Multilevel, Subdivision-Based, Thin Shell Finite Elements: Development and an Application to Red Blood Cell Modeling

Seth Green

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

University of Washington

2003

Program Authorized to Offer Degree: Mechanical Engineering

UMI Number: 3111069

University of Washington

Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Seth Green

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

Co-Chairs of Supervisory Committee:

George M. Turkiyyah

Duane W. Storti

Reading Committee:

George M. Turkiyyah

Duane W. Storti

Paul E. Labossière

Date: 12/15/2003

University of Washington

Abstract

Multilevel, Subdivision-Based, Thin Shell Finite Elements: Development and an Application to Red Blood Cell Modeling

by Seth Green

Co-Chairs of Supervisory Committee:

Professor George M. Turkiyyah
Civil Engineering

Professor Duane W. Storti
Mechanical Engineering

This work focuses on efficient hierarchical, numerical simulation of the deformation of thin walled structures. We address the need to characterize and improve the performance of the subdivision thin shell finite element for practical applications in engineering design and analysis. The contribution of this document is to provide a thorough investigation of thin shell simulation using the subdivision shell element, focusing on novel element design and implementation, accurate boundary conditions, efficient multilevel solution strategies, and applications to the current engineering problem of blood cell membrane simulation.

We describe a unified framework for the simulation of thin bodies via hierarchical, rotation-free thin shell finite elements for meshes with both quadrilateral (Catmull-Clark scheme) and triangular connectivity (Loop's scheme). A corresponding planar beam element is also presented. We present an algorithm that exploits the hierarchical structure of subdivision surfaces to accelerate solution convergence. Our examples show that the run time of the algorithm presented scales nearly linearly with problem size.

We present a new method for enforcing boundary conditions within subdivision finite element simulations of thin shells. The proposed framework is demonstrated to be second order accurate for simply-supported and clamped boundary conditions.

We demonstrate the application of these techniques for the numerical simulation of red blood cell models. Mechanical models of human red blood cells are a necessary component for microstructural simulation of blood flow for artificial organ design. Two specific simulations are demonstrated for constant-volume deformation of a red blood cell: micropipette aspiration and point load application.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table Number Page

ACKNOWLEDGMENTS

I would like to thank all of the members of my supervisory committee for their assistance and insight. Special thanks is deserved by the co-chairs of my committee, George Turkiyyah and Duane Storti. Duane has stood by me since the very beginning and helped me form my ideas into this dissertation. George has guided my work and helped to define the path I took to fulfilling my degree; without his support, my work would not have come to fruition.

Thanks to Paul Labossière, the newest member of my committee, who's genuine interest and unflagging dedication inspired me to carry on in times of success and trouble alike. Thanks to Tom Duchamp, a pioneer in subdivision surface representation, who first introduced me to subdivision analysis techniques and who's wisdom helped guide my path and who's judgement was unfailingly correct. Thanks to Mark Ganter for sparking my interest in computer graphics, and who has been with me since the very beginning of my graduate career. Thanks to the members of the Sangria Project who I have worked with, especially Jim Antaki, for his advise and expertise in blood cells. Thanks also goes to my friends and co-workers Cole Brooking, Rob Blanding, and Scott Johnston with whom I have had the fortune of working with; best of luck to you in your own pursuits.

Lastly special thanks goes to my family, who have watched me grow from baby to child, to high school graduate, to college graduate, to holder of a master's degree, and now finally to a doctor of engineering. I could not have come this far without your love and support.

This work was generously supported by NSF's Information Technology Research program, grant number ACI-0086093. I would also like to thank the Ford Motor Company and The Boeing Company for fellowship assistance during my studies.

DEDICATION

To my parents

Chapter 1

INTRODUCTION

1.1 Context

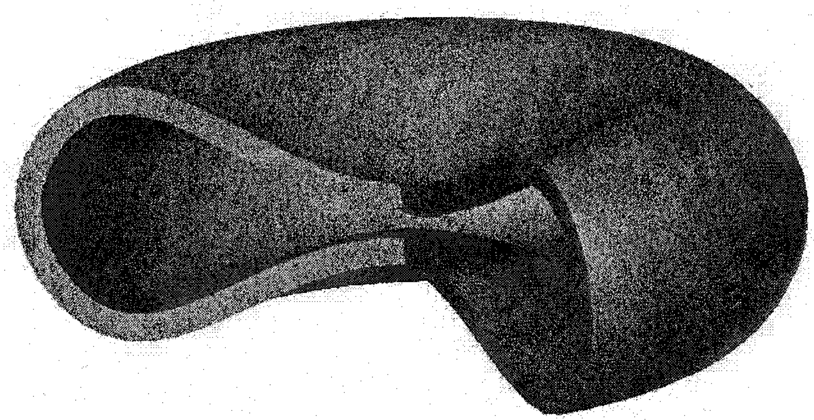


Figure 1.1: Red Blood Cell Membrane Model

This work focuses on efficient hierarchical, numerical simulation of the deformation of thin walled structures, along with the novel techniques, methods, data structures and algorithms developed for the task. Thin walled structures, known as thin shells, are those bodies that have a thickness which is small compared to the entire extent, allowing for specialized theories of deformation. Household examples of bodies that are well modeled as thin shells include pie tins or teapots. Shapes such as these are found in many biological and engineered systems. They are analyzed in many contexts including design of automotive body panels [18], the deployment of air bags [16], stresses in paved road surfaces [21], and the flow of human blood cells [52, 51]. Figure 1.1 shows the membrane of a red blood cell modeled as a thin shell. Accurate modeling of the deformation of curved shells, and their two-dimensional counterpart, curved beams, are of great benefit to engineering design.

Standard numerical shell formulations scale poorly and computational expense grows quickly as simulation accuracy increases [32, 21]. The most efficient algorithms are needed to simulate thin walled structures accurately as part of the engineering design cycle. Furthermore a conscientious representation of boundary conditions is required for accurate results. We propose a multilevel subdivision-based thin shell finite element and related solution algorithms as an efficient numerical simulation scheme for simulating the deformation of thin walled bodies. Specifically subdivision elements show great promise in modeling the membrane of blood cells in the human blood stream; their compact nature and robust handling of large deformations makes them natural candidates for this application.

1.2 Research Objectives

This study addresses the need to characterize and improve the performance of the subdivision thin shell element for practical applications in engineering design and analysis. Our objective is to extend the state of the art in subdivision-based thin shell modeling in four key areas:

- *A thorough investigation and implementation of a multilevel subdivision-based thin shell finite element and the corresponding 2D beam element.* Previous studies have shown the effectiveness of the subdivision thin shell element. This study is the first to present a unified framework for representing quadrilateral and triangular subdivision elements in a multilevel hierarchy. Included in this document is a complete classification of element types including the algorithms, data structures, and evaluation techniques needed for implementation.

- *Development of accurate constraints for solving boundary value problems.* Accurate simulations require accurate representations of boundary conditions. This work presents a scheme for applying boundary conditions that is second order accurate.

- *Development of efficient solution schemes.* Standard numerical shell formulations scale poorly with increased simulation accuracy. This document describes a multi-level solution algorithm for efficient accurate solution of thin shell boundary value problems.

- *Application of this work to a novel problem.* Accurate modeling of human red blood cells is an important problem in engineering and medicine. This document presents preliminary results of mechanical simulations involving red blood cells, and describes the simulation process in detail.

This work brings together knowledge from several separate fields including the theory of thin shells [55, 44], finite element methods [65, 36, 8, 12], subdivision surfaces [17, 59, 3, 20], multiresolution methods [21, 48, 63, 14], and biological modeling [30, 52, 51] with the goal of creating an efficient numerical solver for accurate, mechanical simulation of thin walled bodies. In addition, the formulations described involve only displacement degrees of freedom, which allows for simple integration with other solution techniques and simulations, including fluid solvers.

1.3 Organization of the Document

This document is organized as follows. Background information on the topics of thin shells, subdivision surfaces, continuum mechanics, the finite element method and multiresolution techniques is provided for the reader in Chapter 2.

Chapter 3 describes the development of a subdivision beam element, demonstrating the utility of the subdivision basis functions in engineering analysis and serves as a reduced framework for understanding the concepts inherent to subdivision thin shell analysis.

Chapter 4 describes a novel hierarchical rotation–free thin shell finite element for meshes with both quadrilateral and triangular connectivity. The hierarchical nature of the element allows it to be integrated into a multilevel solver to accelerate solution convergence. This chapter provides an introduction to thin shell analysis and a discussion of current approaches to general shell finite elements, and rotation-free formulations in particular.

Chapter 5 describes the application of constraints to subdivision elements for the solution of boundary value problems. This constraint framework is demonstrated to be second order accurate for several commonly accepted benchmark problems.

Chapter 6 describes an algorithm that exploits the hierarchical, multilevel structure of subdivision surfaces to accelerate the convergence of solution strategies for thin shell simulations. Favorable timing and scaling results are demonstrated for a variety of problems.

Chapter 7 describes the application of subdivision element techniques to the mechanical simulation of human blood cells. These studies focus on recreating two standard experiments involving blood cells: micropipette aspiration and point load deformation. Preliminary results are discussed.

Chapter 8 describes novel applications and demonstrations of the subdivision element including buckling and vibrational analysis. These demonstrations show the applicability of the subdivsion element to a variety of engineering analyses.

Chapter 9 summarizes this study, along with conclusions and recommended areas for future work.

Chapter 2

BACKGROUND

2.1 Thin Shells

(a) Initial Configuration (b) Deformed Configuration

Figure 2.1: Distributor Cap Under Load

The theory of thin shells is a mechanical theory, a model of physical behavior which relates the motion of a body to the forces acting upon it, and characterizes its internal state and deformation. The goal of thin shell analyses are to explain and predict the performance of thin walled bodies under load. For design purposes, performance includes measures of displacement, internal forces, deformation, damage prediction, etc. Figure 2.1 shows the initial and deformed configurations of an elastic distributor cap model under a loading condition illustrated by arrows. Like all engineering models, shell theory is an idealization which strives to relate the essential features of complicated physical phenomena to a small number of state variables. The utility of thin shell analysis has been demonstrated in a many contexts over significant ranges of scale; from modeling stadium roofs $[O(10^4)m^2]$, to cells in the human blood stream $[O(10^{-10})m^2]$.

2.2 Subdivision Surfaces

Subdivision surfaces have become widely used geometric representations of general curved three-dimensional boundary models and thin shell objects. Subdivision surface representations have been shown to be effective for use in many stages of the modeling process including general modeling [23], interrogation [61], reconstruction [38], shape editing systems [66] and more. Their compactness, generality and hierarchical structure have provided effective mechanisms to represent three-dimensional geometry in a robust and computationally efficient manner. All of the three-dimensional bodies illustrated in this work are modeled as subdivision surfaces. More recently, the shape functions used in the subdivision surfaces framework have shown to be an appropriate and effective basis for thin shell simulation [17]. By construction the subdivision shape functions provide the necessary C^1 continuity requirements (and H^2 integrability) for representing the solution of the fourth-order equilibrium equations governing the behavior of thin shells [64]. Subdivision model descriptions also provide the rather elegant property of representing both the geometry and the physics of the deformation using the same mathematical description.

Subdivision surfaces are defined by an initial coarse mesh known as the control mesh. This initial mesh is refined repeatedly (both geometrically and topologically) by rules defined by the chosen subdivision scheme. Repeated refinements lead to a hierarchy of increasingly refined models which approach a limit surface. Figure 2.2 shows a model hierarchy in increasing stages of refinement.

The hierarchical nature of subdivision surfaces makes them a natural candidate for multiresolution simulations. At each level, the subdivision description provides a built in discretization scheme: the topology of the control mesh. As the mesh is subdivided, new degrees of freedom are introduced which can be used to refine not only the geometry of the

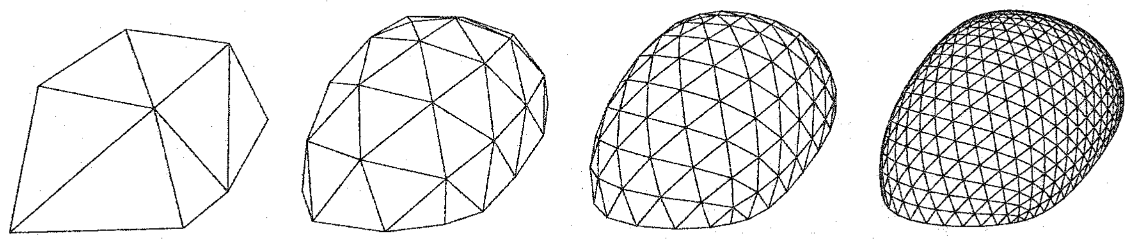


Figure 2.2: Subdivision Refinement Hierarchy

model, but its discretization as well. As each new refinement is derived from the previous level, a natural hierarchical description of a model is created. This hierarchy can be leveraged to great advantage in numerical simulation. A novel, hierarchical solution method for thin walled bodies described by subdivision surfaces is presented in Chapter 6.

