}{32} & \frac{1}{16} & \frac{1}{32} & \frac{3}{32} & -\frac{1}{16} & -\frac{1}{32} \\ -\frac{1}{4} & 0 & -1 & \frac{3}{2} & -\frac{1}{4} & 0 & -\frac{1}{32} & -\frac{1}{16} & \frac{3}{32} & \frac{1}{32} & \frac{1}{16} & -\frac{3}{32} \\ 0 & 0 & 0 & -\frac{1}{4} & \frac{1}{4} & 0 & -\frac{3}{64} & \frac{1}{32} & \frac{1}{64} & -\frac{9}{64} & \frac{3}{32} & \frac{3}{64} \\ 0 & 0 & -\frac{1}{4} & 0 & 0 & \frac{1}{4} & -\frac{1}{64} & -\frac{1}{32} & \frac{3}{64} & -\frac{3}{64} & -\frac{3}{32} & \frac{9}{64} \\ \frac{9}{64} & \frac{3}{64} & -\frac{3}{32} & -\frac{1}{32} & -\frac{3}{64} & -\frac{1}{64} & \frac{1}{4} & 0 & 0 & 0 & -\frac{1}{4} & 0 \\ -\frac{3}{32} & \frac{3}{32} & \frac{1}{16} & -\frac{1}{16} & \frac{1}{32} & -\frac{1}{32} & 0 & \frac{3}{2} & 0 & -\frac{1}{4} & -1 & -\frac{1}{4} \\ -\frac{3}{64} & -\frac{9}{64} & \frac{1}{32} & \frac{3}{32} & \frac{1}{64} & \frac{3}{64} & 0 & 0 & \frac{1}{4} & 0 & -\frac{1}{4} & 0 \\ \frac{3}{64} & \frac{1}{64} & \frac{3}{32} & \frac{1}{32} & -\frac{9}{64} & -\frac{3}{64} & 0 & -\frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 \\ -\frac{1}{32} & \frac{1}{32} & -\frac{1}{16} & \frac{1}{16} & \frac{3}{32} & -\frac{3}{32} & -\frac{1}{4} & -1 & -\frac{1}{4} & 0 & \frac{3}{2} & 0 \\ -\frac{1}{64} & -\frac{3}{64} & -\frac{1}{32} & -\frac{3}{32} & \frac{3}{64} & \frac{9}{64} & 0 & -\frac{1}{4} & 0 & 0 & 0 & \frac{1}{4} \end{pmatrix} & -\mu h \begin{pmatrix} -\frac{1}{8} \\ \frac{1}{8} \\ -\frac{3}{4} \\ \frac{3}{4} \\ -\frac{1}{8} \\ \frac{1}{8} \\ -\frac{1}{8} \\ -\frac{3}{4} \\ -\frac{1}{8} \\ \frac{1}{8} \\ \frac{3}{4} \\ \frac{1}{8} \end{pmatrix} \\ -\mu h \begin{pmatrix} -\frac{1}{8} & \frac{1}{8} & -\frac{3}{4} & \frac{3}{4} & -\frac{1}{8} & \frac{1}{8} & -\frac{1}{8} & -\frac{3}{4} & -\frac{1}{8} & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \end{pmatrix} & -\frac{\mu^2}{\lambda} h^2 \end{pmatrix}. \quad (7.25)$$

The global stiffness matrix generated from $\mathbf{A}^{k_p}$ has a stencil shown in Figure 7.3.

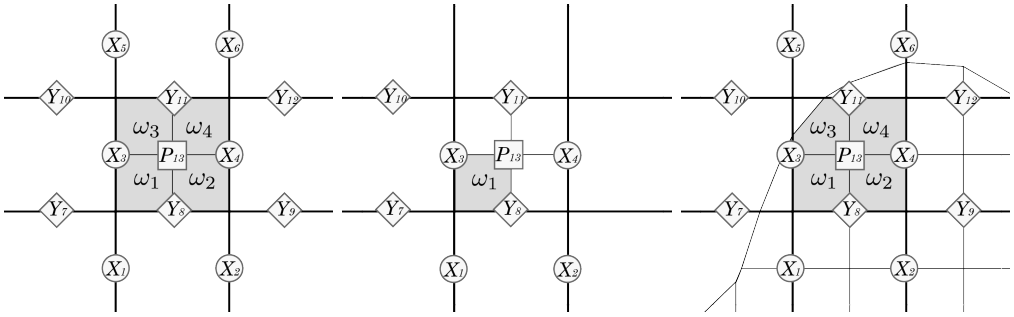


Figure 7.2: Left: one interior pressure cell and the variables corresponding to the 13 degrees of freedom of the elemental stiffness matrix by taking integral over the pressure cell; right: four integral subcells of the pressure cell and the variables that an integral over subcell ω_1 contributes to.

However for boundary cells where $T^p_{k_p} \cap \Omega \neq T^p_{k_p}$, we have to perform the integrations involved in each of $\mathbf{A}^{k_p}_{\omega_i}$ carefully taking into account the boundary

geometry. We discuss this in the next section. First however, we summarize the process of constructing the global stiffness matrix $\mathbf{A}$ from each of 13×13 element stiffness matrices $\mathbf{A}^{k_p}$.

We construct the global stiffness matrix from elemental stiffness matrices $\mathbf{A}^{k_p}$ explained in Algorithm 3, and the derived global stiffness matrix generated from $\mathbf{A}^{k_p}$ has a stencil shown in Figure 7.3

Algorithm 3 Construction of global stiffness matrix $\mathbf{A}$ from elemental $\mathbf{A}^{k_p}$

$\mathbf{A} \leftarrow \mathbf{0}$
for $k_p = 1$ to N_p **do**
  if $T^p_{k_p} \cap \Omega = T^p_{k_p}$ **then** ▷ Interior cell: use precomputed $\mathbf{A}^{k_p}$
    Use $\mathbf{A}^{k_p}$ from equation (7.25)
  else ▷ Boundary cell: compute $\mathbf{A}^{k_p}$ from $\mathbf{A}^{k_p}_{\omega_i}$
    Perform integration over each subquadrant ω_i to compute $\mathbf{A}^{k_p}_{\omega_i}$
    $\mathbf{A}^{k_p} = \mathbf{A}^{k_p}_{\omega_1} + \mathbf{A}^{k_p}_{\omega_2} + \mathbf{A}^{k_p}_{\omega_3} + \mathbf{A}^{k_p}_{\omega_4}$
  end if
  for $i^p = 1$ to 13 **do**
    $i = \text{mesh}(k_p, i^p)$ ▷ The mesh maps the 13 degrees of freedom involved in $\mathbf{A}^{k_p}$ to their position in a global array
    for $j^p = 1$ to 13 **do**
      $j = \text{mesh}(k_p, j^p)$
      $A_{ij} += A^{k_p}_{i^p j^p}$
    end for
  end for
end for

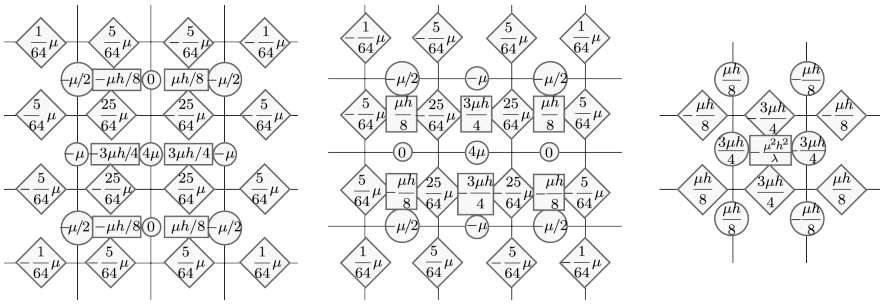


Figure 7.3: Global stiffness matrix stencils centered at an interior x variable(left), y variable(middle) and p variable(right).

7.2.3 Discrete geometric representation and cut cell integration

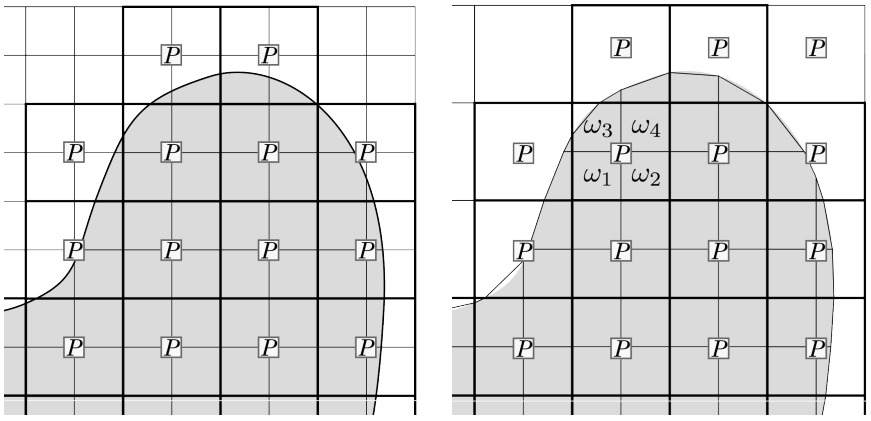


Figure 7.4: A zoom-in view of Figure 7.1(a). A levelset function is sampled on a doubly refined grid(left); a segmented curve $\partial\Omega_h$ is generated to approximate the boundary of the geometric domain(right).

We discretize the domain Ω by embedding it in a regular grid. Specifically, we use a signed distance level set function defined over a doubly refined subgrid:

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

This doubly refined subgrid is thus a superset of all grid nodes in the x, y and p grids. The signed distance values at the nodes of the doubly refined grid $\mathcal{G}^{\phi}$ 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 then approximated within the region enclosed by $\partial\Omega^h$ (see Figure 7.1). 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 7.1.

Thus we can think of our discrete domain as a union of doubly refined uncut quadralaterals on the interior and cut, polygonal regions contained in doubly refined quadralaterals on the boundary (see Figure 7.4).

This partitioning of the domain into doubly refined quadralaterals naturally supports our integration conventions needed for the matrices $\mathbf{A}_{\omega_i}^{k_p}$ discussed in the previous section. The integrals needed for these matrices are evaluated trivially when ω_i is not cut. However when ω_i is cut by the boundary, we can still perform the integrations analytically following ideas from the recent cut cell approach in Bedrossian et al. (2010). The integrands of each term in the matrices $\mathbf{A}_{\omega_i}^{k_p}$ are polynomials in x and y of degree 2. That is, each integral is of the form:

$$\int_{\omega_i \cap \Omega} ax^2 + bxy + cy^2 + dx + ey + f d\boldsymbol{x}.$$

Our level set based representation of the geometry means that the domain of integration $\omega_i \cap \Omega$ is polygonal. In other words, we need to evaluate a second order polynomial over a polygonal domain. This task can be done trivially by noting that

$$\begin{aligned} &\int_{\omega_i \cap \Omega} ax^2 + bxy + cy^2 + dx + ey + f d\boldsymbol{x} \\ &= \int_{\omega_i \cap \Omega} \nabla \cdot \begin{pmatrix} \frac{ax^3}{3} + \frac{bx^2 y}{2} + cxy^2 + \frac{dx^2}{2} + exy + fx \\ 0 \end{pmatrix} d\boldsymbol{x}. \end{aligned}$$

That is, because $\omega_i \cap \Omega$ is polygonal and our integrand can be expressed in terms of the divergence of a cubic function, application of the divergence theorem yields

the easily evaluated forumula:

$$\int_{\omega_i \cap \Omega} \nabla \cdot \begin{pmatrix} \frac{ax^3}{3} + \frac{bx^2y}{2} + cxy^2 + \frac{dx^2}{2} + exy + fx \\ 0 \end{pmatrix} d\boldsymbol{x}$$
$$= \sum_{s=1}^{N_{\partial(\omega_i \cap \Omega)}} n_1(s) \int_{\partial(\omega_i \cap \Omega)_s} \left(\hat{a}(s)t^3 + \hat{b}(s)t^2 + \hat{c}(s)t + \hat{d}(s) \right) dt.$$

Here, the $N_{\partial(\omega_i \cap \Omega)}$ is the number of line segments in the boundary of the polygonal domain, $\partial(\omega_i \cap \Omega)_s$ is the s-th segment in the polygonal boundary, $n_1(s)$ is the x component of the outward normal to the s-th segment and $\hat{a}(s), \hat{b}(s), \hat{c}(s), \hat{d}(s)$ are the cubic coefficients arising in the boundary integrals over each segment. Again, each term in the sum can be evaluated analytically. This careful treatment of the integrals arising in each $\mathbf{A}_{\omega_i}^{k_p}$ is the key to obtaining second order accuracy in L^∞.

7.3 Dirichlet boundary conditions

We have thus far assumed that our solution satisfies the Dirichlet boundary conditions and that our test functions vanish on the Dirichlet boundary. However, because we use a regular 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 can enforce these conditions weakly using

the following variational problem:

$$
\begin{aligned}
&\text{find } (\boldsymbol{u}, p) \in H^1(\Omega) \times H^1(\Omega) \times L^2(\Omega), \text{ such that} \\
&\int_\Omega 2\mu \left(\frac{\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T}{2} \right) : \left(\frac{\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T}{2} \right) + \mu p (\nabla \cdot \boldsymbol{v}) \, d\boldsymbol{x} \\
&\quad = -\int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, d\boldsymbol{x} + \int_{\partial\Omega} \boldsymbol{g} \cdot \boldsymbol{v} \, ds \quad \forall \boldsymbol{v} \in H^1_{0,\Gamma_d}(\Omega) \times H^1_{0,\Gamma_d}(\Omega) \qquad (7.26) \\
&\int_\Omega \left(-\mu q \nabla \cdot \boldsymbol{u} - \frac{\mu^2}{\lambda} pq \right) d\boldsymbol{x} = 0 \quad \forall q \in L^2(\Omega) \qquad (7.27) \\
&\int_{\Gamma_d} \boldsymbol{u} \cdot \boldsymbol{w} \, ds = \int_{\Gamma_d} \boldsymbol{u}_0 \cdot \boldsymbol{w} \, ds \quad \forall \boldsymbol{w} \in (H^{-1/2}(\Gamma_d))^2. \qquad (7.28)
\end{aligned}
$$

Here, we introduce the Dirichlet condition as a constraint. Specifically, we require that the L^2 inner product of the solution and an arbitrary function $\boldsymbol{w} \in (H^{-1/2}(\Gamma_d))^2$ is the same as the inner product of the Dirichlet data $\boldsymbol{u}_0$ with $\boldsymbol{w}$. This makes the problem a constrained minimization.

7.3.1 Discretizing the Dirichlet problem

In order to discretize the Dirichlet condition in the weak formulation, we approximate $(H^{-1/2}(\Gamma_d))^2$ using a subspace $\Lambda^h_x \times \Lambda^h_y = P_0(\mathcal{T}^x \cap \Gamma^h_d) \times P_0(\mathcal{T}^y \cap \Gamma^h_d)$, which is composed of piecewise constant functions over x and y component grid cells that intersect the Dirichlet boundary. Here we use Γ^h_d to denote the portion of $\partial\Omega^h$ over which the Dirichlet constraint is 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 $\boldsymbol{w}_{T^i} = \chi_{T^i}(\boldsymbol{x})\boldsymbol{e}_i$ as the basis functions for $\Lambda^h_x \times \Lambda^h_y = P_0(\mathcal{T}^x \cap \Gamma^h_d) \times P_0(\mathcal{T}^y \cap \Gamma^h_d)$. Here, $\chi_{T^i}(\boldsymbol{x})$ is the characteristic function

of the cell T^i:

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

Note that we have one basis function per boundary x or y cell. If we use N_{xd} and N_{yd} 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 $N_{xd} + N_{yd}$.

With this approximation, the Dirichlet boundary condition constraint can be expressed as a linear system $\mathbf{B}^h \boldsymbol{u}^h = \boldsymbol{u}_0^h$ ($\mathbf{B}^h \in \mathbb{R}^{(N_{xd}+N_{yd})\times(N_x+N_y)}$, $\boldsymbol{u}_0^h \in \mathbb{R}^{(N_{xd}+N_{yd})}$) where each equation enforces an integral constraint over the intersection of the discrete boundary Γ_d^h with some x or y boundary cell T^i:

$$\sum_{k_i \in I_i} u_{ik_i} \int_{T^i \cap \Gamma_d^h} N_{ik_i}(\boldsymbol{x}) \, ds = \int_{T^i \cap \Gamma_d^h} u_{0i}(\boldsymbol{x}) \, ds,$$

where $\boldsymbol{u}_0 = (u_{01}, u_{02})$. In practice, we evaluate the integral for a given boundary cell T^i over the portion of the Dirichlet boundary curve $T^i \cap \Gamma_d^h$ from the four subquadraterals of T^i arising from the doubly refined grid (as discussed in section). This is simple because in each of these subquadralaterals Γ_d^h is just a single line segment. We use the following approximation for the right hand side terms in the constraint system:

$$\int_{T^i \cap \Gamma_d^h} u_{0i}(\boldsymbol{x}) \, ds = \sum_{j=1}^{4} \int_{\omega_j^i \cap \Gamma_d^h} u_{0i}(\boldsymbol{x}) \, ds \approx \sum_{j=1}^{4} u_{0i}(c(\omega_j^i \cap \Gamma_d^h)) \int_{\omega_j^i \cap \Gamma_d^h} 1 \, ds,$$

where ω_j^i is one of the four subquadralaterals of the cell T^i and $c(\omega_j^i \cap \Gamma_d^h))$ is the midpoint of the segment $\omega_j^i \cap \Gamma_d^h$. We use the same treatment for the entries in

the matrix on the left hand side:

$$\int_{T^i \cap \Gamma_d^h} N_{ik_i}(\boldsymbol{x})\, ds = \sum_{j=1}^{4} \int_{\omega_j^i \cap \Gamma_d^h} N_{ik_i}(\boldsymbol{x})\, ds.$$

We note that the integrand here is simply an x or y bilinear interpolating function. Therefore, the four terms in the sum can be evaluated analytically because they are simply quadratics in any linear parameterization of a given boundary segment $\omega_j^i \cap \Gamma_d^h$.

The discrete constrained minimization problem can be solved using a Lagrange multiplier method resulting in the following KKT system:

$$\begin{pmatrix} \mathbf{L}_u^h & \mathbf{G}^{hT} & \mathbf{B}^{hT} \\ \mathbf{G}^h & \mathbf{D}_p^h & \mathbf{0} \\ \mathbf{B}^h & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^h \\ \boldsymbol{p}^h \\ \boldsymbol{\lambda}^h \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}^h \\ \mathbf{0} \\ \boldsymbol{u}_0^h \end{pmatrix}. \tag{7.29}$$

There is one Lagrange multiplier degree of freedom per Dirichlet constraint. In other words, $\boldsymbol{\lambda}^h$ is in $\mathbb{R}^{(N_{xd}+N_{yd})}$. When we consider boundary equations in the sections that follow, we temporarily eliminate pressure variables $\boldsymbol{p}^h$ with the following substitution $\mathbf{L}^h = \mathbf{L}_u^h - \mathbf{G}^{hT}(\mathbf{D}_p^h)^{-1}\mathbf{G}^h$:

$$\begin{pmatrix} \mathbf{L}^h & \mathbf{B}^{hT} \\ \mathbf{B}^h & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^h \\ \boldsymbol{\lambda}^h \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}^h \\ \boldsymbol{u}_0^h \end{pmatrix}. \tag{7.30}$$

This system is extremely ill-conditioned for nearly incompressible materials, however it will simplify the exposition of the forthcoming discussion of Dirichlet boundary condition treatment. Furthermore, when performing equation relaxation in our multigrid solver, we temporarily perform this elimination when treating equations near the boundary of the domain. Before discussing our geometric

multigrid solution approach for these systems of equations, we would like to first discuss some important aspects of the Dirichlet system. Our treatment of the Dirichlet condition is somewhat nonstandard and we list here a few important details related to the constraint matrix $\mathbf{B}^h$.

1. $\mathbf{B}^h$ consists of two decoupled blocks. There is one block for the x boundary equations and one for the y equations. The only non-zero columns of $\mathbf{B}^h$ are associated with nodes that are incident on an x or y boundary cell. Therefore, for sufficiently interior degrees of freedom, the KKT system is exactly the same as (7.24) or (5.6). That is, although the constraint matrix is in $\mathbb{R}^{(N_{xd}+N_{yd})\times(N_x+N_y)}$, it really only acts on a small subset of the N_x+N_y displacement degrees of freedom.

2. $\mathbf{B}^h \in \mathbb{R}^{(N_{xd}+N_{yd})\times(N_x+N_y)}$, where $(N_{xd}+N_{yd}) < (N_x+N_y)$. However, it can be shown that $\mathbf{B}^h$ has full row rank. We refer the reader to the work Bedrossian et al. (2010) for a more detailed discussion of why this is so.

3. Our numerical linear algebra approach to the problem is based on the construction of a matrix $\mathbf{Z}^h \in \mathbb{R}^{(N_x+N_y)\times((N_x+N_y)-(N_{xd}+N_{yd}))}$ (of full column rank) whose columns span the kernel space of $\mathbf{B}^h$. Since $\mathbf{B}^h$ has full row rank, there exists column permutation $\mathbf{Q}$, such that $\mathbf{B}^h\mathbf{Q} = [\mathbf{B}_m|\mathbf{B}_{n-m}]$, where $\mathbf{B}_m \in \mathbb{R}^{(N_{xd}+N_{yd})\times(N_{xd}+N_{yd})}$ is non-singular and

$$\mathbf{B}^h = \begin{pmatrix} \mathbf{B}^x & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^y \end{pmatrix},$$

where $\mathbf{B}^x \in \mathbb{R}^{N_{xd}\times N_x}$ and $\mathbf{B}^y \in \mathbb{R}^{N_{yd}\times N_y}$. However, it can be shown that $\mathbf{B}^h$ has full row rank. We refer the reader to the work Bedrossian et al. (2010) for a more detailed discussion of why this is so.

4. Our numerical linear algebra approach to the problem is based on the construction of a matrix $\mathbf{Z}^h \in \mathbb{R}^{(N_x+N_y)\times((N_x+N_y)-(N_{xd}+N_{yd}))}$ (of full column rank) whose columns span the kernel space of $\mathbf{B}^h$. Since $\mathbf{B}^h$ has full row rank, there exists column permutation $\mathcal{P}$, such that $\mathbf{B}^h\mathcal{P} = [\mathbf{B}_m|\mathbf{B}_{n-m}]$, where $\mathbf{B}_m \in \mathbb{R}^{(N_{xd}+N_{yd})\times(N_{xd}+N_{yd})}$ is non-singular and $\mathbf{B}_{m-n} \in \mathbb{R}^{(N_{xd}+N_{yd})\times((N_x+N_y)-(N_{xd}+N_{yd}))}$. With this permutation, we can construct a so called fundamental basis for the null-space of $\mathbf{B}^h$:

$$\mathbf{Z}^h = \begin{bmatrix} -\mathbf{B}_m^{-1}\mathbf{B}_{n-m} \\ \mathbf{I} \end{bmatrix}. \tag{7.31}$$

5. The vector

$$\boldsymbol{c}^h = \begin{bmatrix} \mathbf{B}_m^{-1}\boldsymbol{u}_0^h \\ 0 \end{bmatrix} \tag{7.32}$$

satisfies $\mathbf{B}^h\mathcal{P}\boldsymbol{c}^h = \boldsymbol{u}_0^h$. Therefore, all solutions $\boldsymbol{u}^h$ can be expressed as $\boldsymbol{u}^h = \mathcal{P}(\boldsymbol{c}^h + \mathbf{Z}^h\boldsymbol{v}^h)$ with $\mathbf{v}^h \in \mathbb{R}^{((N_x+N_y)-(N_{xd}+N_{yd}))}$.

6. Our construction of a null-space for the Dirichlet constraint allows us to eliminate the Lagrange multipliers. Substituting $\boldsymbol{u}^h = \mathcal{P}(\boldsymbol{c}^h + \mathbf{Z}^h\boldsymbol{v}^h)$ in (7.30), we have

$$\mathbf{L}^h\mathcal{P}(\boldsymbol{c}^h + \mathbf{Z}^h\boldsymbol{v}^h) + \mathbf{B}^{h^T}\boldsymbol{\lambda}^h = \boldsymbol{f}^h$$

Left multiplying this equation with $(\mathcal{P}\mathbf{Z}^h)^T$, and applying the property $\mathbf{B}^h\mathcal{P}\mathbf{Z}^h = 0$, we have

$$(\mathcal{P}\mathbf{Z}^h)^T\mathbf{L}^h\mathcal{P}\mathbf{Z}^h\boldsymbol{v}^h + (\mathbf{B}^h\mathcal{P}\mathbf{Z}^h)^T\boldsymbol{\lambda}^h = (\mathcal{P}\mathbf{Z}^h)^T(\boldsymbol{f}^h - \mathbf{L}^h\mathcal{P}\boldsymbol{c}^h)$$

$$(\mathbf{Z}^{h^T}\mathcal{P}^T\mathbf{L}^h\mathcal{P}\mathbf{Z}^h)\boldsymbol{v}^h = (\mathcal{P}\mathbf{Z}^h)^T(\boldsymbol{f}^h - \mathbf{L}^h\mathcal{P}\boldsymbol{c}^h). \tag{7.33}$$

That is, if we can solve $\boldsymbol{v}^h$ from the reduced system (7.33), then the KKT system solution can be reconstructed with $\boldsymbol{u}^h = \mathcal{P}(\boldsymbol{c}^h + \mathbf{Z}^h\boldsymbol{v}^h)$. Without loss of generality, we assume the variables to be reordered such that $\mathcal{P} = \mathcal{I}$. We will make use of this property in the smoother for our geometric multigrid method.

7.3.2 Constructing the null-space for the Dirichlet constraints

Our treatment of the Dirichlet conditions is based on our ability to construct a null-space $\mathbf{Z}^h$ satisfying $\mathbf{B}^h\mathbf{Z}^h = 0$ and a special solution $\boldsymbol{c}^h$ satisfying $\mathbf{B}^h\boldsymbol{c}^h = \boldsymbol{u}_0^h$. The main issue we will discuss now is how to a find fundamental basis $\mathbf{Z}^h$ that produces a numerically well-conditioned system of equations. This topic was originally discussed in the work of Bedrossian et al Bedrossian et al. (2010). However, we found that in our case of nearly incompressible materials, an even more aggressive approach is needed. Bedrossian et al first suggested an ordering for the boundary integral equations and incident boundary nodes that led to a readily inverted upper triangular $\mathbf{B}_m$. However, they showed that although this construction was straightforward, the conditioning of this $\mathbf{B}_m$ deteriorated exponentially in the discretization resolution and was thus not practical. They then showed that it is possible to derive a diagonal $\mathbf{B}_m$ with a slight modification to the definition of the constraints. Specifically, they showed that an aggregation scheme where cells of a "double-wide" grid were used to define the extent of the line integral constraints led to a diagonal $\mathbf{B}_m$. This was done by choosing the center node of the 9 nodes incident on the four original cells in the "double-wide" cell as the representative node in the permutation of columns in $\mathbf{B}^h$. In other

words, the center node of the four aggregated cells was given the same index as the row associated with the constraint over those cells. This aggregation of cells is equivalent to replacing a collection of rows in the original, "single-wide" $\mathbf{B}^h$ with the sum of the collection of rows. The choice of which rows to sum together is determined by the "double-wide" cell they belong to. Since no aggregated rows have a non-zero entry associated with the center node of any other aggregate, the $\mathbf{B}_m$ is diagonal and thus $\mathbf{Z}^h$ is trivially constructed. This aggregation can equivalently be seen as using a subspace $\Lambda_x^{2h} \times \Lambda_y^{2h} \cap H^{-1/2}(\Gamma_d)$ that is based on a coarsened grid. Remarkably, this process does not affect the L^∞ convergence behavior of the scheme. Unfortunately, while the reduced system in Bedrossian et al. (2010) had a satisfactory condition number for the Poisson equation with an incomplete Choleksy preconditioned conjugate gradient solver, we found that this was not the case for nearly incompressible linear elasticity and geometric multigrid. Specifically, the conditioning of the equations at the boundary was poor enough to make the treatment of boundary regions prohibitively costly when performing the smoothing operations used in geometric multigrid. Figure 7.5 illustrated the difference between boundary integral enforced over single cells and over a cell aggregation. The gray domain illustrates the domain of active primal cells. The blue grid is the y variable node grid. The dashed red line indicates integral region of a boundary equation, which has nonzero entries on variables located at the blue circles both hollow and solid. And the solid circle in the right image is the representative node of the illustrated cell aggregation.

We propose a new boundary integral constraint aggregation that allows us to generate a diagonal $\mathbf{B}_m$ and a better-conditioned reduced system. We note that the elements of the original $\mathbf{B}^h$ are scaled by a segment length; therefore, a very small segment (associated with a node that is far from the boundary) will generate very small diagonal elements for a $\mathbf{B}_m$ arising from aggregation. We found this

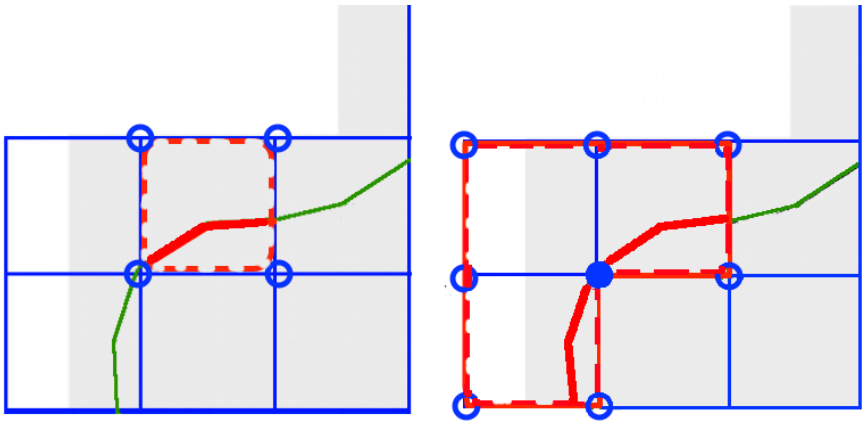


Figure 7.5: Boundary integration cells and aggregations. Left: each boundary equation enforces the boundary integral within a single cell; right: each boundary equation enforces the boundary integral within an aggregation of cells.

to be a source of the suboptimal conditioning. We propose a modification to the aggregation process that is designed to provide larger diagonal entries in subsequent $\mathbf{B}_m$. We first define the weight of each node to be the sum of the column in the original, "single-wide" $\mathbf{B}^h$. This weight is the diagonal entry in $\mathbf{B}_m$ that would arise from an aggregation of the four cells centered around the node. To increase the diagonal entries in the final $\mathbf{B}_m$, we simply select four cell aggregates such that the associated representative nodes will be chosen in descending order of the total weight of each node. After we choose a node to define a four cell aggregate, the nine nodes incident on the four cells are eliminated from consideration. This process of prioritizing the aggregation rather than inheriting it from the coarse grid has a drawback. There may be some rows that are never added to an aggregate region. For these rows, it may be impossible to choose a representative node. We resolve this by aggregating such a row into a spatially adjacent aggregate. In practice, we also found that limiting representative nodes in the aggregation to ghost nodes (i.e. those geometrically outside the domain Ω) gave even better conditioning. We briefly summarize this process in Algorithm

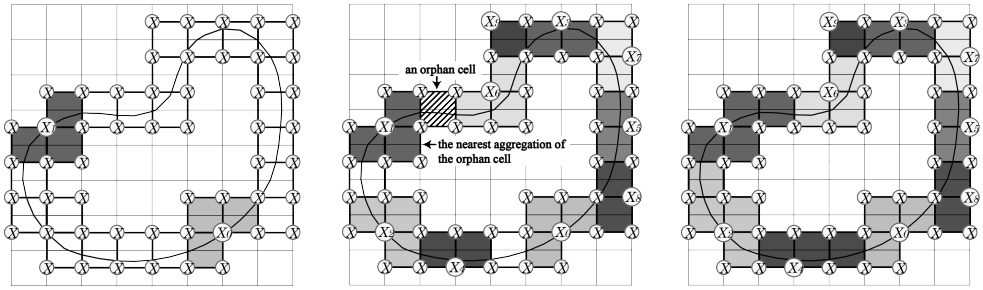
4.

Algorithm 4 Aggregation Selection

procedure AGGREGATIONSELECTION($\mathbf{B}^x$, $\mathbf{B}^y$)
  for v **in** $\{1, 2\}$ **do**
    aggregation cells list $\mathcal{A}ggr^v = \{\}$
    aggregation representative list $\mathcal{R}ep^v = \{\}$
    set all active node^v variables and all integral cells^v active
    row weight sum $w_i \leftarrow \sum_j \mathbf{B}^v_{ij}$
    Sort w in a decreasing order
    for w_j **in** w **do**
      if node^v_j is active **then**
        $\mathcal{R}ep^v \leftarrow \mathcal{R}ep^v \cup \{j\}$
        $\mathcal{A}ggr^v \leftarrow \mathcal{A}ggr^v \cup \{c$ for all cell$_c$ adjacent to node$^v_j\}$
        deactivate all cells adjacent to node^v_j
        deactivate all nodes adjacent to node^v_j
      end if
    end for ▷ Now every ghost node belongs to at least one aggregation
    for all boundary cell^v_i **do**
      if cell$_i$ is inactive **then**
        Find the closest aggregation of cell^v_i $\mathcal{A}ggr^v_k$
        $\mathcal{A}ggr^v_k \leftarrow \mathcal{A}ggr^v_k \cup \{i\}$
      end if
    end for ▷ Pickup orphans
  end for
end procedure

7.4 Multigrid

We develop an efficient multigrid solver for the discrete systems produced by our method. Our method is purely geometric, and based on the Multigrid Correction Scheme (see Algorithm 5). The framework admits a simple implementation, however special care is needed to retain near-textbook multigrid convergence rates, especially in the presence of highly irregular domains or nearly incompressible materials. The sections that follow will detail the key components of our multi-



(a) (b) (c)

Figure 7.6: Cell aggregations. a) x component boundary cells and the first two representative nodes and the incident $y-$ cells of each representative node; b) all representative nodes for y component cells and their incident cells, orphan cells will be attached to their nearest neighbor cells; c) and d) final cell aggregations together with their representative nodes for x and y grids.

grid solver: a hierarchy of discretizations, a smoothing procedure, and appopriate transfer operators (i.e. restriction and prolongation) between levels of the hierarchy. Although our design decisions include certain common practices, these components have been significantly customized to fit the needs of the specific discretization being followed, and facilitate both convergence and computational efficiency even near the incompressible limit.

Algorithm 5 Multigrid defect correction

procedure V-Cycle($\hat{\mathbf{L}}^h$,$\hat{\boldsymbol{u}}^h$,$\hat{\boldsymbol{f}}^h$)
  if problem at low resolution and easy to solve **then**
    $\hat{\boldsymbol{u}}^h \leftarrow (\hat{\mathbf{L}}^h)^{-1}\hat{\boldsymbol{f}}^h$ and return;
  end if
  PreRelaxation($\hat{\mathbf{L}}^h$,$\hat{\boldsymbol{u}}^h$,$\hat{\boldsymbol{f}}^h$)
  Restriction: $\hat{\boldsymbol{f}}^{2h} \leftarrow \mathbf{R}(\hat{\boldsymbol{f}}^h - \hat{\mathbf{L}}^h\hat{\boldsymbol{u}}^h)$
  V-Cycle($\hat{\mathbf{L}}^{2h}$,$\hat{\boldsymbol{u}}^{2h}$,$\hat{\boldsymbol{f}}^{2h}$)
  Prolongation: $\hat{\boldsymbol{u}}^h \leftarrow \hat{\boldsymbol{u}}^h + \mathbf{P}\hat{\boldsymbol{u}}^{2h}$
  PostRelaxation($\hat{\mathbf{L}}^h$,$\hat{\boldsymbol{u}}^h$,$\hat{\boldsymbol{f}}^h$)
end procedure

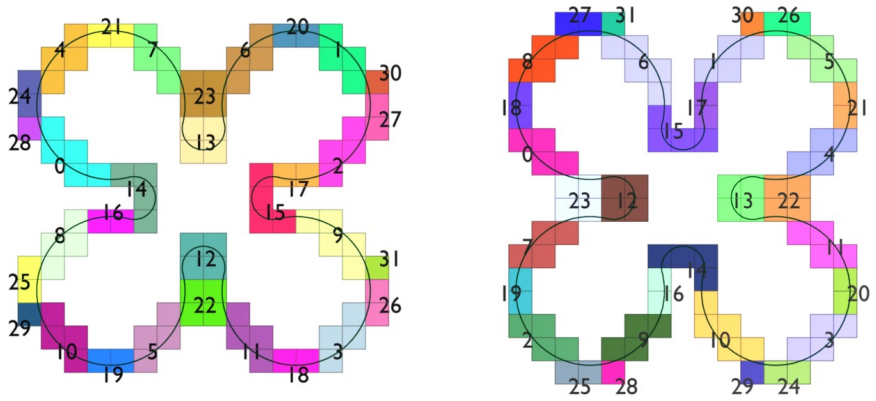


Figure 7.7: Cell aggregations of an example domain for both x(left) and y(right) variables.

7.4.1 Discretization hierarchy

We consider a hierarchy of resolutions, each corresponding to a discretization on a progressively larger grid size. In particular, we employ a grid step of h on the finest level of the hierarchy (numbered as level zero), followed by discretizations with grid step sizes of $2h, 4h, \ldots, 2^L h$, for a total of $L+1$ hierarchy levels. In detail, the hierarchy is constructed as follows:

- At every level of the hierarchy, say the l-th one, we define the background grids $\mathcal{G}^x_{2^l h}, \mathcal{G}^y_{2^l h}, \mathcal{G}^p_{2^l h}$ corresponding to the x-, y-, and p-variables respectively.
- A level set function is computed over the respective doubly-refined subgrids $\mathcal{G}^\phi_h, \mathcal{G}^\phi_{2h}, \mathcal{G}^\phi_{4h}, \ldots$ for each level. Obviously, coarser levels may fail to resolve certain high-frequency features of the domain geometry, leading to possible discrepancies between the discrete systems at various levels, which will be further addressed in our discussion of the smoother and transfer operators.
- Using the level set values associated with a given grid, we generate the discrete domains $\mathcal{T}^x_{2^l h}, \mathcal{T}^y_{2^l h}, \mathcal{T}^p_{2^l h}$, and allocate the unknown arrays $\boldsymbol{u}^{2^l h}$ and

$\boldsymbol{p}^{2^l h}$ as well as the right-hand sides $\boldsymbol{f}^{2^l h}$ and $\boldsymbol{f}_p^{2^l h}$ of the respective equations. Note that, although at the finest level of the hierarchy we have used $\boldsymbol{f}_p^h = 0$ by virtue of our discretization, the right hand side $\boldsymbol{f}_p^{2^l h}$ for coarser levels ($l \geq 1$) will generally be nonzero in the Multigrid Correction Scheme (see Algorithm 5).