The first publications describing subdivision surfaces were put forth by Catmull and Clark [15] and Doo and Sabin [25], both in the late 1970's. In the late 1980's, Loop [42] investigated the triangle-based scheme that bears his name today. These schemes produced visually smooth surfaces, but it was not until many years later that the smoothness properties of these schemes were proven rigorously [66].

The exact manner in which refinement takes place is defined by the chosen subdivision scheme, and many have been put forth and studied in current literature. Any number of scheme definitions could be imagined with some of greater importance than others. The Siggraph 2000 Subdivision Surface course notes [20] lists a number of properties which are desirable for subdivision schemes:

- Efficiency
- Compactness
- Affine Invariance
- Continuity

Most common subdivision schemes are stationary, meaning that the definition is independent of refinement level. Subdivision schemes generally consist of two fundamental steps. First a topological step produces a refined mesh by introducing new faces, vertices and edges into the control mesh. Next a geometric step repositions the vertices of the refined mesh by averaging the vertex locations of the control mesh. The scheme is applied recursively by treating the newly refined mesh as the control mesh of a yet finer level of subdivision and applying the topological and geometric refinement steps to it. If the scheme is well formed, the refined meshes converge to a smooth surface. Most subdivision schemes are designed to produce a limit surface which smoothly approximates (but not necessarily interpolates) the initial control mesh in shape.

2.2.1 Subdivision Refinement

We may refer to the vector of points representing the positions of the vertices of the control mesh after j levels of subdivision as $\mathbf{p}^j$, where $j = 0$ represents the control mesh itself. The positions of the vertices at level $j+1$ are given by the formula

$$\mathbf{p}^{j+1} = \mathbf{S}_j^{j+1}\mathbf{p}^j. \tag{2.1}$$

The limit surface is defined to be the dense point set defined by the limit of an infinite number of subdivisions. As each subdivision involves multiplying by the subdivision matrix, the eigen-properties of S determines the behavior of the scheme. In practice the subdivision matrix is not explicitly constructed; instead subdivision is applied locally using a weighted stencil to define new point locations.

Though the number of possible subdivision schemes is endless, a handful of schemes have been well studied and find widespread use today. Two of the most popular schemes are Loop's scheme for triangular control meshes [42] and the Catmull-Clark scheme for quadrilateral control meshes [15]. Each of these schemes produce four new triangles or quadrilaterals respectively for each step of subdivision refinement. These schemes are designed to replicate tensor product spline geometry in areas of regular connectivity, that is in areas of the mesh where the local topology can be tiled to cover an infinite plane.

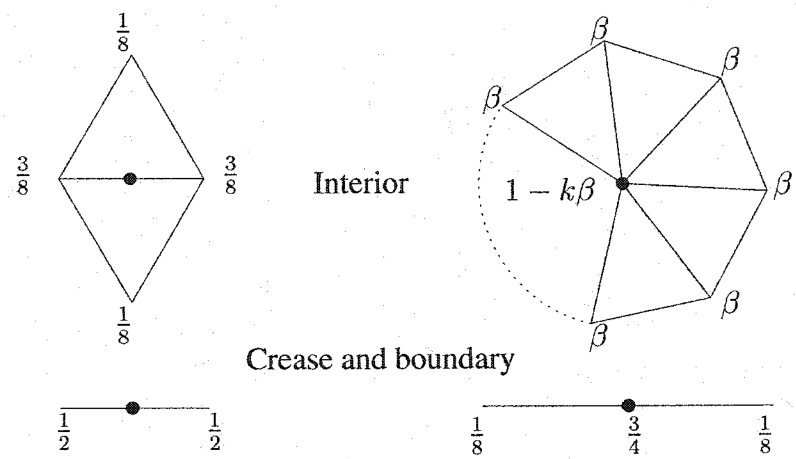


Figure 2.3: Loop Masks

Both Loop's and Catmull-Clark's schemes proceed by dividing each face of a mesh into four smaller faces. Two kinds of vertices result from this process, *odd* vertices are newly inserted by the subdivision process, while *even* vertices correspond directly to a vertex at the previous level of subdivision. The geometric positions of the even and odd vertices after subdivision are calculated by averaging positions from the previous step, as shown by the weighting stencils in Figures 2.3 and 2.4 which are adapted from [20]. The variable k is the integer valence of the affected vertex and β and γ are a functions of k given in Equations 2.2 and 2.3 for Loop and Catmull-Clark surfaces respectively. Excellent reviews of the Loop and Catmull-Clark subdivision schemes may be found in [59, 58] and [20]. Each of these schemes were constructed with an additional property in mind: they are designed so that the limiting sequence of subdivisions converge to a known surface described by a set of basis polynomials. These basis polynomials will provide the basic framework for the finite element analysis of subdivision surfaces.

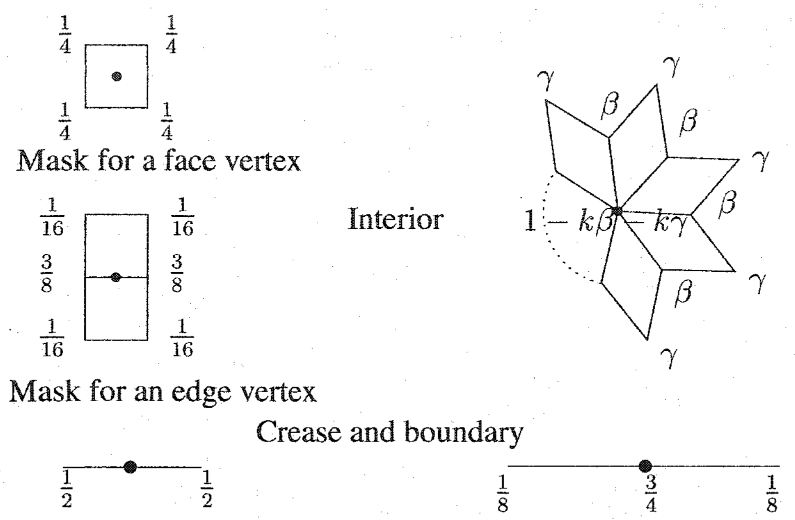


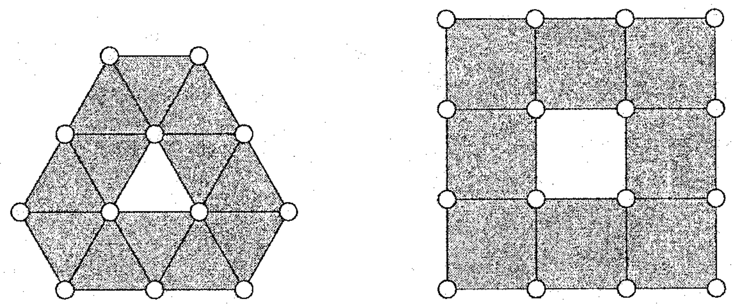
Figure 2.4: Catmull-Clark Masks

$$\beta = \frac{1}{n}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}cos\left(\frac{2\pi}{n}\right)\right)^2\right) \tag{2.2}$$

$$\beta = \frac{3k}{2} \qquad\qquad \gamma = \frac{k}{4} \tag{2.3}$$

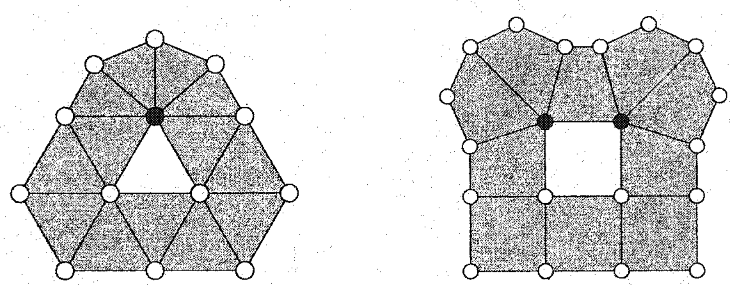
2.2.2 *Evaluation at Arbitrary Parameter Values*

In areas of regular connectivity, both Loop's, and Catmull and Clark's schemes reduce (by design) to a limit surface defined by tensor product polynomials. Any points lying within a regular region may be determined directly through evaluation of the spline basis functions without recourse to subdivision. Those areas of the mesh which do not posses regular connectivity are called extraordinary, and are more difficult to evaluate. Regular connectivities for triangular and quadrilateral meshes are shown in Figure 2.5. Irregular connectivities are shown in Figure 2.6. Nodes with irregular valences are denoted by filled black circles.



(a) Triangular (b) Quadrilateral

Figure 2.5: Regular Connectivity



(a) Triangular (b) Quadrilateral

Figure 2.6: Irregular Connectivity

Stam [58, 59] generalized the subdivision process to allow the efficient evaluation of limit surfaces at arbitrary parameter values in extraordinary regions. He has presented the application of his approach to both Loop's triangular scheme and Catmull-Clark's quadrilateral scheme. Previously limit surfaces were evaluated by a brute force approach of repeated subdivision. While this approach may suffice for graphics, it is insufficient (or at least quite inconvenient) for using the subdivision basis functions within a finite element analysis framework.

Each subdivision step reduces the area of the irregular regions where the limit surface cannot be directly evaluated. Any point that is not itself an extraordinary point will eventually lie within a regular region if enough subdivisions are performed; however, the number of subdivision steps required to evaluate point at a given parameter location can be arbitrarily large.

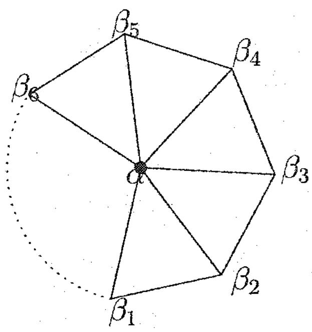
Subdivision may be though of as applying a linear operator, the subdivision operator, to the control mesh vertices of the previous level. By examining the region that always consists of the neighborhood of a point of interest, the subdivision operator will remain constaint and exponentiation of the local subdivision operator may be used to calculate a set of points which form a regular neighborhood around the point of interest. If the eigenstructure of the local subdivision operator is known, it may be diagonalized allowing for efficient exponentiation. The eigendecomposition is numerically unstable, but by applying spectral techniques, an algebraic form for the eigenstructure can be found. The result is that the basis functions for faces with vertices of arbitrary valence are always a linear combination of the canonical basis functions that describe regular patches.

Stam's approach is defined for meshes in which all faces have at most one extraordinary point. A novel extension of this algorithm extended to faces with more that one extraordinary point is described in Section 4.7.

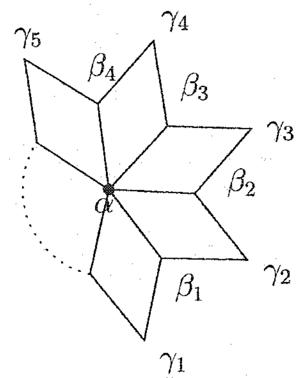
2.2.3 Limit Positions and Tangents

Neither Loop nor Catmull-Clark surfaces are formally C^2 continuous at extraordinary vertices. At these isolated points the spline basis functions no longer define the surface. A great effort has gone into proving continuity, differentiability and integrability properties at these points in the subdivision literature. In lieu of evaluating basis functions, the limit position and tangent plane of an extraordinary point may be written as a linear combination of the values of neighboring control point vertices. Limit normals may be easily calculated from the tangent plane.

The limit position and two vectors which define the limit tangent plane may be evaluated with the masks illustrated in Figure 2.7. These masks are similar to those used for subdivision, but the vertices may not be weighted equally. The mask weights come from an eigen-analysis of the local subdivision matrix. The values of α, β and γ for limit analysis are derived algebraically in [59] and [58].