Multigrid method is based on a hierarchical discretization. On each level l, we first rasterize our domain into regular cells with $h = 2^{-l}$, and define a staggered discretization on $\mathcal{G}_h^x$, $\mathcal{G}_h^y$ and $\mathcal{G}_h^p$. Based on the levelset function values sampled on the grid node points of a doubly refined grid, a geometric discretization of the domain is defined (see Figure 7.1 top right). The active degrees of freedom and the corresponding right hand side values are thus defined on the nodes of $x-$, $y-$ and $p-$ cells that overlap with the discretized domain Ω^h, i.e. the nodes of the dashed line cells in Figure 7.1 bottom and the nodes of the thicker line cells in Figure 7.1 top left.

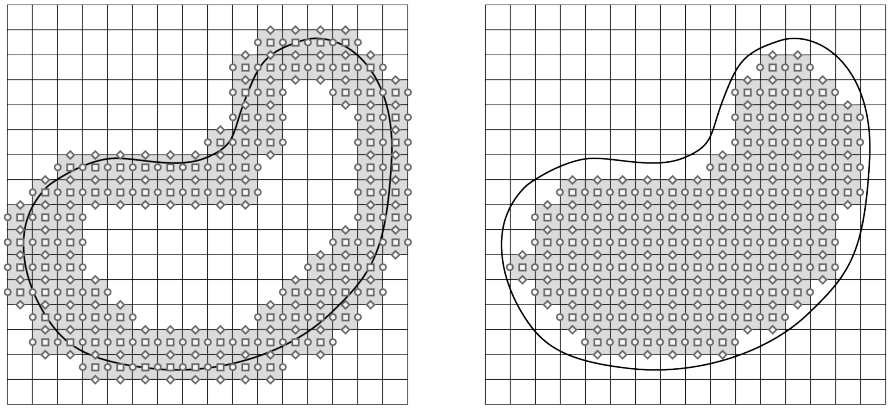
When there is a Dirichlet boundary condition, a constraint matrix $\mathbf{B}^h$ is defined for each level, enforcing the integral of x or y displacement components along the discretized domain boundary $\partial\Omega^h$ of the current level and within each cell aggregation (each cell group with the same color in Figure 7.6) precomputed on the same level, to be the same as that of the Dirichlet values for the corresponding displacement component. Each of such constraints (or cell aggregations) corresponds to one Lagrange multipliers. In the fundamental basis method, we eliminate these multipliers by solving for the fundamental basis coefficients $\boldsymbol{v}^h$ in $\boldsymbol{c}^h + \mathbf{Z}^h \boldsymbol{v}^h$. By definition of c^h (7.32) and $\mathbf{Z}^h$ (7.31), there is a one-to-one mapping between v^h components and active x and y degrees of freedom that are not aggregation representatives. Thus, the reduced system 7.33 is defined on these degrees of freedom only. Due to the fact that the reconstructed $u^h = c^h + \mathbf{Z}^h \boldsymbol{v}^h$

satisfy the constraints automatically, we do not need to restrict any residual for the constraint system, i.e. on coarser level, the Dirichlet boundary constraint values are always zero. Also, $\boldsymbol{\lambda}^h$ do not need to be solved, therefore, we do not need to record their values, nor prolongate their corrections. When we restrict the residuals of the governing equation, i.e. $\boldsymbol{r}^h = \boldsymbol{f}^h - \mathbf{L}^h\boldsymbol{u}^h - \mathbf{B^h}^T\boldsymbol{\lambda}^h$, we restrict zero for all equations that will need a $\boldsymbol{\lambda}^h$ value. In another word, we restrict zero residuals from equations involving boundary nodes, i.e. the nodes in Figure 7.6 cells. Thus, a single level discretization is defined for all levels with powers of 2 resolutions.

7.4.2 Relaxation

The interior equations are uniform and have the same properties, while near the boundary, the equations have very different stencils. We distinguish the interior equations and boundary equations, and apply different relaxations on them. First, we recognize cells that are outside any 5×5 box centered at certain exterior cells, illustrated in Figure 7.8 right. For equations that are incident to these cells, a distributive interior relaxation is applied. Then, we recognize cells that are active cell and within a 9×9 box centered at some exterior cells, illustrated in Figure 7.8 left. We apply a different boundary relaxation within this boundary band. In each single level relaxation, we first sweep over the boundary band, and apply a few iterations of boundary relaxations, then apply one iteration of interior relaxation followed by another few iterations of boundary relaxations.

The efficiency of a multigrid method is closely related to the smoothing efficiency of a single level relaxation. With the Poisson's equation, simple Jacobi or Gauss-Seidel will typically suffice as an efficient smoother. These techniques



(a) Boundary band pressure cells and boundary variables

(b) Distributive pressure cells and variables relaxed using distributive relaxation

Figure 7.8: Boundary band and distributive region.

efficiently reduce the high-frequency component of the error and make it possible for a coarse grid to provide a meaningful correction to a finer grid. This property is fundamentally important for the efficiency of the geometrically hierarchical approach to solving the equations. Unfortunately, the equations of nearly incompressible linear elasticity with augmented pressure require more care than the comparably simplistic discrete Poisson equation. Although our system is not symmetric positive definite, we can modify the equations to a more convenient form as in Zhu et al. (2010) to design a proper geometric multigrid smoother. We confirm that a change of variables leads to approximately block triangulate the discrete system with each diagonal block being a symmetric semi-definite discretization of Poisson. Our smoother is then constructed to be an emulation of the Gauss-Seidel relaxation applied on each block.

7.4.3 Approximated distributive relaxation

We follow the idea in Zhu et al. (2010) and develop a distributive relaxation. First, we apply a change of variable:

$$\begin{pmatrix} \boldsymbol{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{I} & -\nabla \\ \nabla^T & -2\Delta \end{pmatrix} \begin{pmatrix} \boldsymbol{v} \\ q \end{pmatrix} \qquad \text{or} \qquad \hat{\boldsymbol{u}} = \hat{\mathbf{M}}\hat{\boldsymbol{v}} \tag{7.34}$$

and substitute into (7.7) to achieve a new system

$$\begin{pmatrix} \mu\Delta\mathbf{I} & 0 \\ \mu(1+\frac{\mu}{\lambda})\nabla^T & -\mu(1+\frac{2\mu}{\lambda})\Delta \end{pmatrix} \begin{pmatrix} \boldsymbol{v} \\ q \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix} \qquad \text{or} \qquad \hat{\mathbf{L}}\hat{\mathbf{M}}\hat{\boldsymbol{v}} = \hat{\boldsymbol{f}} \tag{7.35}$$

for some auxiliary variable $\hat{\boldsymbol{v}} = (\boldsymbol{v}, q)$. The derived PDE system is a block lower triangular system, and can be solved in a forward substitution process, i.e. first solve the $\boldsymbol{v}$ equations, and then freeze the $\boldsymbol{v}$ variables in the second equation and solve the q equation. Moreover, with certain choice of discretizations, the same triangulation can be realized on the discretized system, i.e. $\hat{\mathbf{L}}^h\hat{\mathbf{M}}^h$ is also a block lower triangular linear system Zhu et al. (2010).

Due to the fact that each of the diagonal blocks of the discrete auxiliary system is a discretization of Laplacian operator, we can relax the whole system using a Gauss-Seidel relaxation on each component of the $\boldsymbol{v}$ variables followed by another Gauss-Seidel relaxation on p variables and achieve the same smoothing efficiency as that of the Gauss-Seidel relaxation applied on Poisson's equation. At any approximation of $\boldsymbol{v}$ and q, the approximate solutions to the augmented system can be reconstructed using (7.34).

In practice, we do not need to explicitly construct $\hat{\boldsymbol{v}}$. In a Gauss-Seidel relaxation applied on $\hat{\boldsymbol{v}}$, we iteratively solve for local corrections $\hat{v}_i \leftarrow \hat{v}_i + \delta e_i$, such that

the local residual $(\hat{f} - \hat{\mathbf{L}}\hat{\mathbf{M}}\hat{v})_i$ is zeroed out. Therefore, $\delta = (\hat{\mathbf{L}}\hat{\mathbf{M}})_{ii}^{-1}\hat{r}_i$. Such correction invokes local corrections to $\hat{\boldsymbol{u}}$ in a distributive pattern, i.e. $\hat{u}_i \leftarrow \hat{u}_i + \delta\hat{\mathbf{M}}e_i$, thus defines the distributive relaxation scheme in Algorithm 6

Algorithm 6 Distributive Smoothing

procedure DISTRIBUTIVESMOOTHING($\hat{\mathbf{L}}^h$,$\hat{\mathbf{M}}^h$,$\hat{\boldsymbol{u}}^h$,$\hat{\boldsymbol{f}}^h$)
  for v **in** $\{u_1, u_2, p\}$ **do** ▷ Must iterate on u_1 and u_2 before p
    for i **in** Lattice[v] **do** ▷ i is an equation index
      $r \leftarrow \hat{\boldsymbol{f}}_i^h - \hat{\mathbf{L}}_i^h \cdot \hat{\boldsymbol{u}}^h$ ▷ $\hat{\mathbf{L}}_i^h$ is the i-th row of $\hat{\mathbf{L}}^h$
      $\delta \leftarrow r/(\hat{\mathbf{L}}^h\hat{\mathbf{M}}^h)_{ii}$ ▷ $(\hat{\mathbf{L}}^h\hat{\mathbf{M}}^h)_{ii}$ is a precomputed constant for each component
      $\hat{\boldsymbol{u}}^{\boldsymbol{h}}$ += $\delta\, m_i^T$ ▷ m_i is the i-th row of $\hat{\mathbf{M}}^h$
    end for
  end for
end procedure

For a staggered finite difference discretization, the triangularization of the discretized system can be achieved by discretizing the change of variable operator using centered difference for gradient and divergence operators and five point stencil for the Laplacian operator as shown in Zhu et al. (2010). However, when we use finite element discretization, there is no discrete change of variables with same sparsity that leads to an exact triangularization. Instead, we discretize the gradient operator in (7.34) using the stencils derived in a finite element method, i.e. $\nabla^h = \frac{1}{\mu h^2}\mathbf{G}^{hT} = \begin{pmatrix} D_x^h \\ D_y^h \end{pmatrix}$ mapping from p variables to x and y variables with the locations illustrated in Figure 7.3-right and the stencils being:

$$D_x^h = \frac{1}{h}\begin{bmatrix} -1/8 & 1/8 \\ -3/4 & 3/4 \\ -1/8 & 1/8 \end{bmatrix}, \quad D_y^h = \frac{1}{h}\begin{bmatrix} 1/8 & 3/4 & 1/8 \\ -1/8 & -3/4 & -1/8 \end{bmatrix}. \tag{7.36}$$

Similarly, the Laplacian operator in (7.34) is discretized from a standard piecewise

bi-linear finite element discretization.

$$M_p^h = \frac{1}{h^2} \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & -8/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} \tag{7.37}$$

Although, the linear system $\hat{\mathbf{L}}^h \hat{\mathbf{M}}^h$ is not block triangular, our numerical results show that the derived distributive relaxation is able to reduce the high-frequency error components efficiently and generate an efficient multigrid solver in section 7.5.

7.4.4 Higher-order defect correction

We also adopt the idea of higher-order defect correction and develop a less expensive distributive relaxation. In a defect correction scheme for an arbitrary linear system $\mathbf{L}\boldsymbol{u} = \boldsymbol{f}$, we solve for the correction $\delta\boldsymbol{u} = \boldsymbol{u}^{\text{exact}} - \boldsymbol{u}$, which satisfies an equation $\mathbf{L}\delta\boldsymbol{u} = \boldsymbol{f} - \mathbf{L}\boldsymbol{u}$. In practice, we approximate the equation with another system $\mathbf{L}^{\text{approx}}\delta\boldsymbol{u} = \boldsymbol{f} - \mathbf{L}\boldsymbol{u}$ that is easier to solve. For example, in a multigrid correction scheme, a coarse grid system is used as the approximated equation for solving the correction at the fine grid resolution, i.e. $\mathbf{L} = \mathbf{L}^{\text{fine}}, \quad \mathbf{L}^{\text{approx}} = \mathbf{L}^{\text{coarse}}$ In the high order defect correction scheme, a lower order discretization is employed as the approximated system for solving a higher-order discretization correction. In our case, the finite difference discretization is a lower order system, and the finite element discretization is a high order system, i.e. $\mathbf{L} = \mathbf{L}^{\text{fem}}, \quad \mathbf{L}^{\text{approx}} = \mathbf{L}^{\text{fdm}}$ In other words, we consider solving the following correction equation:

$$\hat{\mathbf{L}}^{fd,h}\delta\hat{\boldsymbol{u}} = \hat{\boldsymbol{f}} - \hat{\mathbf{L}}^{fe,h}\hat{\boldsymbol{u}}.$$

One of the benefit we obtain from such approximation is that we can use the existing distributive relaxation with exact triangulation of the discretized system (7.35) introduced in Oosterlee and Gaspar (2008).

To be specific, let us rewrite the finite difference system as

$$\hat{\mathbf{L}}^{fd,h}\hat{\boldsymbol{u}} = \begin{pmatrix} \mathbf{L}_u^{fd,h} & \mathbf{G}^{fd,h^T} \\ \mathbf{G}^{fd,h} & \mathbf{D}_p^{fd,h} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{fd,h} \\ \boldsymbol{p}^{fd,h} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}^{fd,h} \\ \mathbf{0} \end{pmatrix}, \tag{7.38}$$

and rewrite the finite element system scaled by $-1/h^2$ to match the scaling of the differential equation

$$-\frac{1}{h^2}\hat{\mathbf{L}}^{fe,h}\hat{\boldsymbol{u}} = -\frac{1}{h^2}\begin{pmatrix} \mathbf{L}_u^{fe,h} & \mathbf{G}^{fe,h^T} \\ \mathbf{G}^{fe,h} & \mathbf{D}_p^{fe,h} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{fe,h} \\ \boldsymbol{p}^{fe,h} \end{pmatrix} = -\frac{1}{h^2}\begin{pmatrix} \boldsymbol{f}^{fe,h} \\ \mathbf{0} \end{pmatrix}. \tag{7.39}$$

In a high order defect correction scheme employing a finite difference discretization, we solve for correction $\delta\hat{\boldsymbol{u}} = \hat{\mathbf{M}}^{fd,h}\delta\hat{\boldsymbol{v}}$ to locally satisfy

$$\hat{\mathbf{L}}^{fd,h}\hat{\mathbf{M}}^{fd,h}\delta\hat{\boldsymbol{v}} = -\frac{1}{h^2}(\hat{\boldsymbol{f}}^{fe,h} - \hat{\mathbf{L}}^{fe,h}\hat{\boldsymbol{u}}^{\text{current}}).$$

This derives to a sparser distributive relaxation shown in Algorithm 7.

Algorithm 7 High Order Defect Correction Distributive Smoothing

procedure HighOrderDefectCorrectionDistributiveSmoothing($\hat{\mathbf{L}}^{fd}$,$\hat{\mathbf{M}}^{fd}$,$\hat{\mathbf{L}}^{fe}$,$\hat{\mathbf{M}}^{fe}$,$\hat{\boldsymbol{u}}$,$\hat{\boldsymbol{f}}^{fe,h}$)
  for v **in** $\{u_1, u_2, p\}$ **do** ▷ Must iterate on u_1 and u_2 before p
    for i **in** Lattice[v] **do** ▷ i is an equation index
      $r \leftarrow \hat{\boldsymbol{f}}_i^{fe,h} - \hat{\mathbf{L}}_i^{fe} \cdot \hat{\boldsymbol{u}}$ ▷ $\hat{\mathbf{L}}_i^{fe}$ is the i-th row of $\hat{\mathbf{L}}^{fe}$
      $\delta \leftarrow r/(\hat{\mathbf{L}}^{fd}\hat{\mathbf{M}}^{fd})_{ii}$
      $\hat{\boldsymbol{u}}$ += $\delta\, m_i^T$ ▷ m_i is the i-th row of $\hat{\mathbf{M}}^{fd}$
    end for
  end for
end procedure

The previous two types of distributive relaxations are not applicable for variables near the domain boundary. In fact, near the boundary, some of the variables in the distribution stencil may not exist. We follow the idea in Zhu et al. (2010), to temporarily build an unaugmented system in the boundary band (see Figure 7.8 left).

7.4.5 Boundary relaxation

Near the domain boundary, the distributive relaxation is not well defined; a special relaxation is required. In Neumann boundary condition case, we eliminate p from the augmented system (7.23) by left multiplying the equation with We follow the idea in Zhu et al. (2010), temporarily build an unaugmented system in a boundary band with variables incident to the boundary band cells demonstrated in Figure 7.8 left. This unaugmented system is relaxed using a few Gauss-Seidel relaxations. And we only apply distributive relaxation on variables that are incident to the distributive cells as shown in Figure 7.8 right.

We eliminate p from the augmented system (7.23) by left multiplying the equation with

$$\hat{\mathbf{U}} = \begin{pmatrix} \mathbf{I} & -\mathbf{G}\mathbf{D}_p^{-1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}.$$

Left multiplying equation (7.23) by $\hat{\mathbf{U}}$, we have

$$\hat{\mathbf{U}}\hat{\mathbf{L}}\hat{\boldsymbol{u}} = \begin{pmatrix} \mathbf{L}_u - \mathbf{G}\mathbf{D}_p^{-1}\mathbf{G}^T & \mathbf{0} \\ \mathbf{G}^T & \mathbf{D}_p \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ p \end{pmatrix} = \hat{\mathbf{U}}\hat{\boldsymbol{f}}. \tag{7.40}$$

In the first equation for $\boldsymbol{u}$, the equation is symmetric and positive definite, hence can be solved again using Gauss-Seidel relaxation. This unaugmented system is a consistent discretization to the original PDE (7.7). Although Gauss-Seidel

relaxation is not an efficient smoother for the unaugmented system if defined everywhere, for the purpose of boundary relaxation, we only build the temporary unaugmented system and relax it within a very narrow boundary band as demonstrated in Figure 7.8, and temporarily freeze the interior variables. The solution is strongly restricted by nearby interior values, therefore, the Gauss-Seidel relaxation is still efficient and stable. Typically, with about 5 to 10 sweeps of boundary relaxation before and after each interior relaxation sweep, the boundary residual is reduced to as small as the interior residual. Once we relaxed $\boldsymbol{u}$ well enough, we freeze $\boldsymbol{u}$ and substitute into the second equation in (7.40) to resolve pressure variables.