(a) Loop Limit Mask



(b) CC Limit Mask

Figure 2.7: Limit Masks

2.3 Continuum Mechanics

Continuum theories idealize that matter is continuous on all scales, ignoring the atomic structure of matter. Attempting to predict the motion of a body by computing the behavior of each of it's molecules would not only be intractable (a single kilogram of iron contains $O(10^{25})$ iron atoms), but wholly unnecessary for most design and analysis purposes. Continuum theory relates the following three fundamental notions to provide a full description of motion for a deformable body:

1. A law of motion

2. An equation of constitution

3. A measure of deformation

The law of motion relates external and internal forces acting on a body to local accelerations. The equation of constitution, also known as the material law, describes how internal forces arise as the body changes shape. Lastly the measure of deformation relates local change in shape to changes in position of nearby points inside the body. If the body is elastic (discussed in Section 2.3.3) then a strong formulation of energy methods may be used to describe motion.

2.3.1 *Motion and internal forces*

The motion of a rigid body is related to the action of external forces by direct application of Newton's laws. Yet not all bodies are rigid; they may change shape under applied loading, or even crack and separate. To explain the behavior of non-rigid bodies it is necessary to define a notion of forces internal to a body and relate them to those that act externally.

External forces may be described by a vector field of forces acting over the domain of the body. Internal forces instead characterize the forces acting within a neighborhood of each body point. For each point in the body, there is a unique value of force in every direction. These forces are conceptualized by the stress field $\boldsymbol{\sigma}$, a tensor quantity which assigns a unique force to every point in the body in every direction.

Cauchy's law of motion [44] relates the internal and external forces of a point in a body to its acceleration.

$$\nabla \cdot \boldsymbol{\sigma} + \rho \mathrm{b} = \rho \frac{d\mathbf{v}}{dt}. \tag{2.4}$$

It is important to note that by Equation 2.4, it is the divergence of the stress field, not the absolute value that gives rise to motion in a body. A body subjected to hydrostatic stresses (stresses that are everywhere constant) undergoes no change in motion due to its internal state of stress; a marble dropped to the bottom of the ocean experiences stress due to the pressures pushing on it, but the stress field will be constant and hence results in no motion. If the divergence of the stress field is identically zero everywhere and density is constant, Equation 2.4 reduces to the familiar $f = ma$. If the body is in static equilibrium, the right hand side of Equation 2.4 is zero, and a relation between the applied forces and internal stresses may be found directly.

2.3.2 *Deformation*

Strain is a local measure of deformation in a body, as compared to a previous reference state. There are many ways in which strain can be defined; the important quality about a strain measure is that it quantifies some local measure of stretching. For a one-dimensional member, strain is easily described as the normalized change in length from l_0 to l_n for an arbitrary small bit of material.

$$\epsilon_{1D} = \frac{l_n - l_0}{l_0} \tag{2.5}$$

Our simple measure of strain (Equation 2.5) is difficult to compute in three dimensions. The scalar Lagrangian strain is defined to be:

$$\epsilon = \frac{1}{2}\left(\frac{l_n^2 - l_0^2}{l_0^2}\right) \tag{2.6}$$

This measure appears more complicated than it's one-dimensional cousin, but it is actually a simpler quantity to compute in three dimensions. By design, Equations 2.5 and 2.6 agree if the deformations involved are modest. Like stress, strain is not a vector quantity, as it relates the stretch between a point and its local neighborhood. The strain tensor ϵ operates such that $\epsilon = \mathbf{n} \cdot \epsilon \mathbf{n}$ for any unit direction vector $\mathbf{n}$.

It is useful to parameterize points on the body $\mathcal{B}$ as a function of generalized coordinates θ^i. Let $\mathbf{r}$ be points in the "rest state" and $\tilde{\mathbf{r}}$ be points in a deformed configuration, respectively.

$$\mathbf{r} = \mathbf{r}(\theta^1, \theta^2, \ldots \theta^n) \qquad \tilde{\mathbf{r}} = \tilde{\mathbf{r}}(\theta^1, \theta^2, \ldots \theta^n). \tag{2.7}$$

For convenience a local set of basis vectors may be chosen to be the covariant derivatives of position

$$\mathbf{g}_i = \mathbf{r}_{,i} = \frac{\partial \mathbf{r}}{\partial \theta^i} \qquad \tilde{\mathbf{g}}_i = \tilde{\mathbf{r}}_{,i} = \frac{\partial \tilde{\mathbf{r}}}{\partial \theta^i}. \tag{2.8}$$

Differential changes in position may be written as

$$d\mathbf{r} = \frac{\partial \mathbf{r}}{\partial \theta^i} d\theta^i = \mathbf{g}_i d\theta^i \qquad d\tilde{\mathbf{r}} = \frac{\partial \tilde{\mathbf{r}}}{\partial \theta^i} d\theta^i = \tilde{\mathbf{g}}_i d\theta^i. \tag{2.9}$$

Which leads to formulas for local squared lengths:

$$l_0^2 = d\mathbf{r} \cdot d\mathbf{r} = d\theta^i g_{ij} d\theta^j \qquad l_n^2 = d\tilde{\mathbf{r}} \cdot d\tilde{\mathbf{r}} = d\theta^i \tilde{g}_{ij} d\theta^j, \tag{2.10}$$

where

$$g_{ij} = \mathbf{g}_i \cdot \mathbf{g}_j \qquad \tilde{g}_{ij} = \tilde{\mathbf{g}}_i \cdot \tilde{\mathbf{g}}_j. \tag{2.11}$$

The strain tensor now takes on the simple definition of

$$\epsilon_{ij} = \frac{1}{2}(\tilde{g}_{ij} - g_{ij}) \tag{2.12}$$

in the undeformed coordinate system.

2.3.3 *Constitution*

Constitutive relations define the behavior of a material under load, and serve to differentiate the properties of one material from another. Cauchy's law of motion 2.4 explains how stress affects motion, but gives no explanation of how it arises. The axiom of locality states that stresses occur as the result of (and only of) corresponding deformations of the body. The notion of deformation captures the difference between the current configuration of a body and some previous shape. This idea is completely general; however it is an especially enlightening description for a solid body where changes in shape are easily compared to a previous known configuration of the body (in contrast to a fluid, which may deform endlessly). K. Hjelmstad [36] eloquently describes this principle as:

> One of the fundamental hypotheses underlying the modeling of constitutive behavior is that cause and effect between force and deformation occurs only at the local level. This simple hypothesis is not provable, and is the subject of great debate. It is however the result of centuries of observation. Insofar as it leads to useful results, it has been embraced by the engineering community.

The notion that stress depends upon strain must be clarified. For a variety of materials stresses depend not only on the current value of strain at a point, but also on it's rate of change and past history. A material is classified as elastic if stress is a function of the value of strain only, and this document will be restricted in scope to such materials. If stresses are linearly related to strains and the material is isotropic (material properties are the same in all directions), then the constitutive relation will have only 2 degrees of freedom and stress may be related to strain as

$$\boldsymbol{\sigma} = \lambda\,(\mathrm{tr}\boldsymbol{\epsilon})\mathbf{1} + \mu\boldsymbol{\epsilon} \tag{2.13}$$

where λ and μ are the Lame constants of the material [44] and tr denotes the trace of a tensor and $\mathbf{1}$ is the unit tensor. These quantities can be related to the more familiar Dilational (Poisson's) Ratio and Elastic (Young's) Modulus by a bit of algebra. If an elastic constitutive law is not linear, then stress may be written as a function of strain, $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\epsilon})$.

Numerical methods used in solid mechanics often approximate stress by the Taylor series expansion

$$\boldsymbol{\sigma}(\boldsymbol{\epsilon} + \delta\boldsymbol{\epsilon}) = \boldsymbol{\sigma}(\boldsymbol{\epsilon}) + \left.\frac{\delta\boldsymbol{\sigma}}{\delta\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}} \delta\boldsymbol{\epsilon} + O(\delta\boldsymbol{\epsilon}^2). \tag{2.14}$$

The variational derivative of stress with respect to strain is known as the tangent constitutive modulus, and is denoted by the symbol C defined in Equation 2.15.

$$\mathsf{C} = \frac{\delta\boldsymbol{\sigma}}{\delta\boldsymbol{\epsilon}}. \tag{2.15}$$

In three-dimensional analysis, $\boldsymbol{\sigma}$ and $\boldsymbol{\epsilon}$ will each be 3x3 second order symmetric tensor quantities. The tangent material modules is a fourth order tensor quantity of size 3x3x3x3.

2.3.4 Energy and Virtual Work

Stress and strain can be defined independently of one another, but it is necessary to pick complementary quantities such that the tensor scalar product (contraction) of the strain and stress tensors yields energy. For large strains, the Green/Lagrange strain tensor and the second Piola-Kirchhoff stress tensors are work complementary [44]. For infinitesimal strains, the Cauchy and the second Piola-Kirchhoff stress tensors are indistinguishable.

Equilibrium problems in solid mechanics may be formulated using the general principle of virtual work as stated in Equation 2.16.

$$\delta\Pi = \int_V \delta\boldsymbol{\epsilon} : \boldsymbol{\sigma} - \delta\mathbf{u} \cdot \mathbf{f}\, dV = 0 \quad \forall \delta\mathbf{u}, \delta\boldsymbol{\epsilon}. \tag{2.16}$$

This relation balances the work done by external forces $\mathbf{f}$ moving through virtual displacements $\delta\mathbf{u}$ with the internal work done by the internal stresses $\boldsymbol{\sigma}$ and their corresponding virtual strains $\delta\boldsymbol{\epsilon}$. If these quantities are equal *for all* virtual displacements and virtual strains then the system will be in equilibrium everywhere.

For an elastic body, strain will be a function of displacement and stress will be a function of strain via Equation 2.17

$$\boldsymbol{\epsilon} = \boldsymbol{\epsilon}(\mathbf{u}) \tag{2.17}$$

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\epsilon}). \tag{2.18}$$

The elastic statement of virtual work simplifies to

$$\delta\Pi(\mathbf{u}) = 0 \quad \forall \delta\mathbf{u}. \tag{2.19}$$

As strain throughout the body is a function of displacement, the $\forall \delta\mathbf{u}$ requirement is a necessary and sufficient condition for equilibrium.

2.4 The Finite Element Method

Finite element techniques are the most widespread and effective methods for solving problems in solid mechanics involving irregularly shaped domains. Large, complex bodies are decomposed into many small elements. A simplified law of motion is associated with each element and the behavior of the whole body may be approximated by the sum of its parts. Mathematically, these methods make use a set of lightly coupled basis functions which exhaust the domain of the body. The choice of the basis affects not only the quality of the results, but also the amount of work necessitated in solving the system. Many finite element formulations (especially thin shell elements) involve the use of high-order quantities such as slope or curvature as state variables [43, 47]. Displacement only formulations, such as those presented in this document, are preferred when coupling with other physical systems in which displacements are the only state variables used explicitly in the model (for instance rotational degrees of freedom do not appear in the Navier-Stokes equations of motion describing a Newtonian fluid).

The equations of motion for a continuum lead to a system of partial differential equations which must be satisfied over the domain of the body, and boundary conditions which are specified on the surface. Unfortunately, closed form solutions to these problems do not exist for irregularly shaped domains in general. The finite element method reduces the continuous equations of motion to a discrete set by restricting the motion of the body to those shapes which may be described by a linear combination of a chosen set of basis functions.

2.4.1 Mathematical Foundation

Equilibrium problems in solid mechanics may be approached by beginning with the statement virtual work:

$$\delta\Pi = \int_V \delta\boldsymbol{\epsilon} : \boldsymbol{\sigma} - \delta\mathbf{u}\cdot\mathbf{f}\,dV = 0 \quad \forall\delta\mathbf{u}, \delta\boldsymbol{\epsilon}. \tag{2.20}$$

This is a weak statement of equilibrium; it only guarantees a correct solution if $\delta\Pi = 0$ for all possible admissible variations in strain and displacement over the body. The Finite Element Method restricts the possible displacements and strains throughout the body to those formed by a linear combination of basis functions N:

$$\mathbf{u} = \mathbf{u}(\theta) \equiv \mathbf{v}_i N^i(\theta) \tag{2.21}$$

$$\boldsymbol{\epsilon} = \boldsymbol{\epsilon}(\mathbf{u}). \tag{2.22}$$

This step reduces the continuous statement of equilibrium to one with a finite number of degrees of freedom by projecting all possible admissible displacements onto the space of basis functions N_i. The discretized statement of virtual work, $\bar{\Pi}$ is:

$$\bar{\Pi} = \int_V \delta\epsilon : \boldsymbol{\sigma} - \delta\mathbf{u} \cdot \mathbf{f}\, dV = 0 \quad \forall \delta\mathbf{v}_i. \tag{2.23}$$

The $\delta\mathbf{v}_i$ themselves are not functions over the domain, and may be removed from the integral. The discrete statement of virtual work may be written in the form

$$\bar{\Pi} = \delta\mathbf{v}_i \cdot \mathbf{q}^i(\mathbf{v}) = 0, \tag{2.24}$$

where the $\mathbf{q}^i$ represents the unbalanced forces caused by variations in $\delta\mathbf{v}_i$. The $\mathbf{q}^i$ are computed by integrating strain and displacement over the body, which in turn involves integrating the basis functions N^i. By the statement of virtual work, the $\mathbf{v}_i$ are arbitrary, leading to the conclusion that

$$\mathbf{q}^i(\mathbf{v}) = \mathbf{0}, \quad i \in 1 \ldots n. \tag{2.25}$$

The particular definition of the $\mathbf{q}^i$ depends on the chosen definitions for strain and stress in terms of displacements.

2.4.2 *Numerical Solution Techniques*

The $\mathbf{q}^i$ form a system of nonlinear equations which may be solved numerically. To avoid the complexity of using high order quantities, the column vectors $\mathcal{V}$ and $\mathcal{Q}$ are formed by concatenating each of the $\mathbf{v}_i$ and $\mathbf{q}^i$ on top of one another respectively:

$$\mathcal{V} = \begin{bmatrix} \mathbf{v}^1 \\ \mathbf{v}^2 \\ \vdots \\ \mathbf{v}^n \end{bmatrix} \qquad \mathcal{Q}(\mathcal{V}) = \begin{bmatrix} \mathbf{q}^1(\mathcal{V}) \\ \mathbf{q}^2(\mathcal{V}) \\ \vdots \\ \mathbf{q}^n(\mathcal{V}) \end{bmatrix}. \tag{2.26}$$

Solving Equation 2.25 is now equivalent to solving

$$\mathcal{Q}(\mathcal{V}) = \mathbf{0}. \tag{2.27}$$

Many solution techniques, such as Newton's method, approximate the function by a first order Taylor series expansion:

$$\mathcal{Q}(\mathcal{V}+d\mathcal{V}) \approx \mathcal{Q}(\mathcal{V}) + \frac{d\mathcal{Q}}{d\mathcal{V}}d\mathcal{V}. \tag{2.28}$$

The derivative $\frac{d\mathcal{Q}}{d\mathcal{V}}$ is the gradient of the vector function $\mathcal{Q}(\mathcal{V})$. Using Voight's notation, this derivative will take the form of a matrix $\mathcal{K}$:

$$\mathcal{K}^{ij} = \frac{d\mathcal{Q}^i}{d\mathcal{V}_j}. \tag{2.29}$$

A newton iteration refining the approximate solution $\mathcal{V}_k$ will take the form

$$\mathcal{V}_{k+1} = \mathcal{V}_k - \mathcal{K}^{-1}(\mathcal{V}_k)\mathcal{Q}(\mathcal{V}_k), \tag{2.30}$$

where the subscript k indicates the k^{th} refinement of the solution. If the problem is linear, $\mathcal{K}$ will be constant, and the iteration will converge in one step.

2.4.3 *Stiffness Matrix*

The matrix $\mathcal{K}$ has a special meaning in solid mechanics, it is known as the stiffness matrix. For a linear problem the product $\mathcal{K}\mathcal{V}$ gives the equilibrium reaction forces, and the double product $\frac{1}{2}\mathcal{V}^T\mathcal{K}\mathcal{V}$ yields the strain energy of the system. For nonlinear problems $\mathcal{K}$ is not constant and is known as the tangent stiffness matrix. The preceding properties are now valid in a differential sense.

The tangent stiffness matrix $\mathcal{K}$ may be calculated by taking derivatives of $\mathcal{Q}$ as defined in Equation 2.29. Depending on the complexity of the basis functions N^i this approach may prove ungainly. Another approach to calculate $\mathcal{K}$ is to find the first variation $\delta^2\Pi$ of the virtual work functional $\delta\Pi$. By writing $\delta\bar{\Pi}$ in the form

$$\delta\bar{\Pi} = \mathbf{v}_i \cdot \mathbf{K}^{ij}\mathbf{v}_j, \tag{2.31}$$

the tangent stiffness matrix entries $\mathcal{K}^{ij}$ may be easily found by "unwinding" the individual entries $\mathbf{K}^{ij}$.

2.4.4 *Practical Considerations*

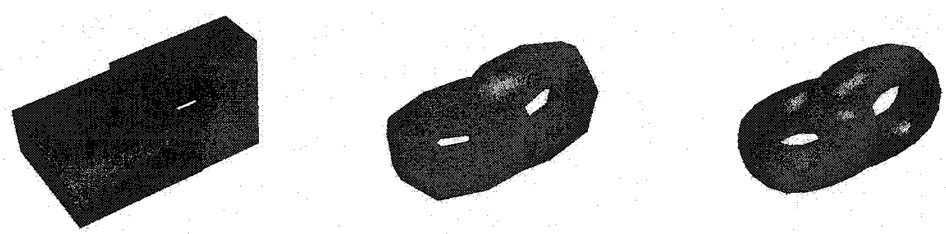
Clearly not all possible displacements can be represented by a finite number of chosen basis functions; however if the basis functions are chosen carefully to best represent the displaced configurations of the body, this discrete statement of virtual work will approximate the continuous one well. The particular choice of basis for displacement affects not only the quality of results that a model will yield, but also the amount of computational effort required to achieve them. The computation of $\mathcal{Q}$ and $\mathcal{K}$ involve integrating products of basis functions over the domain of the body. If the inner product $\int N_i N_j \, dV$ is nonzero for all i and j, then a very dense set of equations will result, which take a large amount of effort to solve. Prior to the advent of computer, orthogonal bases (those for which $\int N_i N_j \, dV$ is nonzero only for $i = j$) were preferred because they result in a very small set of equations to solve. Fourier series analyses are still preferred in the field of signal processing for this reason.

Unlike signals, continuous bodies have irregularly shaped domains and often benefit from non uniform sampling to capture local behavior. Finding an orthogonal set of basis functions to satisfy this need is difficult. An alternative to orthogonal bases are those that are "lightly coupled", those for which $\int N_i N_j \, dV$ is *often* zero. If each basis function N_i has compact support (it is zero everywhere but a small finite region of its domain) then the inner product $\int N_i N_j \, dV$ is guaranteed to be null for all pairs of basis functions with non-overlapping support. The idea of local support allows the choice of basis to be tailored to a specific need in a specific region of the domain.

2.5 Multiresolution Techniques

As elements are added to a model, it gains degrees of freedom and can better fit the actual solution. The analyst is faced with a trade off between accuracy and increased computational cost. The cost of a numerical algorithm is represented by its order. A simulation that scales in time as $O(N^2)$ for N unknowns may need only seconds to solve one thousand degrees of freedom, but could take weeks to solve for one million.

Multiresolution solvers can mitigate the cost associated with increased accuracy by considering a hierarchy of refinements of the model instead of only a single resolution. A well designed multiresolution algorithm can achieve solution times of order $O(N)$, which in general is the best one can hope to do. Multiresolution solvers can accelerate the convergence of numerical simulations by propagating solution knowledge between coarsely discretized versions of a model where solutions may be obtained quickly and finely discretized representations where solutions may be obtained more accurately. This process requires a hierarchical representation of multiple model discretizations with an ability to transfer information between neighboring levels. Figure 2.8 shows a multilevel hierarchy of models.



(a) Coarsest Model (b) Refi ned Model (c) Most Refi ned Model

Figure 2.8: Model Refinement

Chapter 3

A ROTATION-FREE SUBDIVISION-BASED BEAM ELEMENT

3.1 Introduction

This chapter describes a novel rotation-free beam finite element. The degrees of freedom for this element consist only of generalized displacements, not rotations or slopes as with standard elements. Rotation-free elements are of particular interest because of the relative ease of coupling them to other displacement based formulations which do not contain rotations or other high-order quantities as degrees of freedom. Of particular note is the use of a finite element basis which extends over a non-local, but still compact neighborhood of influence. We also develop a corresponding constraints formulation for the application of boundary conditions.

The goal of creating the rotation-free beam element is twofold. Firstly it is a useful element that can allow for analyses which require coupling between beams and other simulations which do not incorporate rotations explicitly. Secondly, as a lower-dimensional analogue of thin shell surfaces, it provides a reduced framework in which many of the issues that pertain to both analyses may be examined in a simpler setting. The ideas and derivations described in this chapter extend to the development of three-dimensional curved shell elements of Chapter 4.

Qualitatively beams are bodies that are long and slender. Many structures may be well described as an assembly of beam elements, such as tall buildings and towers. Other engineering designs include beams as major structural elements, such as vehicles and bridges. These bodies exhibit two primary load carrying mechanisms: resistance to stretching and bending. Beam theory allows engineers to predict the performance of long, slender bodies that are subjected to various forces. Often beam performance is characterized by the final deflected shape under load, and the magnitude of internal forces that develop inside the member. Internal forces which exceed material limits are an important mode of failure for engineered systems.

The engineering theory of beams is fundamental to modern mechanics and illustrates salient aspects of other structural analyses. The goal of beam theory is to use assumptions about the deformed shape of these objects to simplify the continuum equations of motion for analysis purposes. Beam theory reduces the general three-dimensional elastic continuum model down to two scalar resultant quantities, axial and bending strains, by enforcing a kinematic model of deformation. This model extends readily to the more general theory of curved shells. In this chapter we develop a rotation-free isoparametric beam finite element using the B-Spline basis functions.

(a) Initial Configuration

(b) Deformed Configuration

Figure 3.1: Model Beam Under Cantilever Load

The classic cantilever beam bending problem is illustrated in Figure 3.1. Applied load is indicated by the downward pointing arrow on the right, and a clamped boundary condition is indicated by the short diagonal lines to the left. Beam problems involve the application of various forces to the body and boundary conditions applied to the ends. In general, the undeformed configuration of the beam need not be straight. The equations of motion for beams leads to a system of continuous partial differential equations which must be satisfied throughout the volume. Further restricting deformation to a space of basis functions via the Finite Element Method discretizes the problem to a finite number of variables, which may be solved efficiently using numerical methods. The derivation in this chapter will focus on a Euler-Kirchhoff beam model capturing large strains and deformations with simple elastic material properties. The extension to nonlinear elastic materials is straightforward; a full derivation of the nonlinear equations of motion is given in Section A.1.

This chapter is organized as follows: Section 3.2 describes the parameterization of beam geometry and the kinematic assumptions that define beam motion. Computation of strain in the natural coordinate system of the beam is demonstrated in 3.3. Section 3.4 combines virtual work principles and the results of the preceding two sections to solve for equilibrium of a beam element via a stiffness matrix and generalized force vector for arbitrary displacement basis functions. Section 3.5 presents the B-Spline basis functions used to create the rotation-free element. The use of the B-Spline basis to define both beam geometry and displacement is described in Section 3.6. The numerical evaluation methods used to create the stiffness matrix and solution are described in Section 3.7. Boundary conditions and constraints are discussed in Section 3.8. Lastly the extension to nonlinear geometric properties is presented in Section 3.9, and the chapter ends with a summary in Section 3.10.

3.2 Beam Geometry and Kinematics

Any point $\mathbf{r}$ in the initial configuration of a curved beam may be parameterized by a position on the middle curve $\mathbf{x}$ and a distance along two other independent directions $\mathbf{a}_2$ and $\mathbf{a}_3$.

$$\mathbf{r}(\theta^1, \theta^2, \theta^3) = \mathbf{x}(\theta^1) + \theta^2 \mathbf{a}_2(\theta^1) + \theta^3 \mathbf{a}_3(\theta^1) \tag{3.1}$$

Let the vector $\mathbf{a}_1$ be the tangent to the curve, $\mathbf{a}_2$ be the binormal, and $\mathbf{a}_3$ be the unit normal to the curve:

$$\mathbf{a}_1 = \dot{\mathbf{x}} \qquad\qquad \mathbf{a}_3 = \frac{\mathbf{a}_1 \times \mathbf{a}_2}{|\mathbf{a}_1|}, \tag{3.2}$$

where $\dot{\mathbf{x}}$ represents differentiation with respect to θ^1. If the beam exists and deforms only in a two-dimensional plane, the vector $\mathbf{a}_2$ will be a constant unit vector pointing into the plane. The direction $\mathbf{a}_2$ serves as a mathematical convenience, it need not enter into the

(a) Initial Confi guration (b) Deformed Confi guration

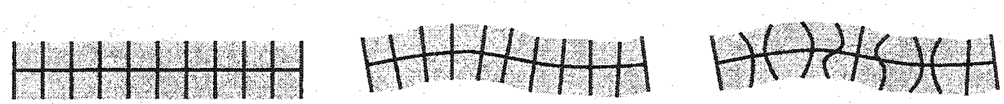
Figure 3.2: Parameterized Beam Geometry

derivation. The $\mathbf{a}_i$ now form a local coordinate system, as illustrated in Figure 3.2(a), in which direction $\mathbf{a}_2$ is assumed to point into the page.