7.4.6 Boundary relaxation for the reduced system in Dirichlet boundary condition case

In Dirichlet boundary condition case, the boundary system (7.30) is a KKT system, which is indefinite, and cannot be resolved using Gauss-Seidel relaxations. Alternative approaches such as Kaczmarz relaxation or box relaxation may be efficient smoothers, however, their computational cost are much more expensive. Instead, we follow the fundamental basis method, to solve $\boldsymbol{v}^h$ from the reduced system (7.33), and reconstruct the solution of the KKT system (7.30) using $\boldsymbol{u}^h = \boldsymbol{c}^h + \mathbf{Z}^h \boldsymbol{v}^h$. Since $\mathbf{L}^h$ is symmetric positive definite, ${\mathbf{Z}^h}^T \mathbf{L}^h \mathbf{Z}^h$ is also symmetric positive definite, hence can be solved using Gauss-Seidel relaxation (see Algorithm 9).

Without loss of generality, we assume the variables to be reordered such that $\mathbf{Q} = \mathcal{I}$. Since $\mathbf{L}^h$ is symmetric positive definite, ${\mathbf{Z}^h}^T \mathbf{L}^h \mathbf{Z}^h$ is also symmetric positive definite, hence can be solved using Gauss-Seidel relaxation (see Algorithm 9). In practice, a Gauss-Seidel iteration on (7.33) iteratively solves for a correction

Algorithm 8 Dirichlet boundary relaxation - $\boldsymbol{v}^h$

$\boldsymbol{v}^h \leftarrow \mathbf{0}$
for $i = 1$ to m **do**
    $\delta \leftarrow \boldsymbol{e}_i^T \mathbf{Z}^{h^T}(\boldsymbol{f}^h - \mathbf{L}^h \boldsymbol{c}^h - \mathbf{L}^h \mathbf{Z}^h \boldsymbol{v}^h)/(\boldsymbol{e}_i^T \mathbf{Z}^{h^T} \mathbf{L}^h \mathbf{Z}^h \boldsymbol{e}_i)$
    $\boldsymbol{v}^h \mathrel{+}= \delta \boldsymbol{e}_i$
end for

on each single degree of freedom by solving the following scalar equation:

$$\boldsymbol{e}_i^T \mathbf{L}_r^h (\boldsymbol{v}^h + \delta \boldsymbol{e}_i) = \boldsymbol{e}_i^T \mathbf{Z}^{h^T}(\boldsymbol{f}^h - \mathbf{L}^h \boldsymbol{c}^h), \tag{7.41}$$

where $\mathbf{L}_r^h = \mathbf{Z}^{h^T} \mathbf{L}^h \mathbf{Z}^h$ i.e.

$$(\mathbf{L}_r^h)_{ii} \delta = \boldsymbol{e}_i^T \mathbf{Z}^{h^T}(\boldsymbol{f}^h - \mathbf{L}^h \boldsymbol{c}^h - \mathbf{L}^h Z^h \boldsymbol{v}^h) = \boldsymbol{e}_i^T \mathbf{Z}^{h^T}(\boldsymbol{f}^h - \mathbf{L}^h \boldsymbol{u}^h), \tag{7.42}$$

and then apply the correction: $\boldsymbol{v}^h \leftarrow \boldsymbol{v}^h + \delta \boldsymbol{e}_i$. Equivalently, $\boldsymbol{u}^h$ is updated as $\boldsymbol{u}^h \leftarrow \boldsymbol{u}^h + \delta \mathbf{Z}^h \boldsymbol{e}_i$ Therefore we can equivalently solve for a correction on $\boldsymbol{u}$ to emulate the Gauss-Seidel iteration on $\boldsymbol{v}^h$ (see Algorithm 10).

Algorithm 9 Dirichlet boundary relaxation - $\boldsymbol{v}^h$

$\boldsymbol{v}^h \leftarrow \mathbf{0}$
for $i = 1$ to m **do**
    $\delta \leftarrow \boldsymbol{e}_i^T \mathbf{Z}^{h^T}(\boldsymbol{f}^h - \mathbf{L}^h \boldsymbol{c}^h - \mathbf{L}^h \mathbf{Z}^h \boldsymbol{v}^h)/(\mathbf{L}^h)_{ii}$
    $\boldsymbol{v}^h \mathrel{+}= \delta \boldsymbol{e}_i$
end for

Algorithm 10 Dirichlet boundary relaxation - $\boldsymbol{u}^h$

$\boldsymbol{u}^h \leftarrow \boldsymbol{c}^h$
for $i = 1$ to m **do**
    $\delta \leftarrow \boldsymbol{e}_i^T \mathbf{Z}^{h^T}(\boldsymbol{f}^h - \mathbf{L}^h \boldsymbol{u}^h)/(\mathbf{L}^h)_{ii}$
    $\boldsymbol{u}^h \mathrel{+}= \delta \mathbf{Z}^h e_i$
end for

During the Dirichlet boundary relaxation applied on $\boldsymbol{u}^h$, a numerical error may be introduced driving $\boldsymbol{u}^h$ away from the linear space $\boldsymbol{c}^h + \mathbf{Z}^h \boldsymbol{v}^h$. Also, when

we apply coarse grid correction in Algorithm 5, the corrected solution $\hat{\boldsymbol{u}}^h + \mathbf{P}\hat{\boldsymbol{u}}^{2h}$ may not be in the solution space. Therefore, a projection onto the solution space needs to be applied after the prolongation step. First of all, since the projected solution $\boldsymbol{u}_p^h = \boldsymbol{c}^h + \mathbf{Z}^h\boldsymbol{v}^h$, for some $\boldsymbol{v}^h$, and according to the definition of $\mathbf{Z}^h$ and $\boldsymbol{c}^h$ in (7.31) and (7.32), we have $\boldsymbol{v}^h = \mathcal{Q}\boldsymbol{u}^h$, where $\mathcal{Q}$ projects a solution vector to a sub-vector by eliminating the degrees of freedom that correspond to aggregation representative nodes. Therefore, the projected solution $\boldsymbol{u}_p^h = \boldsymbol{c}^h + \mathbf{Z}^h\mathcal{Q}\boldsymbol{u}^h$.

7.4.7 Coarsening

In a geometric multigrid method, we define a discretization on each level. In deep interior region, In order to restrict residuals from fine grid to coarse grid, we apply a restriction operator $\mathbf{R}$ for each component defined on the staggered grids with stencils illustrated in Figure 7.9 . We consider two types of prolongation stencils. First, we consider prolongation $\mathbf{P}_{lo} = 2^d\mathbf{R}$. Second, we consider piecewise bi-linear interpolation for $\boldsymbol{u}$ in combination with the same pressure prolongation as in $\mathbf{P}_{lo}$, which we donate as $\mathbf{P}_{hi}$.

In deep interior region, we use the same restriction operator $\mathbf{R}$ as in (5.7) with stencils shown in Figure 7.9. We consider two types of prolongation stencils. First, we consider prolongation $\mathbf{P}_{lo} = 2^d\mathbf{R}$. Second, we consider piecewise bi-linear interpolation for $\boldsymbol{u}$ in combination with the same pressure prolongation as in $\mathbf{P}_{lo}$, which we donate as $\mathbf{P}_{hi}$.

However, on the boundary, there is no guarantee that all dependencies of the coarse grid variable restriction stencils are active fine grid degrees of freedom. Therefore, we truncate our restriction stencils to the active degrees of freedom, which is equivalent to restrict zero residual from inactive regions. Also, when a Dirichlet boundary condition presents, we can not compute the residuals $\boldsymbol{r}^h =$

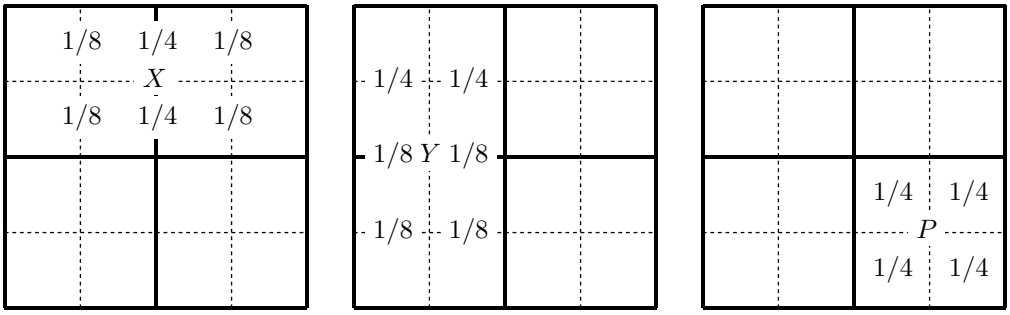


Figure 7.9: Restriction operator stencils.

$\boldsymbol{f}^h - \mathbf{L}^h\boldsymbol{u}^h - \mathbf{B^h}^T\boldsymbol{\lambda}^h$ when a $\boldsymbol{\lambda}$ value is involved. In this case, we apply a boundary relaxation strong enough such that the boundary residuals is smaller than interior relaxations, and restrict zero boundary residual for these equations.

The coarse grid constraint system right hand side should have been computed from the restriction of fine grid constraint system residual. However, due to the fact that our solutions $\boldsymbol{u}_p^h = \boldsymbol{c}^h + \mathbf{Z}^h\boldsymbol{v}^h$ always satisfy boundary constraint exactly, the coarse grid Dirichlet boundary condition is always zero. Also, prolongation is implemented in a distributive way, i.e. we iterate over the active coarse grid corrections, and distribute their values to all active fine level degrees of freedom. Near the domain boundary, this is equivalent to prolongate zero correction from exterior coarse grid locations, which is reasonable. We notice that this prolongation may lead to a solution away from the fundamental basis solution. Therefore, a projection process is applied immediately after prolongation $\boldsymbol{u}^h \leftarrow \boldsymbol{c}^h + \mathbf{Z}^h\mathcal{Q}\boldsymbol{u}^h$.

So far, we discussed the components of a multigrid method, and we continue by discussing the numerical results.

7.5 Numerical examples

We investigate two aspects of our algorithm: discretization error and multigrid efficiency. 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 is enclosed by a smooth curve connecting 8 tangential circles with centers

$$\begin{aligned} \boldsymbol{c}_1 = (0.25, 0.25); \quad \boldsymbol{c}_2 = (0.75, 0.25); \\ \boldsymbol{c}_3 = (0.25, 0.75); \quad \boldsymbol{c}_4 = (0.75, 0.75); \\ \boldsymbol{s}_1 = (0.5, 0.6875); \quad \boldsymbol{s}_2 = (0.5, 0.3125); \\ \boldsymbol{s}_3 = (0.3125, 0.5); \quad \boldsymbol{s}_4 = (0.6875, 0.5); \end{aligned}$$

   and radius 0.2 for the first 4 circles and $r_s = \dfrac{\sqrt{17}}{4} - 0.2$ for the last 4 circles. The radius r_s is chosen such that the circle curves are tangential hence generate a smooth boundary.

   The keyhole domain can also be represented by the zero levelset of the following function:

$$\varphi(\boldsymbol{x}) = \max(\min(\text{dist}(\boldsymbol{x}, \boldsymbol{c}_i, 0.2), \text{dist}(\boldsymbol{x}, \mathbf{0}, r_0)), -\min(\text{dist}(\boldsymbol{x}, \boldsymbol{s}_i, r_s))$$

   where $\text{dist}(\mathbf{x}, \mathbf{x}_0, r) = |\mathbf{x} - \mathbf{x_0}| - r$, and $r_0 = \left| \frac{0.2}{\sqrt{17}}(4, 1) - (0.25, 0.25) \right|$.

   A constant divergence deformation is considered, giving the exact boundary

conditions and the exact solution for the purpose of error computation.

$$\phi_1(x, y) = 2x + \frac{1}{2}\cos \pi x \sin \pi y \tag{7.43}$$

$$\phi_2(x, y) = 2y - \frac{1}{2}\sin \pi x \cos \pi y \tag{7.44}$$

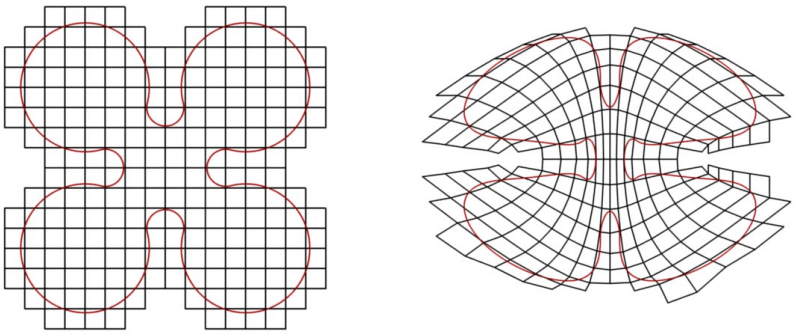


Figure 7.10: A keyhold domain and its deformation.. Left: undeformed keyhole domain; right: deformed keyhole domain

2. **Flower domain** A flower-shaped domain with inner radius 0.2, outer radius 0.4 is considered with a levelset function:

$$\varphi(\boldsymbol{x}) = \text{dist}(\boldsymbol{x}, \mathbf{0}.5, 0.3 + 0.1\cos 5\theta)$$

where θ is the argument of (x, y). A deformation with spatially varying divergence is considered as an exact solution.

$$\phi_1(x, y) = \frac{2x}{\sqrt{\pi}}\cos \frac{\pi}{2}y \tag{7.45}$$

$$\phi_2(x, y) = \frac{2x}{\sqrt{\pi}}\sin \frac{\pi}{2}y \tag{7.46}$$

3. **Spiral domain** We consider a spiral shaped domain, defined by the zero

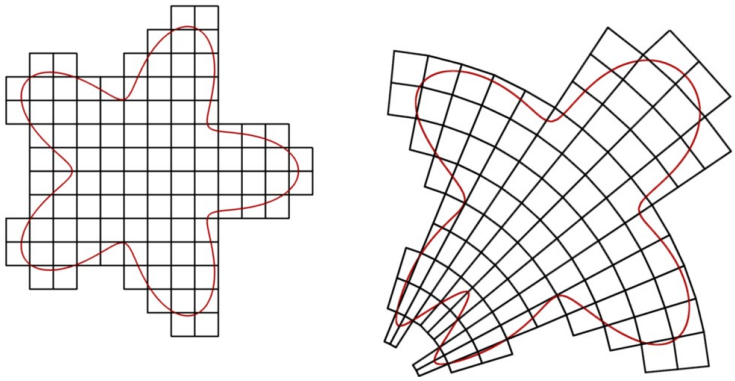


Figure 7.11: A flower domain and its deformation. Left: undeformed flower domain; right: deformed flower domain

levelset of

$$\varphi(\boldsymbol{x}) = r(\boldsymbol{y}) - (0.33 + 0.08\cos 5\theta(\boldsymbol{y})^{\frac{1}{3}})$$

where $\boldsymbol{y}$ is $\boldsymbol{x} - (0.5, 0.5)$ rotated around $(0.5, 0.5)$ by $\theta = 14(2r(\boldsymbol{x}))^{\frac{1}{6}}$ and the deformation:

$$\phi_1(x, y) = \left(\frac{1}{2}x + \frac{1}{2}\right)\cos\left(\frac{\pi}{6} + \frac{2}{3}\pi y\right) \tag{7.47}$$

$$\phi_2(x, y) = \left(\frac{1}{2}x + \frac{1}{2}\right)\sin\left(\frac{\pi}{6} + \frac{2}{3}\pi y\right) \tag{7.48}$$

7.5.1 Discretization error

All our testing domains are embedded in a $[0, 1]^2$ domain, and we discretize this square domain with a regular grid of different resolutions ranging from 32 to 1024 in each direction. We plotted $\log_2 |\boldsymbol{u}^{\text{exact}} - \boldsymbol{u}|_\infty$ versus $\log_2$ resolution and estimated the solution accuracy order by fitting the data with a linear function. We remove the Neumann boundary condition null space by enforcing a non-embedded Dirichlet condition on all degrees of freedom within the domain

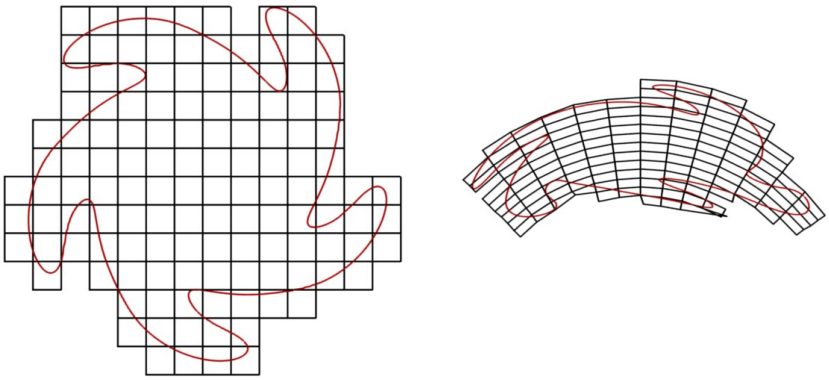


Figure 7.12: A keyhold domain and its deformation. Left: undeformed spiral domain, right: deformed spiral domain

$[7/16, 9/16]^2$. From the plotted error convergence behavior, we observe a second-order convergence for all three types of domains (see Figure 7.13, Figure 7.14, Figure 7.15); for both Neumann and Dirichlet boundary conditions; and for a wide range of material parameters including near-incompressible materials. We notice that the order of accuracy is slightly smaller for domains with complicated boundaries. An important source of the inaccuracy is introduced by the inconsistent domain discretization at different resolutions.

7.5.2 Multigrid efficiency

We also investigated the efficiency of multigrid methods. First of all, we consider a periodic boundary condition problem defined on $[0, 1]^2$ and with the exact solution being

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

Although periodic boundary condition will not appear in practical elasticity

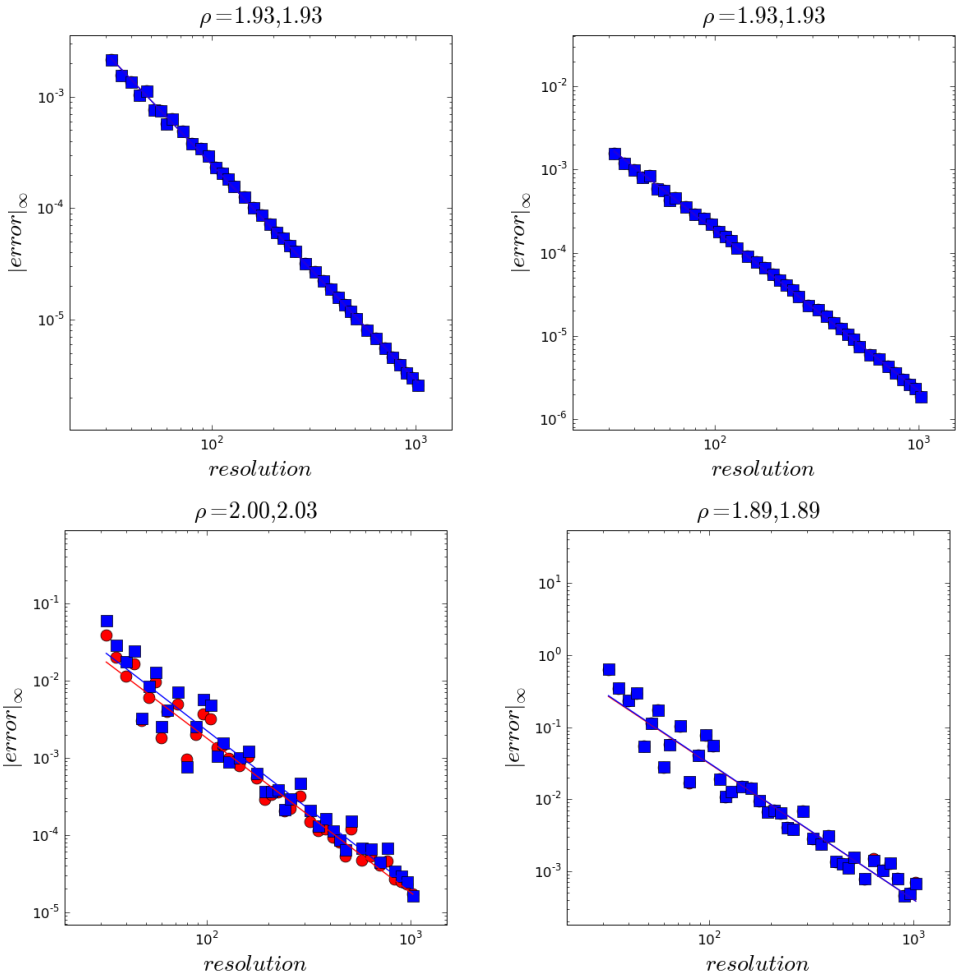


Figure 7.13: Order of accuracy for the keyhole domain. Top: Neumann boundary condition; bottom: Dirichlet boundary condition; left: Poisson's ratio=0.3; right: Poisson's ratio=0.49; square marker: x component; circle marker: y component.