The deformed configuration of the beam $\tilde{\mathbf{r}}$ may be similarly described by the middle curve $\tilde{\mathbf{x}}$ and the vectors $\tilde{\mathbf{a}}_1$, $\tilde{\mathbf{a}}_2$ and $\tilde{\mathbf{a}}_3$, as shown in Figure 3.2(b). The kinematic assumptions that define an Euler (Kirchhoff) beam are that lines initially normal to the middle curve:

- remain straight after deformation
- do not change length
- remain normal to the middle curve of the deformed geometry.

The kinematically allowable deformations of a beam are illustrated in Figure 3.3 The third



(a) Initial Confi guration (b) Allowable Deformation (c) Not Considered

Figure 3.3: Kinematic Beam Deformations

and last assumption eliminates shearing deformation, this is the difference between the Euler-Kirchhoff and Timeshenko beam theories, which allow lines that were initially perpendicular to the middle curve to change orientation. Building the Kirchhoff restrictions into our parameterization of the deformed configuration we find

$$\tilde{\mathbf{r}}(\theta^1, \theta^2, \theta^3) = \tilde{\mathbf{x}}(\theta^1) + \theta^2 \tilde{\mathbf{a}}_2(\theta^1) + \theta^3 \tilde{\mathbf{a}}_3(\theta^1), \tag{3.3}$$

$$\tilde{\mathbf{a}}_1 = \dot{\tilde{\mathbf{x}}} \qquad \tilde{\mathbf{a}}_2 = \mathbf{a}_2 \qquad \tilde{\mathbf{a}}_3 = \frac{\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2}{|\tilde{\mathbf{a}}_1|}. \tag{3.4}$$

This completes the parametric description of the geometry of the beam in the initial and deformed configurations.

3.3 Strain

Computing strain for an initially curved member is most easily accomplished in curvilinear coordinates. The covariant components of the strain tensor ϵ are defined as

$$\epsilon_{ij} = \frac{1}{2}\left(\tilde{g}_{ij} - g_{ij}\right), \tag{3.5}$$

where $\tilde{g}$ and g represent the metric of the deformed and undeformed configurations respectively. By our choice of parameterization of the initial and deformed configurations (Equations 3.1, 3.2, 3.3, 3.4) the kinematic assumptions described in the preceding section will automatically be satisfied. To first order in θ^i (and noting that $\mathbf{a}_1 \cdot \dot{\mathbf{a}}_3 = -\dot{\mathbf{a}}_1 \cdot \mathbf{a}_3$)

$$\tilde{g}_{11} = \tilde{\mathbf{a}}_1 \cdot \tilde{\mathbf{a}}_1 - 2\theta^3 \dot{\tilde{\mathbf{a}}}_1 \cdot \tilde{\mathbf{a}}_3 \qquad g_{11} = \mathbf{a}_1 \cdot \mathbf{a}_1 - 2\theta^3 \dot{\mathbf{a}}_1 \cdot \mathbf{a}_3. \tag{3.6}$$

As a result of the kinematic assumptions, all other strain components may be ignored. The strain in the member contains two principle components, and may be written as:

$$\epsilon_{11} = \alpha + \theta^3 \beta, \tag{3.7}$$

$$\alpha = \frac{1}{2}(\tilde{\mathbf{a}}_1 \cdot \tilde{\mathbf{a}}_1 - \mathbf{a}_1 \cdot \mathbf{a}_1) \qquad \beta = -\dot{\tilde{\mathbf{a}}}_1 \cdot \tilde{\mathbf{a}}_3 + \dot{\mathbf{a}}_1 \cdot \mathbf{a}_3. \tag{3.8}$$

The α term represents strain in the axial direction, while β measures deviation of the normal, which indicates a change in curvature. These are the two primary modes of deformation of beams. The ability of the body to resist changes in length and curvature governs its physical performance.

It is customary in solid mechanics to represent the deformed configuration as a displacement from the initial shape:

$$\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{u}. \tag{3.9}$$

3.4 Equilibrium

The statement of virtual work (Equation 3.10) may always be used to find the equilibrium configuration of a body under load. In the case of an elastic body, an equivalent statement of equilibrium may be derived through minimization of potential energy.

$$\delta\Pi = \int_V \delta\epsilon : \boldsymbol{\sigma} - \delta\mathbf{u} \cdot \mathbf{f}\, dV = 0 \quad \forall \delta\epsilon, \delta\mathbf{u}, \tag{3.10}$$

where $\delta\epsilon$ and $\delta\mathbf{u}$ are compatible strains and displacements which satisfy the essential boundary conditions and $\boldsymbol{\sigma}$ are the internal stresses in equilibrium with the applied external load $\mathbf{f}$. Because $\mathbf{x}$ and $\tilde{\mathbf{x}}$ are chosen to be the middle line, membrane (stretching) and bending components of strain are fully decoupled. For a beam, both membrane and bending strains are scalar quantities. In it's own context, the membrane strains α and the bending strains β will each be designated by the symbol ϵ for simplicity in substituting into the variational equations of motion 3.10, and later 3.12. The total strain is the sum of the membrane and bending parts.

Solving for equilibrium is equivalent to finding the zero of the elastic virtual work functional $\delta\Pi(\mathbf{u})$ where strain is given as a function of displacements. Equation 3.10 is a nonlinear equation with displacements $\mathbf{u}$ (which appear in $\epsilon(\mathbf{u})$, $\boldsymbol{\sigma}(\mathbf{u})$ and/or $\mathbf{f}(\mathbf{u})$). Conceptually the zero of the elastic virtual work functional may be solved for using iterative technique, such as Newton's method. Iterative numerical methods for solving nonlinear problems in solid mechanics will be discussed in depth in the proposal. Many numerical methods, including Newton's approximate the virtual work functional by a Taylor series. The first order Taylor series expansion of the virtual work functional is

$$\delta\Pi(\mathbf{u} + \delta\mathbf{u}) = \delta\Pi(\mathbf{u}) + \left.\frac{\delta\delta\Pi}{\delta\mathbf{u}}\right|_{\mathbf{u}} \delta\mathbf{u} + O(\delta\mathbf{u}^2). \tag{3.11}$$

The variational derivative of the elastic virtual work functional is

$$\frac{\delta\delta\Pi}{\delta\mathbf{u}}\delta\mathbf{u} = \delta^2\Pi = \int_V \delta\epsilon : \mathbf{C}\delta\epsilon + \delta^2\epsilon : \sigma\, dV, \tag{3.12}$$

where $\mathbf{C}$ is the tangent constitutive modulus, defined as

$$\mathbf{C} = \frac{\delta\sigma}{\delta\epsilon}. \tag{3.13}$$

As displacement $\mathbf{u}$ is a continuous field value over the domain of the body, there is little hope in general of solving for $\delta\Pi(\mathbf{u}) = 0$ with the conceptual framework presented here. Discretizing the body into finite elements reduces displacement from a continuous field to a discrete one, allowing the application of numerical techniques.

3.4.1 *Membrane Strains*

Here we present a derivation for the virtual work equations of equilibrium for a subdivision beam. The steps in this process involve defining strain, stress, energy and their variations in displacement. The membrane strains and variations are as follows:

$$\epsilon = \dot{\mathbf{u}} \cdot \mathbf{a}_1 + \frac{1}{2}\dot{\mathbf{u}} \cdot \dot{\mathbf{u}} \tag{3.14}$$

$$\delta\epsilon = \delta\dot{\mathbf{u}} \cdot (\mathbf{a}_1 + \dot{\mathbf{u}}) = \delta\dot{\mathbf{u}} \cdot \tilde{\mathbf{a}}_1 \tag{3.15}$$

$$\delta^2\epsilon = \delta\dot{\mathbf{u}} \cdot \delta\dot{\mathbf{u}}. \tag{3.16}$$

Compute energy as per the previous section:

$$\delta\Pi = \int_V \delta\dot{\mathbf{u}} \cdot \tilde{\mathbf{a}}_1\mathbf{C}\epsilon - \delta\mathbf{u} \cdot \mathbf{f}\, dV \tag{3.17}$$

$$\delta^2\Pi = \int_V \delta\dot{\mathbf{u}} \cdot \tilde{\mathbf{a}}_1\mathbf{C}\tilde{\mathbf{a}}_1 \cdot \delta\dot{\mathbf{u}} + \delta\dot{\mathbf{u}} \cdot \delta\dot{\mathbf{u}}\mathbf{C}\epsilon\, dV. \tag{3.18}$$

Restrict the displacements to the subdivision basis

$$\mathbf{u}(\theta) = N^i(\theta)\mathbf{v}_i \qquad\qquad \delta\mathbf{u} = N^i(\theta)\delta\mathbf{v}_i. \tag{3.19}$$

Solving for equilibrium $\delta\Pi = 0 \;\forall\delta\mathbf{u}$ yields a system of nonlinear equations:

$$\delta\mathbf{v}_i \cdot \int_V \dot{N}^i \tilde{\mathbf{a}}_1 \mathbf{C}\epsilon - N^i \mathbf{f}\, dV = 0 \quad \forall\delta\mathbf{v}_i. \tag{3.20}$$

A Newton-like solver will use the second variation $\delta^2\Pi$ to solve the system

$$\delta^2\Pi = \delta\mathbf{v}_i \mathbf{K}^{ij} \delta\mathbf{v}_j \tag{3.21}$$

$$\mathbf{K}^{ij} = \int_V \dot{N}^i \dot{N}^j \left[\tilde{\mathbf{a}}_1 \otimes \mathbf{C}\tilde{\mathbf{a}}_1 + \mathbf{C}\epsilon\mathbf{1}\right] dV. \tag{3.22}$$

Assuming that the area A of the cross section remains constant along the length of the element,

$$\mathbf{K}^{ij} = A \int_{\theta=0}^{\theta=1} \dot{N}^i \dot{N}^j \left[\tilde{\mathbf{a}}_1 \otimes \mathbf{C}\tilde{\mathbf{a}}_1 + \mathbf{C}\epsilon\mathbf{1}\right] |\mathbf{a}_1| d\theta. \tag{3.23}$$

3.4.2 *Bending*

The bending strains are defined as:

$$\begin{aligned} \epsilon &= -\dot{\tilde{\mathbf{a}}}_1 \cdot \tilde{\mathbf{a}}_3 + \dot{\mathbf{a}}_1 \cdot \mathbf{a}_3 \\ &= -(\dot{\mathbf{a}}_1 + \ddot{\mathbf{u}}) \cdot \frac{(\mathbf{a}_1 + \dot{\mathbf{u}}) \times \tilde{\mathbf{a}}_2}{|\mathbf{a}_1 + \dot{\mathbf{u}}|} + \dot{\mathbf{a}}_1 \cdot \mathbf{a}_3. \end{aligned} \tag{3.24}$$

The same process may be applied to define the bending stiffness matrix as was used to define the membrane stiffness. The derivation is relegated to appendix Section A.1. The result has the form

$$\mathbf{K}^{ij} = I \int_{\theta=0}^{\theta=1} \ldots |\mathbf{a}_1| d\theta, \tag{3.25}$$

where I represents the moment of inertia of the cross-sectional area of the beam. If the cross section is rectangular and solid,

$$I = \frac{width * height^3}{12}. \tag{3.26}$$

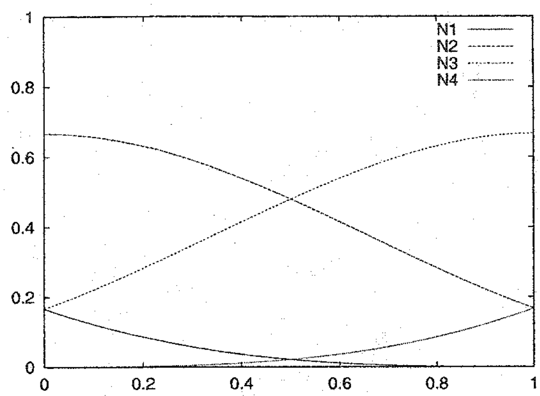
3.5 Spline Basis

The spline basis provides an excellent choice for describing both the geometry and deformation (Equation 3.19) of a beam finite element The four cubic B-Spline (basic spline) basis polynomials [29, 20] are defined in Equation 3.27 and plotted in Figure 3.4(a). Each basis function has compact support over the range $0 \leq \theta \leq 1$ and is defined to be 0 outside of this domain. Concatenating each of the individual basis functions together creates a full spline basis function with compact support over the range $-2 \leq \theta \leq 2$. A full B-Spline basis curve, formed piecewise from each of the four B-Spline polynomials is shown in Figure 3.4(b). The full cubic B-Spline basis functions have a number of noteworthy properties: they form a partition of unity, are C^2, and are refinable (the B-Spline basis may be written as the sum of translates and dilates of itself). For beam analysis we will only be concerned with the first two properties, though the third property is the source for much of the interest taken in the B-Spline basis [32, 29], and it is very important to the study of shells using subdivision surfaces. The reader should take note that these functions are *not* interpolating, unlike most standard finite element shape functions. They also extend over a 2-neighborhood of influence, that is each control point affects not only the spline segment immediately to the left and right, but also the next spline segment in each direction. The simple polynomial form of the basis functions allows them to be evaluated directly.