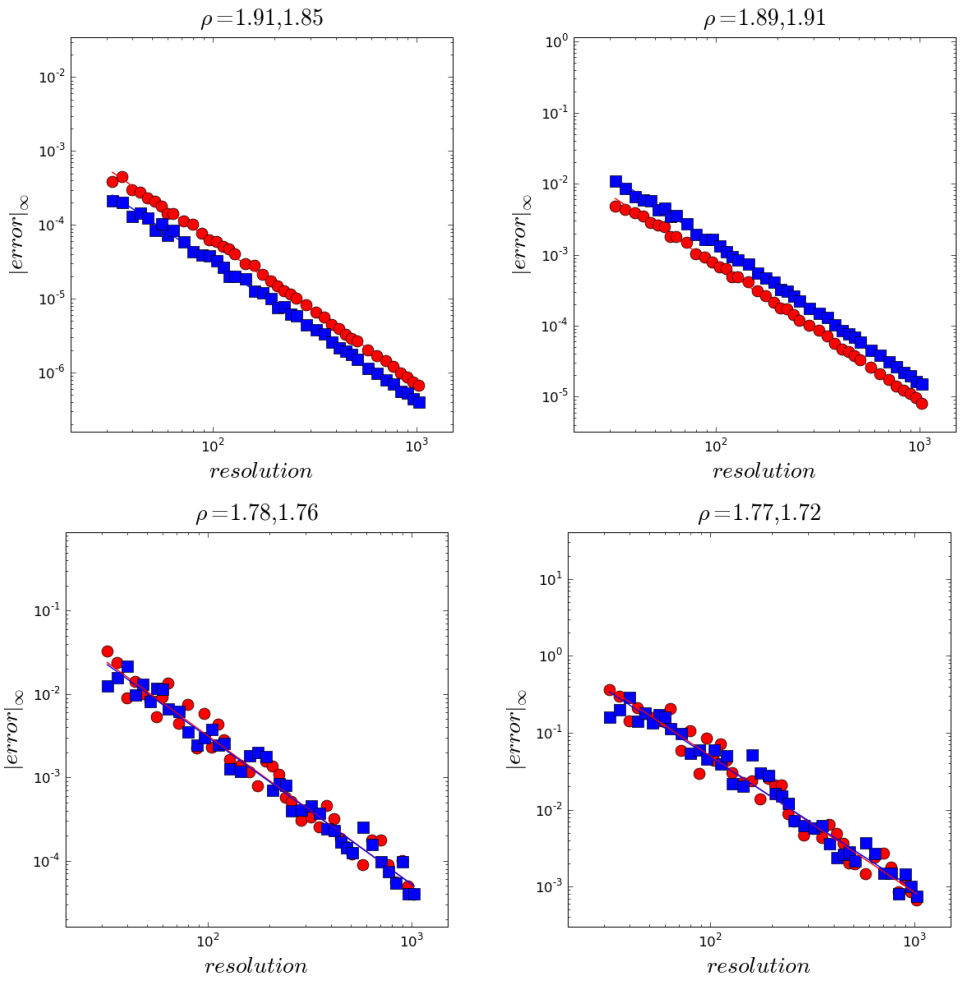


Figure 7.14: Order of accuracy for the flower domain. Top: Neumann boundary condition, bottom: Dirichlet boundary condition; left: Poisson's ratio=0.3; right: Poisson's ratio=0.49; square marker: x component; circle marker: y component.

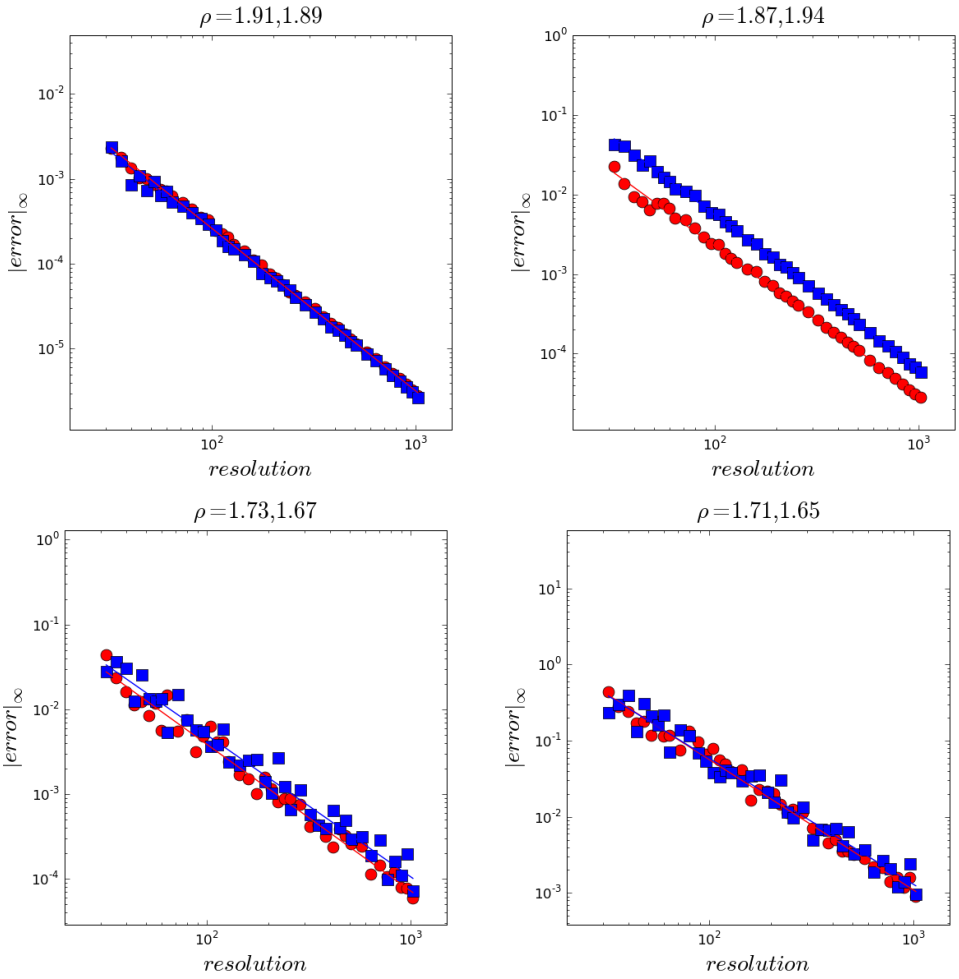


Figure 7.15: Order of accuracy for the spiral domain. Top: Neumann boundary condition; bottom: Dirichlet boundary condition; left: Poisson's ratio=0.3; right: Poisson's ratio=0.49; square marker: x component; circle marker: y component.

problems, we consider periodic boundary condition problem to evaluate multigrid solver avoiding issues that may arise with boundary relaxation. We first fix the problem resolution to 128×128 and apply finite element distributive relaxation and the distributive relaxation for the finite difference defect correction problem as the interior relaxations. And we also apply the bilinear prolongation and a prolongation with $\mathbf{P} = 4\mathbf{R}^T$. While low incompressibility problems generates convergence rates no larger than 0.3 for multigrid V-(1,1) cycle with all different prolongation and distribution options, we focus on the the harder high-incompressible case with Poisson's ratio being 0.49, and investigate both V-(1,1) cycle and W-(1,1) cycle convergence. As shown in Table 7.1, both finite element distributive relaxation and finite difference defect correction scheme generate convergence rate less than 0.5 with a multigrid V-(1,1) cycle. Although finite difference distributive relaxation generates slower convergence than finite element distributive relaxation for V-(1,1) cycle, with the help of a bilinear interpolation or W-(1,1) cycle, we are able to generate as good convergence rate as 0.23.

We further investigates the multigrid convergence rate for various resolutions by sticking to one scheme which uses finite element distribution, low order prolongation and a V-(1,1) cycle, and plot the convergence rate for problems discretized with resolutions from 32 to 1024 (see Figure 7.16). A consistent multigrid convergence rate is observed under refinement.

While all schemes give a nice convergence rate in periodic cases, the convergence rate with Neumann boundary condition and Dirichlet boundary condition various. In non-trivial boundary condition cases, the convergence rate is mainly restricted by the efficiency of a boundary relaxation. Therefore, the convergence rate of a V-(1,1) cycle and a W-(1,1) cycle are very similar, and also different

boundary condition	distribution	multigrid cycle	$\mathbf{P}_{hi}$	$\mathbf{P}_{lo}$
Periodic	FD	V-(1,1)	0.24	0.42
	FD	W-(1,1)	0.23	0.25
	FE	V-(1,1)	0.13	0.24
	FE	W-(1,1)	0.13	0.30
Dirichlet	FD	V-(1,1)	0.72	0.72
	FD	W-(1,1)	0.72	0.72
	FE	V-(1,1)	0.37	0.36
	FE	W-(1,1)	0.42	0.42
Neumann	FD	V-(1,1)	0.70	0.70
	FD	W-(1,1)	0.68	0.68
	FE	V-(1,1)	0.50	0.50
	FE	W-(1,1)	0.35	0.35

Table 7.1: Multigrid V-(1,1) cycle asymptotic convergence rates - periodic boundary condition. The discretization resolution is 128×128, and the Poisson's ratio is 0.49. For optional prolongations, $\mathbf{P}_{hi}$ is bilinear interpolation and $\mathbf{P}_{lo}$ is for $\mathbf{P} = 4\mathbf{R}^T$. For the optional distributions, FE is the distribution matrix discretized with bilinear finite element method, and FD is using defect correction by employing finite difference distributive relaxation.

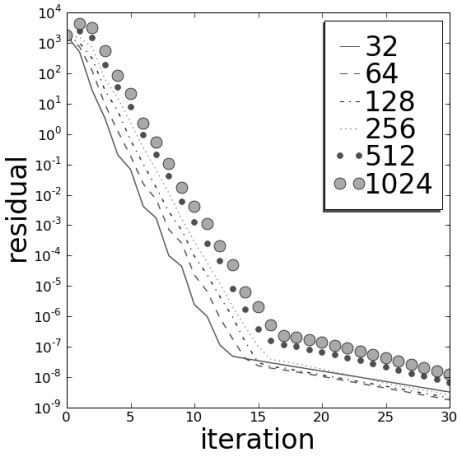


Figure 7.16: Multigrid V-(1,1) cycle convergence - periodic boundary condition with problem resolution from 32 to 1024. The interior relaxation is a finite element distributive relaxation, and the Poisson's ratio is 0.49.

prolongation schemes generate very similar convergence rates (see Tabel 7.1 for the convergence rate of all algorithm options for the flower domain problem at a fixed resolution of 128×128). The difference between using a finite element distributive relaxation and a finite element distributive relaxation reflects the efficiency of the whole smoother in combination with the boundary relaxations.

We further investigate the convergence rate under refinement. Due to the fact that different prolongation scheme and multigrid cycle generates similar convergence rate. We only plot the asymptotic convergence rate of V(1,1) cycle with $\mathbf{P} = 4\mathbf{R}^T$ and using finite element distributive relaxation in interior. For both Dirichlet and Neumann boundary condition problem, we plot the asymptotic convergence rate of resolution from 32 to 1024 and the residual reduction at each iterations for representative resolution numbers that are powers of 2. We observed consistent convergence rate at all resolutions (see Figure 7.17 and Figure 7.18).

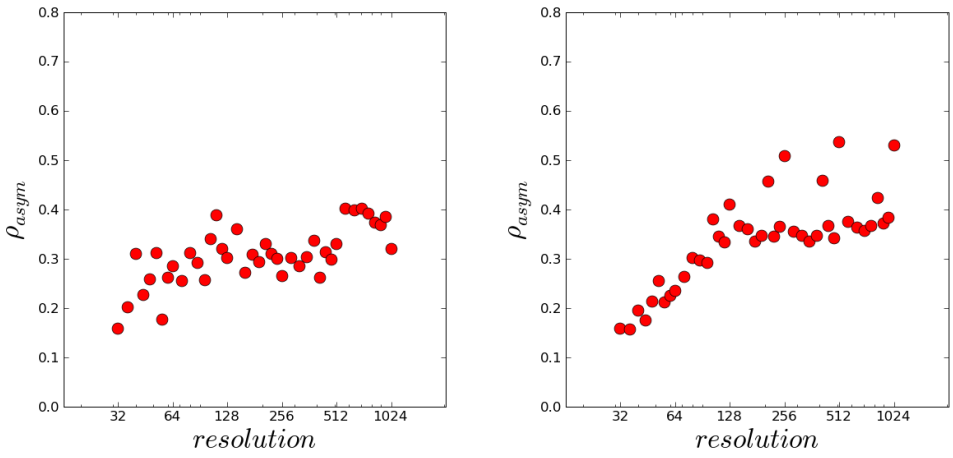


Figure 7.17: Multigrid V-(1,1) cycle convergence rates - flower domain with problem resolution from 32 to 1024 . The interior relaxation is a finite element distributive relaxation, and the Poisson's ratio is 0.49. Left: Dirichlet boundary condition; right: Neumann boundary condition.

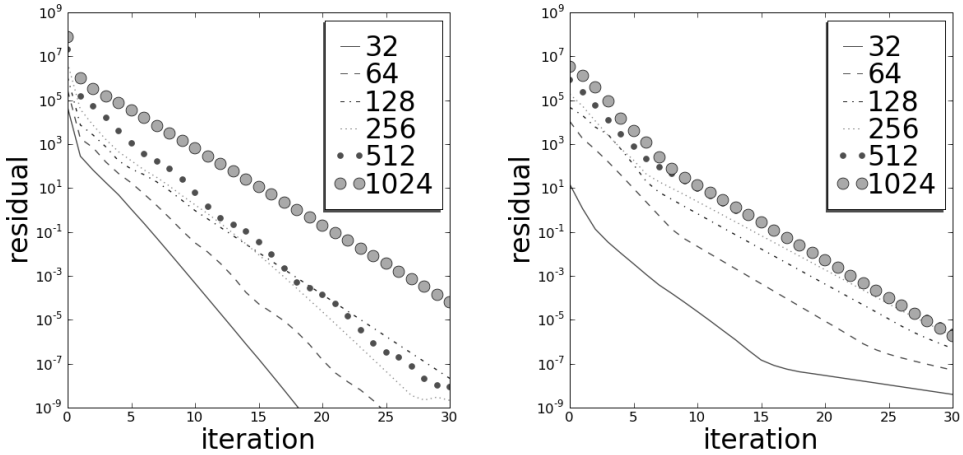


Figure 7.18: Multigrid V-(1,1) cycle residual convergence - flower domain with problem resolution from 32 to 1024 . The interior relaxation is a finite element distributive relaxation, and the Poisson's ratio is 0.49. Left: Dirichlet boundary condition; right: Neumann boundary condition.

CHAPTER 8

Soft constraint system

In the previous chapters, we propose the constraints by enforcing Dirichlet boundary conditions. In practical applications, such conditions are not alway convenient to be specified. For example, in character animations, artists design the bones motion paths which can be used to prescribe the Dirichlet boundary conditions. However, the target configurations can be complicated. As a consequence, the approximated bone positions may propose conflicting Dirichlet conditions at discrete level or leading to solutions that are physically not plausible. Soft constraints, although as a type of inexact constraints, are more robust and applicable to more general control problems.

We implement a soft constraint discretization and combine it with the previous co-rotational linear elasticity dynamic system. We also develop a multigrid method to solve the soft constraint system efficiently. Finally, the collision problem is modeled as soft constraint problems and generating robust results.

8.1 Soft constraint energy

In elasticity problem, we minimize an elasticity energy $\mathcal{E}_{elasticity}$. We modify this energy by adding a penalty energy which estimates the discrepancy between the embedded positions and the target positions $\hat{\mathcal{E}} = \mathcal{E}_{elasticity} + \mathcal{E}_{constraint}$. This discrepancy is measured by the geometric measure of their differences:

1. point constraints: $\mathcal{E}_{constraint} = \frac{\tau_p}{2} \sum_i |\boldsymbol{\phi}(\boldsymbol{x}_i) - \boldsymbol{\phi}_i^{\text{target}}|^2$;

2. line constraints: $\mathcal{E}_{constraint} = \frac{\tau_l}{2} \int_\Gamma |\boldsymbol{\phi}(\boldsymbol{x}(t)) - \boldsymbol{\phi}^{\text{target}}(t)|^2 \, dl$;

3. surface constraints: $\mathcal{E}_{constraint} = \frac{\tau_s}{2} \int_\Omega |\boldsymbol{\phi}(\boldsymbol{x}(s,t)) - \boldsymbol{\phi}^{\text{target}}(s,t)|^2 \, dsdt$.

We discretize the above three types of constraints using a spring system between a set of material points (we call constraint sources) and the corresponding constraint targets.

$$\mathcal{E}_{constraint} = \frac{\tau}{2} \sum_i |\boldsymbol{\phi}(x_i) - \boldsymbol{\phi}_i^{\text{target}}|^2 ,$$

where $\boldsymbol{\phi}(x_i)$ is approximated by the embedded value, which is interpolated from the discrete solution $\boldsymbol{\phi}^h$ via an interpolation mapping $\mathbf{A}^h$ from the solution vector to the embedded vector.

Therefore, the discretized linear system is

$$(\mathbf{L}^h - \tau h^{(\gamma-d)} {\mathbf{A}^h}^T \mathbf{A}^h)\boldsymbol{\phi} = \boldsymbol{f} - \tau h^{(\gamma-d)} {\mathbf{A}^h}^T \boldsymbol{\phi}^{\text{target}},$$

where $\gamma = 0, 1, 2$ for point constraints, line constraints and surface constraints. $\mathbf{L}^h$ is a discretized system of the elasticity force $-\frac{\mathrm{D}\mathcal{E}_{elasticity}}{\mathrm{D}\phi}$. In this chapter, we consider both linear and co-rotational linear elasticity model with finite difference or mixed finite element discretizations. Although our system is a discretization of the augmented equation (6.2), effectively, we classify the variables close to the constraints as boundary variables, and treat them in the boundary relaxation. In the boundary band, we recover an unaugmented system as in 9.1, 6.1.4 and 7.4.5, and apply boundary relaxation to (5.6). The essential part of the boundary relaxation is a Gauss-Seidel relaxation on the unaugmented system $\mathbf{L}^{\text{unaug}}\boldsymbol{u} =$

$\boldsymbol{f}^{\text{unaug}}$.

With convex elasticity energy $\mathcal{E}_{elasticity}$, $\mathbf{L}^h$ is negative definite, and ${\mathbf{A}^h}^T\mathbf{A}^h$ is symmetric positive definite, thus the overall system is symmetric definite, which can be relaxed again using Gauss-Seidel relaxation.

8.2 Coarsening of soft constraint operator

We split the operators and generate the coarse level discretization for each of the two operators: $\mathbf{L}^h$ and ${\mathbf{A}^h}^T\mathbf{A}^h$. $\mathbf{L}^h$ is coarsened in exactly the same way as we discussed before; the coarsening of $\mathbf{K}^h = {\mathbf{A}^h}^T\mathbf{A}^h$ is generated in three different approaches.

8.2.1 Galerkin coarsening

For an arbitrary restriction $\mathbf{R}^h$, prolongation $\mathbf{P}^h$ and a linear system $\mathbf{K}^h\boldsymbol{u}^h = \mathbf{f}^h$, Galerkin coarsening defines the following a coarse grid problem

$$\mathbf{R}^h\mathbf{K}^h\mathbf{P}^h\boldsymbol{u}^{2h} = \mathbf{R}^h(\boldsymbol{f}^h - \mathbf{K}^h\boldsymbol{u}^h).$$

For symmetric positive definite system $\mathbf{K}^h$, if $\mathbf{R}^h = \mathbf{P}^h$ or is a constant scaling away from $\mathbf{P}^h$, then the Galerkin coarsening solution gives the optimal coarse grid correction $\boldsymbol{U} = \boldsymbol{u}^{2h}$ to minimize

$$\frac{1}{2}(\boldsymbol{u}^h + \mathbf{P}^h\boldsymbol{U})^T\mathbf{K}^h(\boldsymbol{u}^h + \mathbf{P}^h\boldsymbol{U}) - {\boldsymbol{f}^h}^T(\boldsymbol{u}^h + \mathbf{P}^h\boldsymbol{U}).$$

We implemented Galerkin coarsening in an efficient way using the sparsity and uniformity of restriction and prolongation.

Before applying Galerkin coarsening in a multigrid scheme, a coarsening test

is applied to convince of the quality of Galerkin coarsening (Algorithm 11).

Algorithm 11 CoarseningTest

procedure CoarseningTest($\mathbf{L}^h$,$\boldsymbol{f}^h$)
    solve $\boldsymbol{\phi}_0^h \leftarrow (\mathbf{L}^h)^{-1}\boldsymbol{f}^h$ exactly
    define a coarse grid correction $\boldsymbol{\phi}_0^{2h} \leftarrow \delta^{2h}(\boldsymbol{x}_0)$, and coarse grid right hand side $\boldsymbol{f}^{2h} \leftarrow 0$
    compute coarse grid residual $\boldsymbol{f}_0^{2h} \leftarrow -\mathbf{A}^{2h}\boldsymbol{\phi}_0^{2h}$
    apply coarse grid perturbation by $\boldsymbol{\phi}_1^h \leftarrow \boldsymbol{\phi}_0^h + \mathbf{P}\boldsymbol{\phi}_0^{2h}$
    compute and restrict residual $\boldsymbol{f}_1^{2h} = \mathbf{R}(\boldsymbol{f}^h - \mathbf{L}^h\boldsymbol{\phi}_1^h)$
    compare $\boldsymbol{f}_0^{2h}$ with $\boldsymbol{f}_1^{2h}$
    solve $\mathbf{A}^{2h}\boldsymbol{\phi}_1^{2h} \leftarrow \boldsymbol{f}_1^{2h}$ exactly
    compare $\boldsymbol{\phi}_0^{2h}$ with $\boldsymbol{\phi}_1^{2h}$
end procedure

Since $\boldsymbol{f}_1^{2h} = \mathbf{R}(\boldsymbol{f}^h - \mathbf{L}^h\boldsymbol{\phi}_1^h) = \mathbf{R}(\boldsymbol{f}^h - \mathbf{L}^h\boldsymbol{\phi}_0^h - \mathbf{L}^h\mathbf{P}\boldsymbol{\phi}_0^{2h}) \approx -\mathbf{R}\mathbf{L}^h\mathbf{P}\boldsymbol{\phi}_0^{2h}$, we expect $\boldsymbol{f}_1^{2h}$ to be equal to $\boldsymbol{f}_0^{2h}$, and $\boldsymbol{\phi}_0^{2h} = -\boldsymbol{\phi}_1^{2h}$. In practice, due to the fact that our unaugmented residual is generated from the restricted augmented residual, we cannot directly verify this relationship, instead, $\boldsymbol{\phi}_0^{2h}$ and $\boldsymbol{\phi}_1^{2h}$ are plotted and visually $\boldsymbol{\phi}_1^{2h}$ approximates $\boldsymbol{\phi}_1^{2h}$ well (see Figure 8.2).

8.2.2 Re-discretization and natural coarsening

We investigate a coarse grid discretization generated by re-discretizing the constraint energy on the coarser level. In the following part of this chapter, we denote $\mathbf{R}$ as the restriction operator and $\mathbf{P}$ as the prolongation operator.

We show that for bi-linear interpolation on collocated grids, energy discretization on coarse grid leads to the exact Galerkin coarsening. In fact, $\mathbf{A}^h\mathbf{P}\boldsymbol{\phi}^{2h}$ first interpolates from a coarse solution to fine solution using bi-linear interpolation, then interpolate from fine solution to constraint points set using again bi-linear interpolation, the result is the same as directly interpolated from a coarse grid solution with bi-linear interpolation for all sampling points. By uniqueness,

$\mathbf{A}^h\mathbf{P} = \mathbf{A}^{2h}$. If $\mathbf{R} = \frac{1}{2^d}\mathbf{P}^T$

$$\mathbf{R}(-\frac{\tau}{h^d}\mathbf{A}^{h^T}\mathbf{A}^h)\mathbf{P} = -\frac{\tau}{(2h)^d}(\mathbf{A}^h\mathbf{P})^T\mathbf{A}^h\mathbf{P} = -\frac{\tau}{(2h)^d}(\mathbf{A}^{2h})^T\mathbf{A}^{2h}$$

Therefore, the energy rediscretization on a coarse grid leads to the exact Galerkin coarsening.

Based on that observation, we try to use energy rediscretization as a promising candidate for staggered grid coarsening. In practice, it does keep a nice multigrid convergence rate even when the soft constraint term dominates.

The energy rediscretization scheme is easy to implement, however, in this approach, the energy of all discretization levels depends on all constraint sources regardless of level of discretization. Therefore, even for the coarsest grid, the interpolation operators for all constraint sources need to be accumulated. This leads to an important increase for cost of coarse grid discretization. Galerkin coarsening has a computational cost that reduces with the number of degrees of freedom, so we are also interested in directly applying Galerkin coarsening .

8.2.3 Subsampling

In our testing examples, the line constraint is given in an analytical way, and this derives a natural subsampling coarsening (see Figure 8.1). The subsampling reduces sample points by half, therefore its weight for each subsampled constraint need to be doubled to keep the consistence of energy, and for surface constraint the weight needs to be quadrupled.

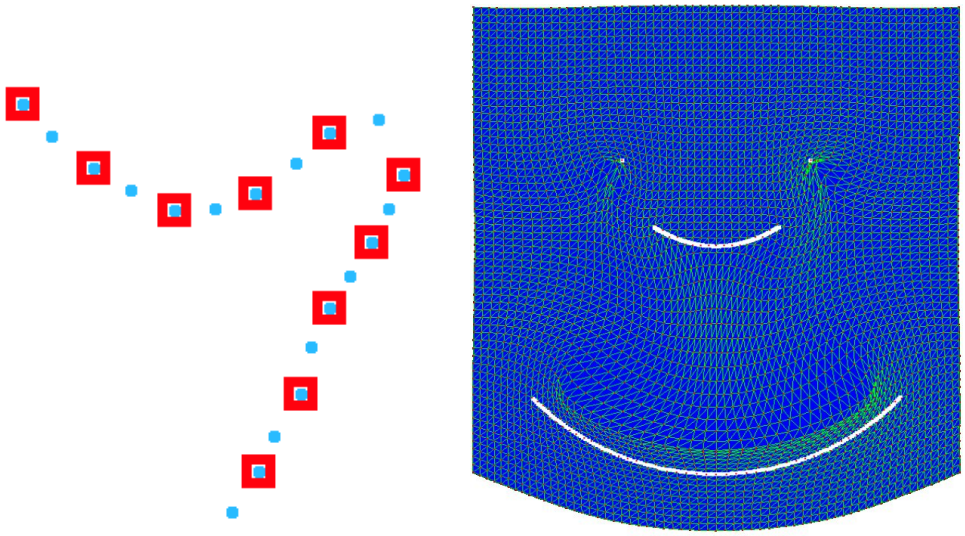


Figure 8.1: Soft constraints samples. Left: line constraints coarsening; right: two-dimensional example with point and line constraints.

8.3 Examples and results

8.3.1 Two-dimensional examples

We implement the soft constraint discretization and a multigrid method and tested them on a two-dimensional example.

We start with a quasistatic problem where point constraints and line constraints are applied (see Figure 8.1). The active region is a square domain with Dirichlet boundary conditions defined on left and right sides, and free surface boundary conditions defined on the other two sides. We first compute the solutions using an LU decomposition exact solver for different constraint stiffness values for reference. We visually validated that under high stiffness limit, the constraint solution approaches constraint target.

Single level relaxations converge slower and slower as τ increases to 10^9. Visually, the solver has difficulty in reducing error components shown in Figure 8.3.

Moreover, a high frequency residual supported by the constraint system persists. In practice, a box smoother based on PLU decomposition or an approximated box smoother with two Gauss-Seidel relaxation sweeps within each box will increase the smoothing effect significantly.

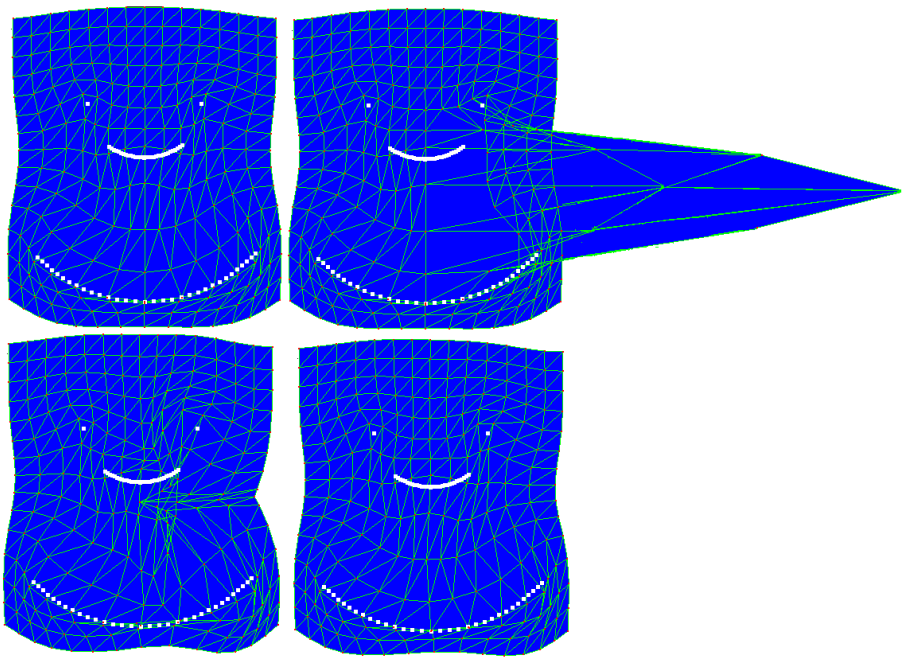


Figure 8.2: Constraint coarsening test (top left: exact solution; top right: solution perturbed by δ on one coarse grid variable. Bottom left: solution after coarse grid correction; bottom right: after post relaxation)

We implemented the multigrid method for another example with both point and line constraints. We used Galerkin coarsening and listed in Table 8.1 the multigrid convergence rate for two-grid cycle, V-(1,1) cycle and W-(1,1) cycle on a 256×256 resolution domain. A coupled box relaxation is used near the constraints with a boundary band of width being 4 cells. The box relaxation is approximated with 2 or 3 Gauss-Seidel local sweeps over a box of width up to 3 cells.

Table 8.1: 2D soft constraint multigrid convergence

τ	multigrid cycle	box iterations	box size	convergence
10^7	two-grid	5	1	0.29
10^8	two-grid	5	1	0.31
10^9	two-grid	5	1	0.89
10^9	two-grid	1	3	0.29
10^7	V-(1,1)	5	1	0.49
10^8	V-(1,1)	5	1	0.48
10^9	V-(1,1)	5	1	0.80
10^9	V-(1,1)	1	2	0.50
10^7	W-(1,1)	5	1	0.30
10^8	W-(1,1)	5	1	—-
10^9	W-(1,1)	1	2	0.32

8.3.2 Stiff constraint

For τ being relatively small, a small number of boundary relaxations generate efficient V cycle or W cycle, but for very stiff constraints, we cannot achieve a good convergence even with a lot of boundary relaxations. From a visualization of the the stagnated solution (Figure 8.3) and the residual image, we realize that one of the reason is the inefficient smoother near the constraints. Due to the narrow support of constraint operator, the constraint system is often singular, and when the constraint dominates, the boundary system becomes ill-conditioned, and a Gauss-Seidel solver fails. Fortunately, a wider box solver will increase the smoothing effect significantly. In Table 8.1, we can see that even for very stiff constraint, an approximated box solver with two Gauss-Seidel sweeps on a box of 2 or 3 cells wide will generate a two grid multigrid cycle with convergence rate as low as 0.3.

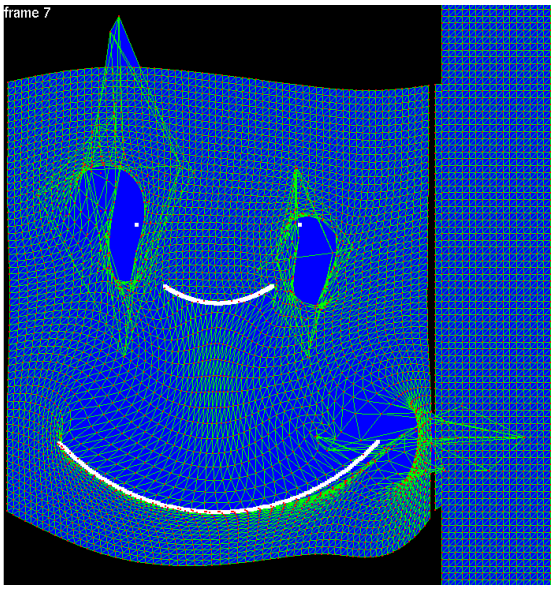


Figure 8.3: High stiffness difficulty

8.3.3 Collision

We also applied our method to three dimensional cases and resolve collision problems. First of all, we consider geometric surfaces with analytical signed distance function. At each time step, we compute the levelset value of all particles from one object agains the signed distance function of the other objects. We apply constraints to all particles that has negative values of $\varphi(\boldsymbol{\phi})$. A constraint is applied on all such particles with target being the particle projected along normal direction of the levelset gradient (see Figure 8.4).

$$\boldsymbol{\phi}_i^{\text{target}} = \boldsymbol{\phi}_i - \frac{\nabla\varphi(\boldsymbol{\phi}_i)}{|\nabla\varphi(\boldsymbol{\phi}_i)|}\varphi(\boldsymbol{\phi}_i)$$

We applied our algorithm on the dynamic system to resolve collision effects as shown in Figure 8.4.

We also applied our algorithm on collisions between deformable objects. At each time step, the signed distance function of each deformed objects is updated.

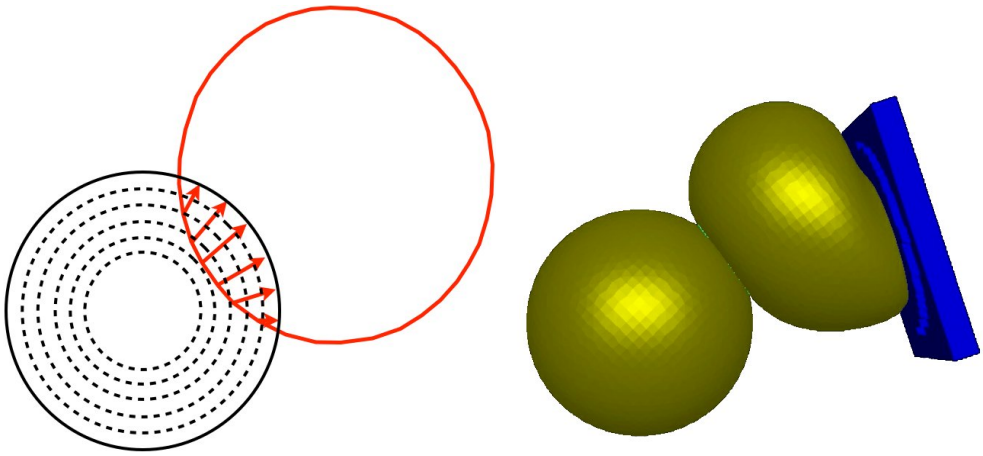


Figure 8.4: Collision detection and resolved collisions between a deformable sphere and an undeformable sphere. Left: collision detection and collision constraint target estimation. Black: levelset object; red: triangular mesh; arrows pointing from collided points to constraint targets. Right: colliding spheres example. A deformable sphere is attached to a brick in blue specified by Dirichlet boundary condition, and pushed to collide with a rigid sphere.

And the penetrated particles are detected between each of the objects and any other objects. Our results demonstrated a well resolved fast collision between two deformable objects even if one of them is very thin in Figure 8.5.

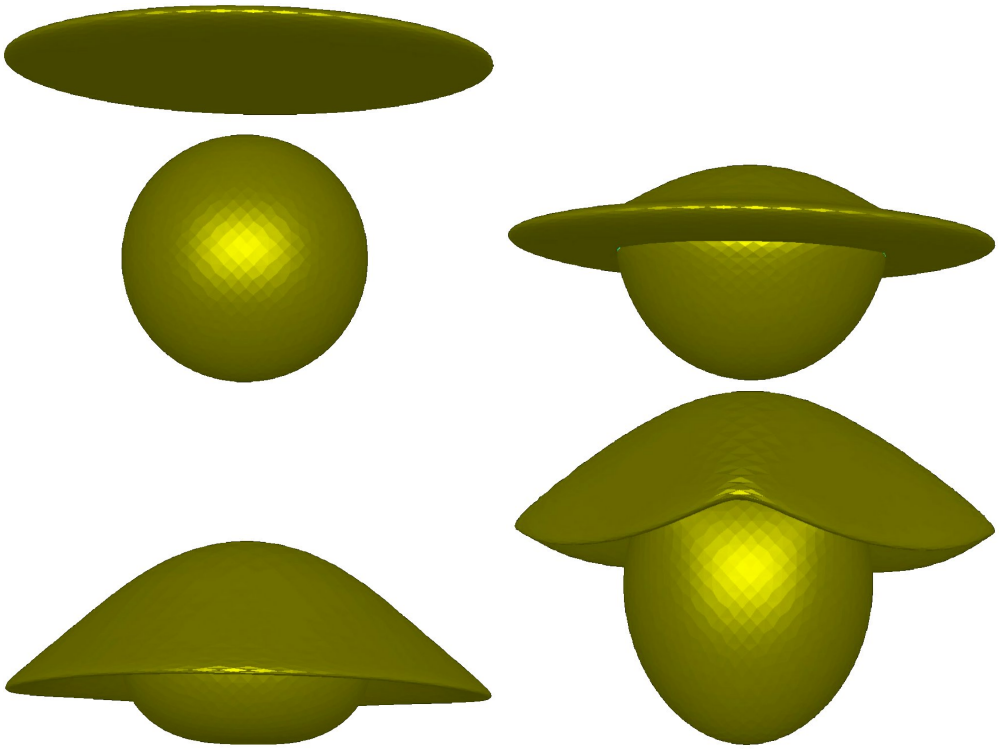


Figure 8.5: Two deformable objects colliding against each other.