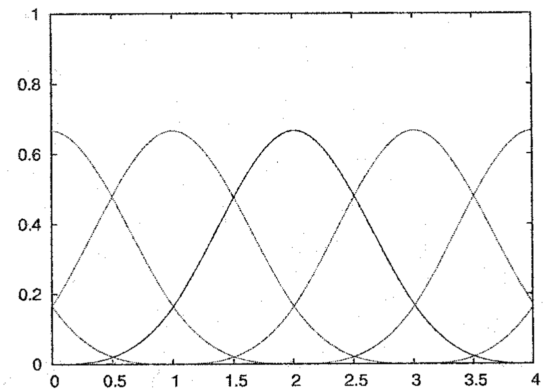
$$\begin{aligned} N_1 &= \tfrac{1}{6}(1-\theta)^3 & N_2 &= \tfrac{1}{6}(3\theta^3 - 6\theta^2 + 4) \\ N_3 &= \tfrac{1}{6}(-3\theta^3 + 3\theta^2 + 3\theta + 1) & N_4 &= \tfrac{1}{6}\theta^3 \end{aligned} \tag{3.27}$$

3.6 Discretization

As shown in Figure 3.5(a) the geometry of a general curved beam may be described piecewise by weighting the geometric points of a control polygon (illustrated as the dotted lines) by the B-Spline basis functions. Several spline segments and a larger control polygon are shown in Figure 3.5(b); the resulting curve is C^2.

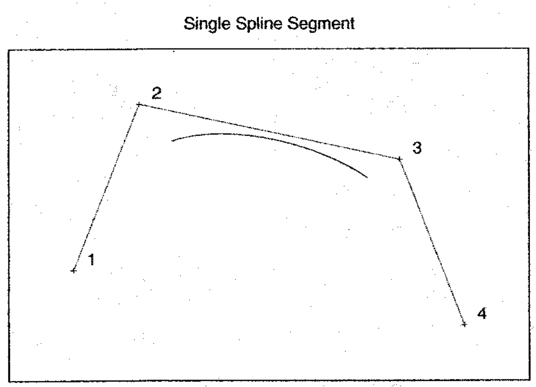


(a) Spline Basis Functions

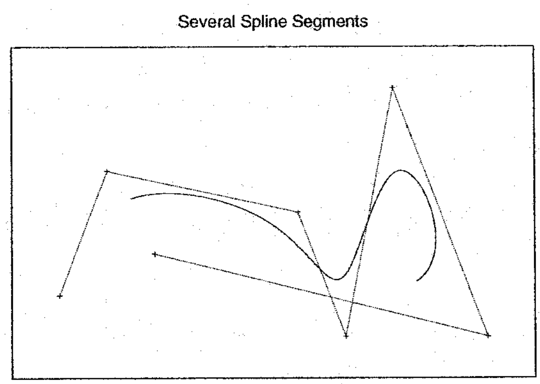


(b) Spline Basis Over Several Intervals

Figure 3.4: B-Spline Basis



(a) Spline with Local Numbering



(b) Several Spline Segments

Figure 3.5: Beam Geometry

The control polygon is an ordered set of geometric control points. The equation of each spline segment may be written as

$$\mathbf{x}(\theta) = \sum_{1}^{4} N^I(\theta)\mathbf{x}_I, \tag{3.28}$$

where the $\mathbf{x}_I$ represent the vertices of the control polygon ordered as shown in Figure 3.5(a). Similarly the deformed configuration of the beam may be written as

$$\tilde{\mathbf{x}}(\theta) = \sum_{1}^{4} N^I(\theta)\tilde{\mathbf{x}}_I = \sum_{1}^{4} N^I(\theta)(\mathbf{x}_I + \mathbf{u}_I) = \mathbf{x}(\theta) + \sum_{1}^{4} N^I(\theta)\mathbf{u}_I. \tag{3.29}$$

Each spline segment requires four geometric control points which are shared with neighboring splines and may be used to define a single beam finite element. Standard finite elements are described by both boundary and interior nodes, but only boundary nodes are traditionally shared. In the spline approach local control point numbers 1,2,3 are shared with the left neighboring element, and control points 2,3,4 are shared with the right neighboring element.

Displacements along the beam may be discretized using the very same basis functions that represent geometry. Finite elements of this type are known as isoparametric elements. Representing both geometry and displacement by the same basis creates a unified approach to representation of form and function. Most finite elements have limited geometric representation capabilities; models are generally designed using one representation scheme and them "meshed" into elements. Splines are a widely accepted modeling paradigm for curves.

If a total of m segments are arranged in a loop, exactly m control points corresponding to m spline basis functions will be needed, for a total of $n = 2m$ degrees of freedom. Thus, ignoring boundaries for non-closed geometry, each beam segment requires only 2 *amortized* degrees of freedom. This compares favorably to the standard cubic Euler beam element (with axial strain) which uses 3 amortized DOF per element (6 DOF total, four for displacement and two for rotation, each shared half with the neighboring elements).

3.7 Evaluation

Equations 3.23 and 3.20 and the corresponding bending equations must be evaluated to create the global tangent stiffness matrix and vector of unbalanced forces. All of these quantities involve products of basis functions corresponding to degrees of freedom of the model. Because the B-Spline basis functions have non-zero support only over a neighborhood of elements the integration may efficiently performed one element at a time, each time using a stencil which includes only those basis functions attached to nodes in the neighborhood

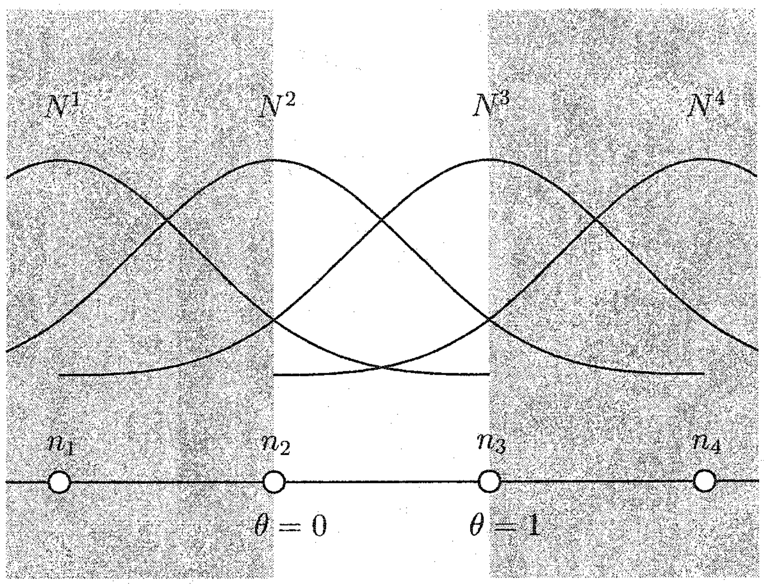


Figure 3.6: Beam Element Stencil

of the element.

The element-wise integration stencil is illustrated in Figure 3.6. The white area between nodes 2 and 3 indicates the area in which integration takes place, and spans the domain $0 \leq \theta \leq 1$. The basis functions corresponding to nodes 1,2,3 and 4 must are included when evaluating the local stiffness matrix $\mathbf{K}$ and vector of unbalanced forces $\mathbf{q}$ for the element. This process is repeated for all elements in the model. Each of the local $\mathbf{q}$ and $\mathbf{K}$ are assembled to form the global $\mathcal{Q}$ and $\mathcal{K}$ matrices of Equations 2.24 and 2.31.

Even though the basis functions are simple polynomials, the integral is itself rational due to the presence of the Jacobian term. This term arises due to the transformation of variables inherent in a parametric description; unlike standard beam formulations, the subdivision element is not arc-length parameterized. Consequently numerical methods must be used to approximate the results. A three point Gauss quadrature scheme is sufficient in practice to evaluate all of the integrated values.

3.8 Boundary Conditions

General problems in solid mechanics, and beam analysis in particular, are formulated with certain constraints posed at the boundaries of the model. These constraints enforce connections between the model and other members of a mechanical system, such as a rigid support or another structural element. Because the B-Spline basis functions are not interpolating, imposing boundary conditions can require the use of Lagrangian constraints. Two families of boundary conditions have been implemented and explored. Compatible constraints are compatible in spirit with prior literature and are discussed in Section 3.8.1. A new improved family of boundary conditions known as Point-wise Constraints are described in Section 3.8.2.

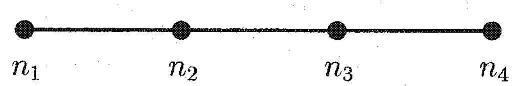


Figure 3.7: Beam Boundary Nodes

A canonical B-Spline beam element is illustrated in Figure 3.7. Nodes $n_1 \dots n_4$ are shown; to each node n_i there corresponds a generalized displacement degree of freedom $\mathbf{u}_i$. The most common types of constraints applied to beams in practice are specifying that one end cannot translate, cannot rotate (does not change slope), or both. The scope of the following discussion will be limited to constraining the position and rotation of node n_2.

3.8.1 Compatible Constraints

The boundary conditions described in this section are designed in the same spirit of those inspired by [53] and described in [17]. Because they were created to be compatible with prior approaches in the literature, they are designated as compatible constraints. Table 3.1 gives relations for applying compatible boundary conditions to node n_2. The constraint is written in terms of displacements $\mathbf{u}_i$ corresponding to node n_i.

Table 3.1: Compatible Constraints

Displacement	Rotation	Boundary Condition
Free	Free	$\mathbf{u}_1 = 2\mathbf{u}_2 - \mathbf{u}_3$
Fixed	Free	$\mathbf{u}_2 = 0,\ \mathbf{u}_1 = -\mathbf{u}_3$
Free	Fixed	$\mathbf{u}_1 = \mathbf{u}_2 = \mathbf{u}_3$
Fixed	Fixed	$\mathbf{u}_1 = \mathbf{u}_2 = \mathbf{u}_3 = 0$

These constraints are generally simple to implement. The case of fixing both the displacement and rotation of an endpoint is especially simple; because a set of degrees of freedom are identically set to zero, the corresponding rows and columns of the stiffness matrix may simply be eliminated, resulting in a smaller problem size. As in the case of shells (discussed in Chapter 4) an endpoint with free displacement and rotation is still constrained. In all cases shown in Table 3.1 the vector $\mathbf{u}_1 - \mathbf{u}_2$ is defined to be opposite of $\mathbf{u}_3 - \mathbf{u}_2$, enforcing reflective symmetry about node n_2. This has the unfortunate side effect of constraining both bending strains and axial strains simultaneously, and they cannot be decoupled.

3.8.2 Point-wise Constraints

An alternative set of constraints may be formulated by constraining the actual displacement and slope of an endpoint, without requiring any symmetry of the neighborhood. These constraints apply to the displacement formulation of the geometry of the deformed configuration of the beam defined in Equation 3.9. The constraint is applied to the discretized geometry using the basis functions of Equation 3.27. The point-wise displacement and slope constraints are derived in sections 3.8.2 and 3.8.2. No constraints are necessary at all if neither displacement nor rotation are fixed. Equation 3.30 describes a constraint for preventing displacement. Equation 3.33 describes a constraint for preventing changes in slope. Both of these constraints may be combined to form a fixed boundary condition which allows neither changes in displacement, nor slope.