CHAPTER 9

Conclusion and future works

We develop a multigrid framework for deformable solids simulation, which is efficient in resolving a wide range of materials including linear elastic materials from compressible to incompressible limit, as well as some nonlinear materials. The framework is efficient in resolving from a medium size problem to problems with millions of degrees of freedom. With certain simplification of the discretization, a simulation rate of near interactive speed is achieved for very high resolution problems with arbitrary irregular geometries. A second order accuracy is achieved by using a mixed finite element method. And the framework is extended to include soft constraint problems. The proposed framework is promising for further investigation, and the strength of the approach would be improved if some of the difficulties can be better resolved.

9.1 Efficient boundary treatment

The proposed multigrid methods using both finite difference method and finite element method, achieve an optimum efficiency with regular geometry and trivial boundary conditions. However, their efficiencies are suboptimal for arbitrary geometry problems. Although the boundary relaxation requires smaller and smaller amount of computational effort as the resolution increases. The efficiency of the overall methods would be improved with better boundary treatment. This issue

is related to two potential problems: the inaccuracy of the boundary coarsening and the lack of efficiency of boundary relaxations.

First, if the boundary conditions and nearby interior equations can be resolved in a coupled way, then the boundary relaxation is much more efficient. While the finite element method naturally resolves Neumann boundary conditions, efforts are made to eliminate Neumann boundary conditions in the finite difference method presented in section . Non-embedded Dirichlet boundary conditions in finite difference case are trivial boundary conditions, and can be relaxed in coupling with interior by simply fixing the values of the Dirichlet variables. In finite element case, the Dirichlet conditions are proposed in a week form, based on which we derive a simple approach to eliminated the Dirichlet boundary conditions using fundamental basis. Therefore, no additional error should be introduced to the boundary equations. However, the finite element case still requires extra boundary relaxations to achieve the expected multigrid convergence rate. This is related to the ill-conditioning of the reduced system due to the arbitrary cut cells.

Secondly, the discrepancy between coarse and fine discretizations is a source of inaccurate coarsening. In fact, we plan to investigate the combination of a Galerkin coarsening in the boundary band and a geometric coarsening in interior.

In the incompressible limit, the augmented linear elasticity problem degenerates to the Stokes' problem. Thus we are aiming at a continuation towards the Stokes' problem. Although the distributive relaxation naturally extends to the incompressible limit, the boundary relaxation does not. In fact, we eliminate pressure variable to generate an unaugmented system that is consistent with the unaugmented differential equation; however, for the Stokes' problem, there is no unaugmented system. Also, the unaugmentation process requires extra computa-

tion and memory cost. How to develop an efficient boundary relaxation without unaugmenting the system is of future interest.

9.2 Nonlinear hyperelastic solids

In the proposed solution for the co-rotational elasticity problem, we use the multigrid method to solve for a linearization of the nonlinear problem. This, however, is not a full linearization; thus, the nonlinear iteration does not converge efficiently especially with drastic rotations, when no accurate prior information about the rotation is available. We develop a full linearization for the co-rotational elasticity equation, which is similar to the derivation in Moita and Crisfield (1996). This will facilitate a more efficient solution for the co-rotational elasticity problem.

Given our experience with linear multigrid methods, we are also interested in investigating the nonlinear multigrid method (FAS). One of the essential differences in implementation lies in the fact that a nonlinear local relaxation requires a frequent update of local systems. Notice that each such update requires accessing neighboring solutions as well as an SVD, which will increase the cost of one relaxation sweep. The balance between the increasing of computation cost per iteration and the efficiency improvement of one multigrid cycle requires investigation be investigated. In nonlinear elasticity problems, the stable solutions between multiple levels might bifurcate under refinement, which is another source of inaccurate coarse grid approximations.

Klaas et al. (1999) proposed a general mixed formulation for hyperelastic models. We are interested in investigating realistic models with nonlinear incompressibility. More accurate biomechanics models for human flesh and tissues can be anisotropic Irving et al. (2004); Teran et al. (2005a). An algebraic coarsening

or line relaxation was developed for anisotropic elliptic equations. However, the algebraic coarsening will generate unstructured grids and lose the nice regularity, and the investigation of a line relaxation with non-grid aligned anisotropy direction is of interest.

9.3 Collision detection and stable solution

The proposed soft constraint method can be applied to resolve collision between deformable objects. We are interested in investigating efficient self-collision detection techniques that will not become a performance bottleneck, given the efficiency of the elasticity solver. Multi-resolution collision detection and response techniques (e.g. Otaduy et al. (2007)) would be expected to be the most compatible candidates.

The soft constraint method is an inexact constraint. To get better constraint conditions, stiffer constraints are required. Our method will not be efficient for very high stiffness problems. In fact, this issue has similar features to the instability issue of the original system under the incompressible limit. One may consider our treatment of a weak Dirichlet boundary condition in section 7.3 as an extreme case of a soft constraint. We are interested in investigating similar approaches.

The soft constraint solver potentially can be used for articulated character animation as well. Our method is implemented in a research-purpose implementation. We are interested in developing a user interface for animating articulated characters.

9.4 Adaptivity

Adaptive data structure economized up to 80% on the number of variables as indicated in Debunne et al. (1999); Sifakis et al. (2007b). However, due to different element sizes, the resulting numerics may not be well conditioned. In fact, the multigrid method is particularly friendly with adaptively refined regular grids. We are interested in investigating adaptive multigrid methods.

9.5 Parallel implementation for irregular models

Finally, although our initial investigation has demonstrated excellent potential for scaling on many-core platforms, a more principled investigation needs to assess the performance of our method on platforms with SIMD capability, and address a broader spectrum of constitutive behaviors and interacting geometries.

Bibliography

AlexanderLinke (2008). *Divergence-Free Mixed Finite Elements for the Incompressible Navier-Stokes Equation*. PhD thesis, University of Erlangen. 25

Andersson, J., Hutton, C., Ashburner, J., Turner, R., and Friston, K. (2001). Modeling geometric deformations in EPI time series. *Neuroimage*, 13(5):903–919. 4

Arbogast, T., Wheeler, M., and Zhang, N. (1996). A nonlinear mixed finite element method for a degenerate parabolic equation arising in flow in porous media. *SIAM Journal on Numerical Analysis*, 33(4):1669–1687. 25

Arnold, D., Brezzi, F., and Fortin, M. (1984). A stable finite element for the Stokes equations. *Calcolo*, 21(4):337–344. 26

Arnold, D., Falk, R., and Winther, R. (2000). Multigrid in H (div) and H (curl). *Numerische Mathematik*, 85(2):197–217. 27

Arnold, D. and Winther, R. (2002). Mixed finite elements for elasticity. *Numerische Mathematik*, 92(3):401–419. 25

Arnold, D. N. (1990). Mixed finite element methods for elliptic problems. *Computer Methods in Applied Mechanics and Engineering*, 82(1-3):281–300. 24

Babuška, I. and Suri, M. (1992). Locking effects in the finite element approximation of elasticity problems. *Numerische Mathematik*, 62(1):439–463. 25

Bank, R. (1996). Hierarchical bases and the finite element method. *Acta numerica*, 5:1–43. 20

Baraff, D. and Witkin, A. (1992). Dynamic simulation of non-penetrating flexible bodies. *SIGGRAPH Computer Graphics*, 26(2):303–308. 13, 14, 15, 23

Baraff, D. and Witkin, A. (1998). Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 43–54, New York, NY, USA. ACM. 12, 23

Barbič, J., da Silva, M., and Popović, J. (2009). Deformable object animation using reduced optimal control. *ACM Transactions on Graphics*, 28(3):1–9. 20

Barbič, J. and James, D. (2005). Real-Time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Transactions on Graphics*, 24(3):982–990. 14, 15, 20, 22

Barr, A. (1981). Superquadrics and angle-preserving transformations. *IEEE Computer graphics and Applications*, 1(1):11–23. 12

Barr, A. (1984). Global and local deformations of solid primitives. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, page 30. ACM. 12

Barzel, R., Hughes, J., and Wood, D. (1996). Plausible motion simulation for computer graphics animation. *Computer Animation and Simulation96*, pages 184–197. 3

Bedrossian, J., von Brecht, J., Zhu, S., Sifakis, E., and Teran, J. (2010). A Second Order Virtual Node Method for Poisson Interface on Irregular Domains. *Journal of Computational Physics*, 229:6405–6426. 108, 113, 115, 116

Belhachmi, Z. and Tahir, S. Mixed finite element discretization of some variational inequalities arising in elasticity problems in domains with cracks. *2004-Fez conference on Differential Equations and Mechanics Electronic Journal of Differential Equations, Conference 11, 2004*, page 3340. 25

Benzi, M. and Golub, G. (2005). A preconditioner for generalized saddle point problems. *SIAM Journal on Matrix Analysis and Applications*, 26(1):20–41. 26

Benzi, M., Golub, G., and Liesen, J. (2005). Numerical solution of saddle point problems. *Acta numerica*, 14:1–137. 23

Berkley, J., Turkiyyah, G., Berg, D., Ganter, M., and Weghorst, S. (2004). Real-time finite element modeling for surgery simulation: An application to virtual suturing. *IEEE Transactions on visualization and computer graphics*, pages 314–325. 4

Bischoff, J., Arruda, E., and Grosh, K. (2000). Finite element modeling of human skin using an isotropic, nonlinear elastic constitutive model. *Journal of Biomechanics*, 33(6):645–652. 4

Bochev, P., Dohrmann, C., and Gunzburger, M. (2007). Stabilization of low-order mixed finite elements for the Stokes equations. *SIAM Journal on Numerical Analysis*, 44(1):82–101. 26

Boland, J. M. and Nicolaides, R. A. (1984). On the stability of bilinear-constant velocity-pressure finite elements. *Numerische Mathematik*, 44(2):219–222. 26

Bolz, J., Farmer, I., Grinspun, E., and Schröoder, P. (2003). Sparse matrix solvers on the gpu: conjugate gradients and multigrid. pages 917–924. 8, 21, 27

Bonet, J. and Wood, R. (1997). *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press. 15, 17, 28

Börgers, C. and Widlund, O. (1990). On finite element domain imbedding methods. *SIAM Journal on Numerical Analysis*, 27(4):963–978. 17

Braess, D. and Ming, P. (2005). A finite element method for nearly incompressible elasticity problems. *Mathematics of Computation*, 74(249):25–52. 25

Brandt, A. (1973). Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems. In *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*, pages 82–89. Springer. 27

Brandt, A. (1977a). Multi-Level Adaptive Solutions. *Mathematics of computation*, 31(138):333–390. 6, 29, 56, 79

Brandt, A. (1977b). Multi-Level Adaptive Solutions. *Mathematics of computation*, 31(138):333–390. 27

Brandt, A. (1986). Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation*, 19(1-4):23–56. 27, 34

Brandt, A. (1994). Rigorous quantitative analysis of multigrid, I: constant coefficients two-level cycle with L 2-norm. *SIAM Journal on Numerical Analysis*, 31(6):1695–1730. 44

Brandt, A. and Dinar, N. (1978a). Multi-grid solutions to elliptic flow problems. In *Numerical methods for partial differential equations: proceedings of an advanced seminar*, page 53. Academic Press. 27

Brandt, A. and Dinar, N. (1978b). Multigrid solutions to elliptic flow problems. *Numerical methods for partial differential equations*, pages 53–147. 40, 47

Brenner, S. (2009). Fast Solvers for Mixed Finite Element Methods. *Mixed Finite Element Technologies*, pages 57–88. 26

Brenner, S., Li, F., and Sung, L. (2007). A locally divergence-free nonconforming finite element method for the time-harmonic Maxwell equations. *Mathematics of Computation*, 76(258):573. 25

Brezzi, F. and Fortin, M. (1991). *Mixed and hybrid finite element methods*. Springer-Verlag: New York. 24, 30, 38, 96

Brezzi, F., Marini, L., Micheletti, S., Pietra, P., Sacco, R., and Wang, S. (2005). Discretization of semiconductor device problems (I). *Handbook of Numerical Analysis*, 13:317–441. 26

Bridson, R., Fedkiw, R., and Anderson, J. (2005). Robust treatment of collisions, contact and friction for cloth animation. In *ACM SIGGRAPH 2005 Courses*, page 2. ACM. 24

Bridson, R., Marino, S., and Fedkiw, R. (2003). Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 28–36. Eurographics Association. 3

Briggs, W. L., Henson, V. E., and McCormick, S. F. (2000). *A multigrid tutorial (2nd ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. 27

Bro-Nielsen, M. et al. (1998a). VR simulation of abdominal trauma surgery. In *In Medicine Meets Virtual Reality 6*. Citeseer. 4

Bro-Nielsen, M., Inc, H., and Rockville, M. (1998b). Finite element modeling in surgery simulation. *Proceedings of the IEEE*, 86(3):490–503. 4

Cai, Z., Jones, J., McCormick, S., and Russell, T. (1997). Control-volume mixed finite element methods. *Computational Geosciences*, 1(3):289–315. 26

Cao, Z. (2003). Fast Uzawa algorithm for generalized saddle point problems* 1. *Applied Numerical Mathematics*, 46(2):157–171. 26

Capell, S., Burkhart, M., Curless, B., Duchamp, T., and Popovic, Z. (2007). Physically based rigging for deformable characters. *Graphical Models*, 69(1):71–87. 24

Capell, S., Green, S., Curless, B., Duchamp, T., and Popović, Z. (2002a). A multiresolution framework for dynamic deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 41–47. ACM. 14, 17, 21

Capell, S., Green, S., Curless, B., Duchamp, T., and Popovic, Z. (2002b). Interactive skeleton-driven dynamic deformations. *ACM Transactions on Graphics*, 21(3):586–593. 15, 24

Chen, Z., Cockburn, B., Jerome, J., and Shu, C. (1995). Mixed-RKDG finite element methods for the 2-D hydrodynamic model for semiconductor device simulation. *VLSI Design*, 3(2):145–158. 25

Choi, K.-J. and Ko, H.-S. (2005a). Stable but responsive cloth. In *ACM SIGGRAPH 2005 Courses*, page 1, New York, NY, USA. ACM. 3

Choi, M. and Ko, H. (2005b). Modal warping: Real-time simulation of large rotational deformation and manipulation. *IEEE Transactions on Visualization and Computer Graphics*, pages 91–101. 20

Cohen, M. (1992). Interactive spacetime control for animation. *ACM SIGGRAPH Computer Graphics*, 26(2):293–302. 23

Cotin, S., Delingette, H., and Ayache, N. (2000). A hybrid elastic model for

real-time cutting, deformations, and force feedback for surgery training and simulation. *The Visual Computer*, 16(8):437–452. 4

Cotin, S., Delingette, H., and Ayache, N. (2002). Real-time elastic deformations of soft tissues for surgery simulation. *Visualization and Computer Graphics, IEEE Transactions on*, 5(1):62–73. 16

Cotin, S., Delingette, H., Bro-Nielsen, M., Ayache, N., Clément, J., Tassetti, V., and Marescaux, J. (1996). Geometric and physical representations for a simulator of hepatic surgery. *Medicine Meets Virtual Reality: Health Care in the Information Age*, page 139. 15

Cutting, C., Oliker, A., Haring, J., Dayan, J., and Smith, D. (2002). Use of three-dimensional computer graphic animation to illustrate cleft lip and palate surgery. *Computer Aided Surgery*, 7(6):326–331. 4

Debunne, G., Cani, M.-P., Desbrun, M., and Barr, A. (2000). Adaptive simulation of soft bodies in real-time. In *Proceedings of the Computer Animation*, page 15, Washington, DC, USA. IEEE Computer Society. 13

Debunne, G., Desbrun, M., Barr, A., and Cani, M. (1999). Interactive multiresolution animation of deformable models. In *Eurographics Workshop on Computer Animation and Simulation*, volume 99, pages 133–144. Citeseer. 21, 159

Debunne, G., Desbrun, M., Cani, M., and Barr, A. (2001). Dynamic real-time deformations using space and time adaptive sampling. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 31–36. ACM. 14, 21

Desbrun, M. and Gascuel, M. (1996). Smoothed particles: A new paradigm for animating highly deformable bodies. In *Proceedings of the Eurographics workshop on Computer animation and simulation*, volume 96, pages 61–76. Citeseer. 18

Dick, C., Georgii, J., Burgkart, R., and Westermann, R. (2008). Computational steering for patient-specific implant planning in orthopedics. pages 83–92. 27

Dick, C., Georgii, J., and Westermann, R. (2010). A real-time multigrid finite hexahedra method for elasticity simulation using CUDA. Technical report, Computer Graphics and Visualization Group, Technische Universität München, Germany. 22, 27

Dongarra, J. and Kontoghiorghes, E. (2001). *Parallel numerical linear algebra*. Nova Science. 8, 21

Faloutsos, P., van de Panne, M., and Terzopoulos, D. (1997). Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):201–214. 13

Fung, Y. (1993). *Biomechanics: mechanical properties of living tissues*. Springer. 16

Gaspar, F., Gracia, J., Lisbona, F., and Oosterlee, C. (2008). Distributive smoothers in multigrid for problems with dominating grad-div operators. *Numerical linear algebra with applications*, 15(8):661–683. 27, 40

Georgii, J., Echtler, F., and Westermann, R. (2005). Interactive simulation of deformable bodies on GPUs. *Simulation and Visualization*, 2005:247–258. 22

Georgii, J. and Westermann, R. (2005). Mass-spring systems on the GPU. *Simulation Modelling Practice and Theory*, 13(8):693–702. 12

Georgii, J. and Westermann, R. (2006). A multigrid framework for real-time simulation of deformable bodies. *Computers and Graphics*, 30(3):408–415. 16, 27, 59

Georgii, J. and Westermann, R. (2008). Corotated finite elements made fast and stable. In *Proceedings of the 5th Workshop On Virtual Reality Interaction and Physical Simulation*. 16, 59

Gibson, S. and Mirtich, B. (1997). A survey of deformable modeling in computer graphics. *MERL, TR-97*, 19. 11

Girault, V. and Raviart, P.-A. (1979). *Finite element approximation of the Navier-Stokes equations*, volume 749 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin. 26

Gissler, M., Becker, M., and Teschner, M. (2006). Local constraint methods for deformable objects. In *Proceedings of the 3rd Workshop in VR Interactions and Physical Simulation (VRIPHYS)*, pages 1–8. Citeseer. 22

Goldenthal, R., Harmon, D., Fattal, R., Bercovier, M., and Grinspun, E. (2007). Efficient simulation of inextensible cloth. *ACM Transactions on Graphics*, 26(3):49. 3

Goodnight, N., Woolley, C., Lewin, G., Luebke, D., and Humphreys, G. (2003). A multigrid solver for boundary value problems using programmable graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 102–111, Aire-la-Ville, Switzerland. Eurographics Association. 27

Gopalakrishnan, J. and Tan, S. (2009). A convergent multigrid cycle for

the hybridized mixed method. *Numerical Linear Algebra with Applications*, 16(9):689–714. 27

Gortler, S. and Cohen, M. (1995). Hierarchical and variational geometric modeling with wavelets. In *Proceedings of the 1995 symposium on Interactive 3D graphics*. ACM. 21

Gourret, J., Thalmann, N., and Thalmann, D. (1989). Simulation of object and human skin formations in a grasping task. *ACM SIGGRAPH Computer Graphics*, 23(3):21–30. 4

Green, S., Turkiyyah, G., and Storti, D. (2002). Subdivision-based multilevel methods for large scale engineering simulation of thin shells. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 265–272. ACM. 27

Griebel, M., Oeltz, D., and Schweitzer, M. A. (2003). An algebraic multigrid method for linear elasticity. *SIAM Journal on Scientific Computing*, 25(2):385–407. 71, 81

Grinspun, E., Krysl, P., and Schröder, P. (2002). CHARMS: a simple framework for adaptive simulation. *ACM Transactions on Graphics*, 21(3):281–290. 21

Guo, Z., Peng, X., and Moran, B. (2006). A composites-based hyperelastic constitutive model for soft tissue with application to the human annulus fibrosus. *Journal of the Mechanics and Physics of Solids*, 54(9):1952–1971. 4

Hackbusch, W. and Trottenberg, U. (1982). Multigrid methods. 6, 27

Han, H. and Wu, X. (1998). A new mixed finite element formulation and the MAC method for the Stokes equations. *SIAM Journal on Numerical Analysis*, 35(2):560–571. 26, 98

Han, H. and Yan, M. (2008). A Mixed Finite Element Method on A Staggered Mesh for Navier-Stokes Equations. *Journal of Computational Mathematics*, 26(6):816–824. 26

Hansbo, P. and Larson, M. (2002). Discontinuous Galerkin methods for incompressible and nearly incompressible elasticity by Nitsche's method. *Computer methods in applied mechanics and engineering*, 191(17-18):1895–1908. 16

Harlow, F., Welch, J., et al. (1965). Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of fluids*, 8(12):2182–2189. 30

Hauth, M. and Strasser, W. (2004). Corotational simulation of deformable solids. In *Proc WSCG*, volume 12, pages 137–145. Citeseer. 16, 59

Heidelberger, B., Teschner, M., Keiser, R., Müller, M., and Gross, M. (2004). Consistent penetration depth estimation for deformable collision response. In *Proceedings of Vision, Modeling, Visualization VMV04, Stanford, USA*, pages 339–346. Citeseer. 22, 23

Hiptmair, R. (1997). Multigrid method for H (div) in three dimensions. *Electronic Transactions on Numerical Analysis*, 6:133–152. 27

Hsu, W. M., Hughes, J. F., and Kaufman, H. (1992). Direct manipulation of free-form deformations. *SIGGRAPH Computer Graphics*, 26(2):177–184. 12

Hughes, C., Grzeszczuk, R., Sifakis, E., Kim, D., Kumar, S., Selle, A., Chhugani, J., Holliman, M., and Chen, Y. (2007). Physical simulation for animation and visual effects: parallelization and characterization for chip multiprocessors. volume 35, page 231. ACM. 8, 21

Hughes, T. (1987). *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Prentice Hall. 76

Hutchinson, D., Preston, M., and Hewitt, T. (1996). Adaptive refinement for mass/spring simulations. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96*, pages 31–45, New York, NY, USA. Springer-Verlag New York, Inc. 13

Irving, G., Schroeder, C., and Fedkiw, R. (2007). Volume conserving finite element simulations of deformable models. In *ACM SIGGRAPH 2007 papers*, page 13, New York, NY, USA. ACM. 16, 24

Irving, G., Teran, J., and Fedkiw, R. (2004). Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 131–140. Eurographics Association. 16, 30, 157

Isaacs, P. M. and Cohen, M. F. (1987). Controlling dynamic simulation with kinematic constraints. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 215–224, New York, NY, USA. ACM. 23

James, D. and Fatahalian, K. (2003). Precomputing interactive dynamic deformable scenes. *ACM Transactions on Graphics*, 22(3):879–887. 20

James, D. and Pai, D. (1999). ArtDefo: accurate real time deformable objects. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 65–72. ACM Press/Addison-Wesley Publishing Co. 15

Kaasschieter, E., Frijns, A., and Huyghe, J. (2003). Mixed finite element mod-

elling of cartilaginous tissues. *Mathematics and Computers in Simulation*, 61(3-6):549–560. 25

Kaufmann, P., Martin, S., Botsch, M., and Gross, M. (2009a). Flexible simulation of deformable models using discontinuous Galerkin FEM. *Graphical Models*, 71(4):153–167. 16, 17

Kaufmann, P., Martin, S., Botsch, M., and Gross, M. (2009b). Flexible simulation of deformable models using discontinuous galerkin fem. *Graphical Models*, 71(4):153–167. 73

Kazhdan, M. and Hoppe, H. (2008). Streaming multigrid for gradient-domain operations on large images. *ACM Transactions on Graphics*, 27(3):1–10. 27

Klaas, O., Maniatty, A., and Shephard, M. (1999). A stabilized mixed finite element method for finite elasticity.:: Formulation for linear displacement and pressure interpolation. *Computer Methods in Applied Mechanics and Engineering*, 180(1-2):65–79. 26, 157

Koch, R. M., Gross, M. H., Carls, F. R., von Büren, D. F., Fankhauser, G., and Parish, Y. I. H. (1996). Simulating facial surgery using finite element models. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 421–428, New York, NY, USA. ACM. 14

Krysl, P., Grinspun, E., and Schröder, P. (2003). Natural hierarchical refinement for finite element methods. *International Journal for Numerical Methods in Engineering*, 56(8):1109–1124. 20

Kui, L., Liu, G., and Zienkiewicz, O. (1985). A generalized displacement method for the finite element analysis of thin shells. *International Journal for Numerical Methods in Engineering*, 21(12):2145–2155. 26

Lapeer, J., Gasson, D., and Karri, V. (2010). A hyperelastic finite element model of human skin for interactive real-time surgical simulation. *IEEE Transactions on Biomedical Engineering*, 99:1. 4

Lee, S.-H., Sifakis, E., and Terzopoulos, D. (2009). Comprehensive biomechanical modeling and simulation of the upper body. *ACM Transactions on Graphics*, 28(4):1–17. 4, 72

Lee, Y., Terzopoulos, D., and Waters, K. (1995). Realistic modeling for facial animation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 55–62, New York, NY, USA. ACM. 13

Lekien, F. and Marsden, J. (2005). Tricubic interpolation in three dimensions. *International Journal for Numerical Methods in Engineering*, 63(3):455–471. 84

Liu, A., Tendick, F., Cleary, K., and Kaufmann, C. (2003). A survey of surgical simulation: applications, technology, and education. *Presence: Teleoperators & Virtual Environments*, 12(6):599–614. 4

Losasso, F., Gibou, F., and Fedkiw, R. (2004). Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics*, 23(3):457–462. 79

Marescaux, J., Clément, J., Tassetti, V., Koehl, C., Cotin, S., Russier, Y., Mutter, D., Delingette, H., and Ayache, N. (1998). Virtual reality applied to hepatic surgery simulation: the next revolution. *Annals of Surgery*, 228(5):627. 4

Martin, S., Kaufmann, P., Botsch, M., Grinspun, E., and Gross, M. (2010). Unified simulation of elastic rods, shells, and solids. In *ACM SIGGRAPH 2010 papers*, pages 1–10, New York, NY, USA. ACM. 19

Metaxas, D. and Terzopoulos, D. (1992). Dynamic deformation of solid primitives with constraints. volume 26, pages 309–312, New York, NY, USA. ACM. 15, 23

Milliron, T., Jensen, R. J., Barzel, R., and Finkelstein, A. (2002). A framework for geometric warps and deformations. *ACM Transactions on Graphics*, 21(1):20–51. 12

Moita, G. and Crisfield, M. (1996). A finite element formulation for 3-D continua using the co-rotational technique. *International Journal for Numerical Methods in Engineering*, 39(22):3775–3792. 16, 157

Molino, N., Bao, Z., and Fedkiw, R. (2004). A virtual node algorithm for changing mesh topology during simulation. In *ACM SIGGRAPH 2004 Papers*, pages 385–392, New York, NY, USA. ACM. 72

Molino, N., Bridson, R., Teran, J., and Fedkiw, R. (2003). A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *12th Int. Meshing Roundtable*, pages 103–114. Citeseer. 17, 21

Müller, M., Dorsey, J., McMillan, L., Jagnow, R., and Cutler, B. (2002). Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, July*, pages 49–54. Citeseer. 2, 15, 59

Müller, M. and Gross, M. (2004). Interactive virtual materials. In *GI '04: Proceedings of Graphics Interface 2004*, pages 239–246, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. Canadian Human-Computer Communications Society. 16, 59

Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J. (2007). Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118. 23

Müller, M., Heidelberger, B., Teschner, M., and Gross, M. (2005a). Meshless deformations based on shape matching. *ACM Transactions on Graphics*, 24(3):471–478. 19

Müller, M., Keiser, R., Nealen, A., Pauly, M., Gross, M., and Alexa, M. (2004). Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pages 141–151, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association. 18

Müller, M., Teschner, M., and Gross, M. (2005b). Physically-based simulation of objects represented by surface meshes. In *Computer Graphics International, 2004. Proceedings*, pages 26–33. IEEE. 16, 72

Nealen, A., Müller, M., Keiser, R., Boxerman, E., and Carlson, M. (2006). Physically based deformable models in computer graphics. In *Computer Graphics Forum*, volume 25, pages 809–836. Wiley Online Library. 11, 16

Nedel, L. and Thalmann, D. (1998). Real time muscle deformations using mass-spring systems. In *Computer Graphics International, 1998. Proceedings*, pages 156–165. IEEE, IEEE Computer Society. 12

Ni, X., Garland, M., and Hart, J. (2004). Fair morse functions for extracting the topological structure of a surface mesh. pages 613–622. 27

Nicolaides, R. A. and Wu, X. (1996). Analysis and convergence of the MAC

scheme. II. Navier-Stokes equations. *Mathematics of Computation*, 65(213):29–44. 26

O'Brien, J. and Hodgins, J. (1999a). Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 137–146. ACM Press/Addison-Wesley Publishing Co. 16

O'Brien, J. F. and Hodgins, J. K. (1999b). Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 137–146, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co. 18, 30

Oosterlee, C. and Gaspar, F. (2008). Multigrid relaxation methods for systems of saddle point type. *Applied Numerical Mathematics*, 58(12):1933–1950. 27, 127

Otaduy, M., Germann, D., Redon, S., and Gross, M. (2007). Adaptive deformations with fast tight bounds. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 181–190. Eurographics Association. 21, 158

Pauly, M., Gross, M., and Kobbelt, L. P. (2002a). Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization '02*, VIS '02, pages 163–170, Washington, DC, USA. IEEE Computer Society. 18

Pauly, M., Keiser, R., Adams, B., Dutré, P., Gross, M., and Guibas, L. (2005). Meshless animation of fracturing solids. In *ACM SIGGRAPH 2005 Papers*, pages 957–964. ACM. 18

Pauly, M., Kobbelt, L., and Gross, M. (2002b). Multiresolution modeling of point-sampled geometry. *Technical Report*, 3(7):8. 21

Picinbono, G., Delingette, H., and Ayache, N. (2005). Nonlinear and anisotropic elastic soft tissue models for medical simulation. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1370–1375. IEEE. 14

Platt, D. and Fleischer, K. (1989). Heating and Melting Deformable Models (from Goop to Glop). In *Proceedings Graphics Interface'89*, pages 219–226. 18

Platt, J. (1992). A generalization of dynamic constraints. *CVGIP: Graphical Models and Image Processing*, 54(6):516–525. 12, 22

Quarteroni, A. and Valli, A. (1999). *Domain decomposition methods for partial differential equations*. Oxford University Press, USA. 17

Raghupathi, L., Grisoni, L., Faure, F., Marchal, D., Cani, M., and Chaillou, C. (2004). An intestinal surgery simulator: Real-time collision processing and visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 10(6):708–718. 4

Rappoport, A., Sheffer, A., and Bercovier, M. (1995). Volume-preserving free-form solid. In *Proceedings of the third ACM symposium on Solid modeling and applications*, pages 361–372. ACM. 12

Rivers, A. R. and James, D. L. (2007). Fastlsm: fast lattice shape matching for robust real-time deformation. In *ACM SIGGRAPH 2007 papers*, volume 26, page 82, New York, NY, USA. ACM. 19, 29, 72

Sederberg, T. and Parry, S. (1986). Free-form deformation of solid geometric models. *SIGGRAPH Computer Graphics*, 20(4):151–160. 12

Selle, A., Lentine, M., and Fedkiw, R. (2008). A mass spring model for hair simulation. *ACM Transactions on Graphics*, 27(3):1–11. 3, 12

Shi, L., Yu, Y., Bell, N., and Feng, W. (2006). A fast multigrid algorithm for mesh deformation. In *ACM SIGGRAPH 2006 Papers*, pages 1108–1117. ACM. 27

Sifakis, E., Der, K., and Fedkiw, R. (2007a). Arbitrary cutting of deformable tetrahedralized objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 73–80. Eurographics Association. 18

Sifakis, E., Neverov, I., and Fedkiw, R. (2005). Automatic determination of facial muscle activations from sparse motion capture marker data. In *ACM SIGGRAPH 2005 Papers*, pages 417–425, New York, NY, USA. ACM. 4, 23

Sifakis, E., Shinar, T., Irving, G., and Fedkiw, R. (2007b). Hybrid simulation of deformable solids. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 81–90, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association. 21, 72, 79, 159

Singh, K. and Fiume, E. (1998). Wires: a geometric deformation technique. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 405–414, New York, NY, USA. ACM. 12

Sørensen, T. and Mosegaard, J. (2006). An introduction to GPU accelerated surgical simulation. *Biomedical Simulation*, pages 93–104. 22

Stenberg, R. and Suri, M. (1996). Mixed *hp* finite element methods for problems in elasticity and Stokes flow. *Numerische Mathematik*, 72(3):367–389. 25

Sumner, R. and Popović, J. (2004). Deformation transfer for triangle meshes. In *ACM SIGGRAPH 2004 Papers*, pages 399–405. ACM. 3

Szekely, G., Brechbühler, C., Hutter, R., Rhomberg, A., and Schmid, P. (1998). Modelling of soft tissue deformation for laparoscopic surgery simulation. *Medical Image Computing and Computer-Assisted InterventionMICCAI98*, page 550. 4

Tejada, E. and Ertl, T. (2005). Large steps in GPU-based deformable bodies simulation. *Simulation Modelling Practice and Theory*, 13(8):703–715. 22

Teran, J., Blemker, S., Hing, V., and Fedkiw, R. (2003). Finite volume methods for the simulation of skeletal muscle. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 68–74. Eurographics Association. 16, 17

Teran, J., Sifakis, E., Blemker, S., Ng-Thow-Hing, V., Lau, C., and Fedkiw, R. (2005a). Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics*, pages 317–328. 4, 157

Teran, J., Sifakis, E., Irving, G., and Fedkiw, R. (2005b). Robust quasistatic finite elements and flesh simulation. pages 181–190. 14, 83

Terzopoulos, D. and Fleischer, K. (1988a). Deformable models. *The Visual Computer*, 4(6):306–331. 2, 13, 18

Terzopoulos, D. and Fleischer, K. (1988b). Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 269–278, New York, NY, USA. ACM. 13, 20, 27

Terzopoulos, D. and McInerney, T. (1996). Deformable models in medical image analysis: A survey. *Medical Image Analysis*, 1(2):91–108. 4

Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. (1987). Elastically deformable models. 21(4):205–214. 2, 13, 14, 16, 22, 30

Terzopoulos, D. and Qin, H. (1994). Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136. 13

Terzopoulos, D. and Waters, K. (1990). Physically-based facial modeling, analysis, and animation. *Journal of visualization and Computer Animation*, 1(2):73–80. 3, 12

Terzopoulos, D. and Witkin, A. (1988). Physically based models with rigid and deformable components. *IEEE Computer Graphics and Applications*, 8(6):41–51. 15

Thomaszewski, B., Pabst, S., and Blochinger, W. (2007). Exploiting parallelism in physically-based simulations on multi-core processor architectures. 21

Thomaszewski, B., Wacker, M., and Straßer, W. (2006). A consistent bending model for cloth simulation with corotational subdivision finite elements. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 107–116. Eurographics Association. 16

Trottenberg, U., Oosterlee, C., and Schuller, A. (2001). *Multigrid*. Academic Press, Inc., Orlando, FL, USA. 27, 40, 48, 55

Vassilev, T. and Spanlang, B. (2002). A mass-spring model for real time deformable solids. *Proceedings of East-West Vision*, 2002:149–154. 12

Vassilevski, P. and Lazarov, R. (1996). Preconditioning mixed finite element saddle-point elliptic problems. *Numerical linear algebra with applications*, 3(1):1–20. 26

Wang, J., Wang, Y., and Ye, X. (2009). A Robust Numerical Method for Stokes Equations Based on Divergence-Free $H(\mathrm{div})$ Finite Element Methods. *SIAM Journal on Scientific Computing*, 31(4):2784–2802. 25

Wieners, C. (2000). Robust multigrid methods for nearly incompressible elasticity. *Computing*, 64(4):289–306. 27

Wilhelms, J. and Van Gelder, A. (1997). Anatomically based modeling. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 173–180. ACM Press/Addison-Wesley Publishing Co. 4

Witkin, A. and Welch, W. (1990). Fast animation and control of nonrigid structures. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 243–252. ACM, ACM. 22, 23

Wittum, G. (1989). Multi-grid methods for Stokes and Navier-Stokes equations. *Numerische Mathematik*, 54(5):543–563. 27

Wittum, G. (1990). On the convergence of multi-grid methods with transforming smoothers. *Numerische Mathematik*, 57(1):15–38. 27

Wu, X., Downes, M., Goktekin, T., and Tendick, F. (2001). Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In *Computer Graphics Forum*, volume 20, pages 349–358. John Wiley & Sons. 13

Wu, X. and Tendick, F. (2004). Multigrid integration for interactive deformable body simulation. pages 92–104. Springer. 27

Ye, X. and Hall, C. (1997). A discrete divergence-free basis for finite element methods. *Numerical Algorithms*, 16(3):365–380. 25

Zhou, K., Doyle, J., and Glover, K. (1996). *Robust and optimal control*. Prentice Hall Englewood Cliffs, NJ. 20

Zhu, Y., Sifakis, E., Teran, J., and Brandt, A. (2010). An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Transactions on Graphics*, 29(2):1–18. 9, 123, 124, 125, 128