Point-wise Displacement Constraint

The displacement $\mathbf{u}_2$ of node n_2 may be found by evaluating the weighted sum $N^i\mathbf{v}_i$ evaluated at $\theta = 0$. This relation is shown more generally for any parametric location in matrix form in Equation 3.30 which forms the constraint. At $\theta = 0$ the vector of basis functions $[N_1, N_2, N_3, N_4]$ takes on the values $[\frac{1}{6}, \frac{2}{3}, \frac{1}{6}, 0]$.

$$\begin{bmatrix} N_1(\theta) & N_2(\theta) & N_3(\theta) & N_4(\theta) \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \end{bmatrix} = \mathbf{0} \tag{3.30}$$

Point-wise Rotation Constraint

The rotation of any point of the beam may be specified by fixing the slope. The slope may be derived from the parametric tangent vectors $\mathbf{a}_1$ and $\tilde{\mathbf{a}}_1$ of the initial and deformed configurations respectively. It is important to make a distinction between the slope and tangent vectors. The slope is geometric property of the middle line of the beam, whereas the tangent vector is a parametric one. It would be a mistake to fix the slope of the beam by requiring the tangent vector of the deformed and undeformed configurations of the beam to be the same vector value. Although this would effectively constrain the slope, it would also introduce the unwanted artifact of constraining the axial strain of the body as well. The compatible constraints described in Section 3.8.1 suffer from this inadequacy.

A constraint on the slope of the beam may be expressed in terms of the parametric tangent of the deformed configuration $\tilde{\mathbf{a}}_1$ and the normal of the undeformed configuration $\mathbf{a}_3$ vector as:

$$\mathbf{a}_3 \cdot \tilde{\mathbf{a}}_1 = 0. \tag{3.31}$$

This constraint only affects the direction of the tangent but not its magnitude. This formulation allows the slope to be fixed while not imposing unwanted constraints on the axial strains. Discretizing Equation 3.31 leads to:

$$\mathbf{a}_3 \cdot \left(\mathbf{a}_1 + \frac{\partial N_i}{\partial \theta} \mathbf{v}^i \right) = 0. \tag{3.32}$$

As $\mathbf{a}_3 \cdot \mathbf{a}_1 = 0$ by definition, the final constraint may be formulated as

$$\sum_i \frac{\partial N_i}{\partial \theta} (\mathbf{a}_3 \cdot \mathbf{v}^i) = 0. \tag{3.33}$$

As is the case with the point-wise displacement constraint, this formulation may be applied anywhere along the length of the beam, not only at nodal locations.

The point-wise constraints give superior convergence compared to compatible constraint implementations. Figure 3.8 shows a convergence plot for both types of constraints applied to the simple linear cantilever beam problem illustrated in Figure 3.1. The problem was modeled with equally spaced nodes across the length of the beam. In both cases a fixed constraint was applied to the node corresponding to the leftmost end of the beam; due to the use of local neighborhoods in the geometric description, this is always the second node from the left.

Energy was computed as $\mathbf{v}^T \mathbf{K} \mathbf{v}$ for the finite element simulation, and as $\int_L \mathbf{f} \cdot \mathbf{u}\, dx$ for the known solution to the linear cantilever problem. The compatible constraint solution converges linearly to a known answer. The point-wise implementation exhibits fourth order convergence in the energy norm for this problem, a significant advantage.

3.9 Nonlinear Analysis

The subdivision beam element extends easily to the finite deformation range by starting with a statement of virtual work based on the Green strain tensor. In this case the statement of virtual work is not a linear function of displacement, and an iterative solution approach is required to solve for equilibrium.

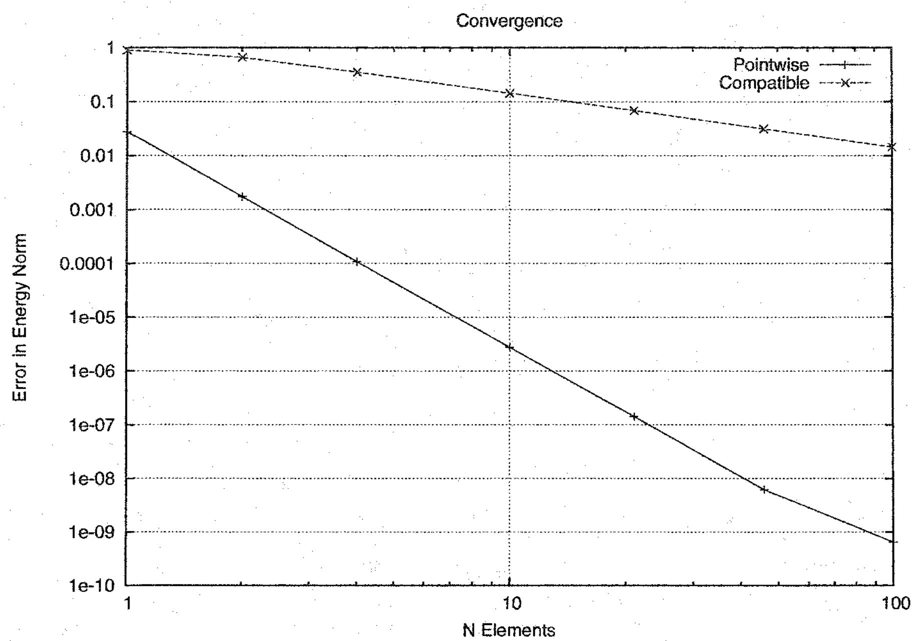


Figure 3.8: Energy Norm Convergence For Cantilever Beam

The derivation of the equations virtual work for the subdivision beam finite element are long and are presented in Section A.1 of the appendix. We employ a straightforward Newton iteration approach to solving these equations. At each step a tangent stiffness matrix is generated corresponding to the gradient of the virtual work functional at the current position. At each step a new equilibrium position is calculated using the tangent stiffness matrix. A new tangent stiffness matrix is computed and the solution is refined until the sum of unbalanced forces falls below an accuracy threshold.

The first few deformed shapes found by an iterative approach to an example problem are illustrated below in Figure 3.9. The problem shown is that of a simple cantilever beam with fixed displacement and rotation at the left end, and a downward vertical load at the right. The problem is modeled with a single subdivision finite element with appropriate boundary conditions. The specific beam chosen is an $S6x17.25$ steel beam [31] with a cross-sectional area of $32.7cm^2$, a moment of inertia of $1095cm^4$, a length $500cm$, an elastic modulus of

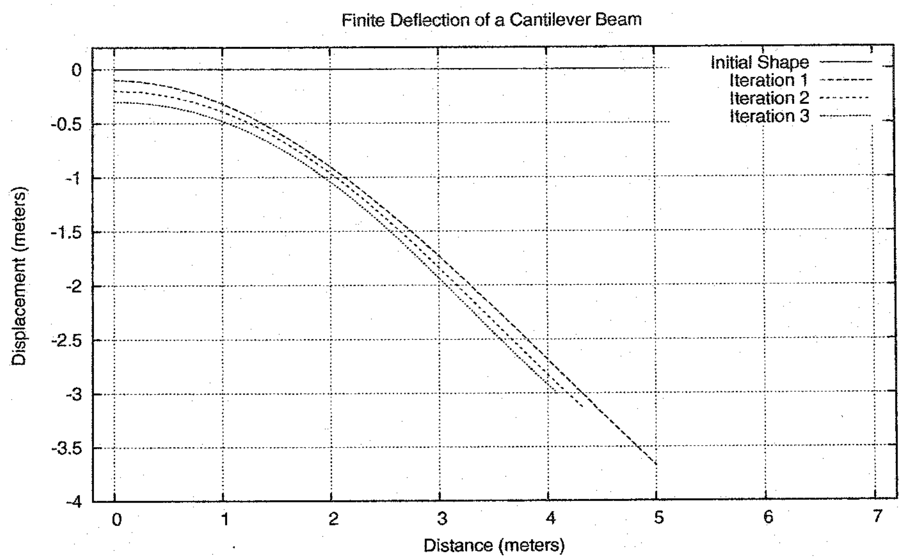


Figure 3.9: Nonlinear Solution of a Cantilever Beam

$200GPa$ and an end load of $100kN$.

The initial configuration is a straight line shown at the top. The first iterative solution is the same as the linear solution, as initially all displacements are zero. While this linear solution estimates the overall shape of the solution quite well, it neglects to account for the stretch of the beam due to vertical displacement. Further iterations return the beam to near its original length, including some stretching caused by the load. Further iterations yield little visual improvement after the 3rd iteration. For clarity each iteration is displaced vertically by a small amount to avoid overlapping plots.

3.10 Summary

This chapter has described a novel rotation-free beam finite element which uses only generalized displacements are used as nodal variables. The derivation of the subdivision beam finite element has been demonstrated for the non-local subdivision basis functions. Relevant boundary conditions have been developed with demonstrated high-order convergence. This element shows excellent convergence properties in the linear case, and extends easily to the finite deformation range. The subdivision beam element is not only useful in its own right, but serves as a two dimensional analogue of curved, three dimensional rotation-free shell element described in Chapter 4.

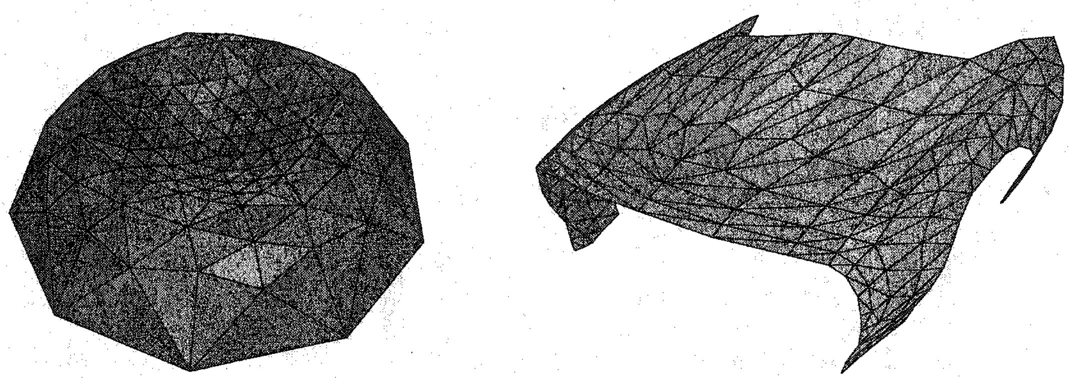
Chapter 4

A MULTI-LEVEL,SUBDIVISION-BASED, ROTATION-FREE THIN SHELL FINITE ELEMENT: THEORY AND IMPLEMENTATION

4.1 Introduction

This section describes the derivation and implementation of a hierarchical, rotation-free thin shell finite element for meshes with both quadrilateral and triangular connectivity. This formulation makes use of only generalized displacements as nodal variables, not rotations or slopes. Like the beam element of Chapter 3, the shell element uses a basis which extends over a non-local, but still compact neighborhood of influence. Rotation-free elements are of particular interest because of the relative ease of coupling them to other displacement based formulations which may not admit rotations or other high-order quantities as degrees of freedom. Subdivision approaches provide a unified framework for describing the geometry and deformation of thin bodies. By construction, the subdivision shape functions provide the necessary continuity requirements for representing the solution of the fourth-order equilibrium equations governing the behavior of Kirchhoff-Love thin shells. These elements also possesses a natural hierarchy which allows for efficient numerical preconditioning of the discretized equations of equilibrium.

Shell theory is applicable to model thin bodies which are well described by a thickened surface. They may be initially flat like floors and pieces of paper, or curved like a soup bowls and automotive body panels. Although these bodies may be analyzed as general continua, the few simple assumptions of then shell theory simplify the equations of motion considerably and elucidate the behavior of the body. The basic postulate of thin shell theory is the assumption that behavior throughout the thickness of the body is be well modeled as a function of the deformation of the "middle surface," a thin imaginary sheet that is equidistant from the top and bottom surfaces of the physical body. Shell theory relates the motion of thin bodies to loads applied to the middle surface, with specific attention to the deformation of the body and the internal forces that result. An example of a blood cell membrane and a car hood modeled as discretized shells are shown in Figure 4.1(b) (see also Figure 1.1).



(a) Red Blood Cell (b) Car Hood

Figure 4.1: Discretized Thin Shell Models

The study of engineering plates and shells has been an active topic of research for over a century. Only the simplest geometries and loading conditions admit closed form solutions; numerical methods are required to solve problems involving general shapes. Finite element methods are now used to solve nearly all general engineering shell problems. Finite element methods proceed by approximating large, complex bodies as the sum of many small pieces called elements. The geometry of each element is approximated using a few discrete control values, effectively reducing the continuous law of motion or equilibrium to a discrete one. The discretization of the body and the approximation used for each element cannot be chosen at random; care must be taken to ensure that the discrete motion of each element well approximates the continuous motion of the body.

Carefully designed approaches are necessary to avoid the physical and numerical difficulties inherent to simulating shell motion. A complete and well written review of thin shell solution techniques is presented in [60] and a history of the development of shell finite elements is given in [43]. Solution techniques must delicately balance distinct modes of deformation, which may affect the solution by different orders of magnitude. Formulating engineering models which are both physically correct and numerically robust for arbitrary situations remains an intense area of study.

Plate and shell theories arise from the deceptively simple hypothesis that lines normal to the hypothetical middle surface of a thin body remain straight after deformation. This kinematic restriction leads to the three principal modes of shell deformation: membrane, bending and shear straining. The interaction of these three modes is the cause for much of the current study in shell analysis.

The rest of this chapter is devoted to developing hierarchical rotation free triangular and quadrilateral thin shell elements using subdivision surface basis functions. An overview of the machinery of classical elasticity and the finite element method are discussed in sections 2.3 and 2.4. The theory of thin shells and existing approaches to thin shell finite elements are presented in Section 4.2. Subdivision-based approaches are discussed in 4.3. The geometry of thin shells is presented in Section 4.4 which involves parameterizing the middle surface and its normal. Section 4.5 describes the computation of strain between initial reference geometry and a deformed configuration of the body as a function of displacement in a natural coordinate system. The mathematical foundations of the thin shell element are presented in Section 4.6; variational principles are used to discretize the deformation of the body. Section 4.7 describes both the triangular and quadrilateral elements implemented for this work and the novel evaluation techniques used. Section 4.8 shows the triangular and quadrilateral subdivision basis functions for a variety of local topologies. Relationships between the numbers and types of elements are described in Chapter 4.9. The multilevel nature of these elements and their implementation are discussed in sections 4.10 and 4.11. The chapter finishes with a summary in Section 4.12.

The shell element itself is only part of the subdivision thin shell finite element framework. The approach to representing model boundaries, and a novel application of boundary conditions are described in Chapter 5. An efficient solution strategy to the discretized equations of motion is presented in Chapter 6.

4.2 Background

4.2.1 Reissner-Mindlin Thick Shell Model

Reissner-Mindlin shell theories allow for shearing throughout the thickness of the shell as well as out of plane and membrane stretching in-plane effects, and best model modestly thick shells. These formulations are attractive for finite element implementations because they require only simple C^0 compatibility between elements which simplifies the underlying basis functions. However they often do not perform well in the thin shell limit. For thin shells, membrane and shearing energies dominate bending when the strains are of comparable magnitudes. Spurious shears arise when an inability to represent a geometric state of pure bending induces artificial shearing energy in a discretized formulation. Due to the dominance of the shearing energies, these spurious shears can pollute bending modes and greatly over-stiffen the analysis. It is difficult to create Reissner-Mindlin elements in which the shearing strains are completely decoupled from bending modes of deformation.

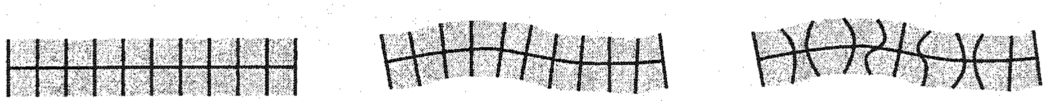
4.2.2 Curved Elements

Curved geometries may be approximated as a collection of several flat or nearly flat elements. However true curved elements are preferable. Bending of flat elements is purely inextensional, that is changes in curvature do not induce any membrane strain. Closed convex surfaces are rigid to inextensional bending and thus cannot be accurately modeled by flat elements [60]. Doubly-curved shells with positive Gaussian curvature are much less likely to bend inextensionally than those with zero or negative Gaussian curvature. Membrane forces are a significant load carrying mechanism and contribute greatly to the stiffness of curved bodies; ignoring the coupling between bending and membrane strains is detrimental to solution accuracy. Flat elements may also exhibit spurious moments along element boundaries. Despite their disadvantages in accuracy, flat and nearly flat shells (shallow shells) are simpler to formulate mathematically as they may have their strains approximated in Cartesian components. General curved shell analysis requires the use of curvilinear coordinates, which increases the complexity of derivations, but provides increased accuracy.

4.2.3 Kirchhoff-Love Thin Model

The Kirchhoff-Love model of thin shells places three kinematic restrictions on the allowable modes of deformation. Lines initially normal to the middle surface

- remain straight after deformation,
- do not change length,
- remain normal to the middle curve of the deformed geometry.



(a) Initial Confi guration (b) Allowable Deformation (c) Not Considered

Figure 4.2: Kinematically Allowable Deformations

Figure 4.2, borrowed from the previous chapter, illustrates the allowable modes of deformation. Instead of only applying to the middle line as in the case of beams, Figure 4.2 now applies to any possible cut made normal to the middle surface of the shell. The last restriction is the one which differentiates Kirchhoff-Love analysis from Reissner-Mindlin theory. This restriction eliminates shear strains throughout the cross section and therefore eliminates spurious shears as an issue. The Kirchhoff-Love model has been shown to be very effective at modeling thin shells; those for which the ratio of shell thickness to change in radius of curvature (bending) is very small; however, increased accuracy in the thin shell limit comes with a price: the basis functions used in a finite element model must be C^1 compatible (formally H^2 square integrable) across element boundaries. Until recently basis functions which posses these properties have been difficult to formulate, which has inhibited their acceptance in commercial applications. So called "Discrete Kirchhoff" elements which enforce the Kirchhoff normality constraint only along isolated lines or points have proven quite popular [60].

Shell finite element formulations (both thick and thin) make use of several non displacement degrees of freedom. A common plate element, the DKT-CST (Discrete Kirchhoff Triangle [bending] Constant Strain Triangle [membrane]) element has three nodes with three displacement degrees of freedom and two rotational degrees of freedom each. Non displacement degrees of freedom are difficult to couple to other types of analyses such as fluid flows and there has been a great degree of interest in creating rotation-free shell elements for this reason.

4.2.4 Existing Rotation-Free Approaches: BST and BSN elements

Oñate and Zárate [47] have successfully created a family of rotation free plate and shell elements using only displacement degrees of freedom. They describe two distinct approaches, the element-centric BPT (Basic Plate Triangle) and the vertex centric BPN (Basic Plate Nodal patch) as shown in Figure 4.3. In each formulation the idea of considering the displacements of a neighborhood of elements is essential. The BPT element considers the displacement of the current triangle as well as the neighbors which it shares edges with. As each triangle possesses exactly three edge-sharing neighbors, (excluding elements on the boundary) the stiffness matrix remains of constant size. The BPN approach considers the instead the union of one third of each triangle valent to a central vertex (forming a ring around it) to be an element for integration purposes. The size of the stiffness matrix of each BPN varies with the valence of the central vertex.

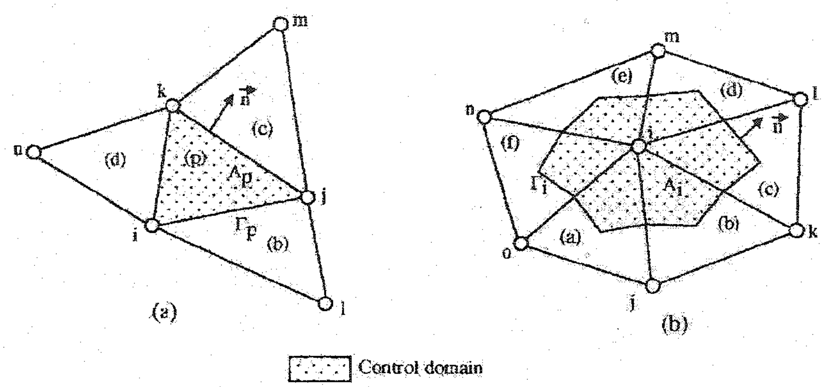


Figure 4.3: BTP and BPN Elements (reprinted from [47])

Integration of the curvature tensor (second derivatives of the displacement basis functions) over the area of an element is transformed by Green's identities to a line integral around the periphery of the triangle of the normal component of the gradients of the displacement field. In order to ensure continuity in the BPT approach, the displacement gradient is calculated separately for each element adjacent to an edge, and then averaged. In the refinement limit this difference is assumed to be negligible. The BPN element has the advantage of not requiring continuity across shared edges, but can be difficult to evaluate.

Each of the previously described plate elements can be augmented with a constant triangle in-plane element which shares nodal degrees of freedom. The result are the BST and BSN shell elements. Combining the membrane and bending elements required a local to global coordinate transformation which remains discontinuous along triangle boundaries. Although both membrane and bending strains are included, the elements themselves are flat and not curved. One of the principle contributions of this work is the idea of using the degrees of freedom of a neighborhood of a mesh face to describe the displacement over an element.

4.3 Subdivision-based Approaches

Subdivision surfaces have become widely used geometric representations of general curved three dimensional boundary models and thin shell objects. Their compactness, generality and hierarchical structure have provided effective mechanisms to support evaluating, querying, editing, fitting, and manipulating three-dimensional geometries in a robust and computationally efficient manner. Subdivision surfaces descriptions are discussed in detail in Chapter 2.2.

Recently, Cirak et. al. [17] have demonstrated the utility of subdivision surface shape functions for use as finite element basis functions for the analysis and simulation of the mechanical deformation of thin shell structures. By construction, the subdivision shape functions provide the necessary C^1 continuity requirements (and H^2 integrability) for representing the solution of the fourth-order equilibrium equations governing the behavior of thin shells. Subdivision model descriptions also provide the rather elegant property of representing both the geometry and the physics of the deformation with the same mathematical description.

When used within a finite element framework, the Kirchhoff-Love model requires basis functions that are everywhere C^1 continuous and have square integrable curvature tensors. Roughly speaking, this requires the surface to be C^2 at all but a finite number of isolated points. Few basis function definitions can guarantee these properties as the model is deformed. Recently subdivision surfaces, specifically Loop's scheme [42, 59], have been shown to have these remarkable properties. Cirak et. al. have demonstrated that subdivision surface representations of thin shell models within the Kirchhoff-Love framework are both appropriate and elegant.

4.4 Thin Shell Geometry

To create a thin shell element, the geometry of the body must be parameterized in a known way. From the parameterization of the body in an original and deformed configuration a measure of local deformation, or strain, may be computed; strain is one of the measures that make up the law of virtual work for continua. Let $\mathbf{x}$ and $\tilde{\mathbf{x}}$ represent the position of a point on the middle surface of a shell in the original and deformed configurations respectively. Here we follow the notation of [17] for ease of comparison. Any point in the volume of the undeformed body is represented by $\mathbf{r}$, which may be written the sum of a position on the middle surface $\mathbf{x}$ and a displacement along a thickness direction $\mathbf{a}_3$. The deformed configuration of the body may be similarly parameterized in terms of $\tilde{\mathbf{r}}$, $\tilde{\mathbf{x}}$, $\tilde{\mathbf{a}}_e$ respectively as shown in Equation 4.1

$$\begin{aligned} \mathbf{r}(\theta^1,\theta^2,\theta^3) &= \mathbf{x}(\theta^1,\theta^2) + \theta^3\mathbf{a}_3(\theta^1,\theta^2), \quad -\frac{h}{2} \le \theta^3 \le \frac{h}{2} \\ \tilde{\mathbf{r}}(\theta^1,\theta^2,\theta^3) &= \tilde{\mathbf{x}}(\theta^1,\theta^2) + \theta^3\tilde{\mathbf{a}}_3(\theta^1,\theta^2), \quad -\frac{h}{2} \le \theta^3 \le \frac{h}{2}. \end{aligned} \tag{4.1}$$

This mapping is illustrated in Figure 4.4. θ^1 and θ^2 are parameters of the middle surface, while θ^3 is a parameter in the thickness direction. The natural covariant surface basis

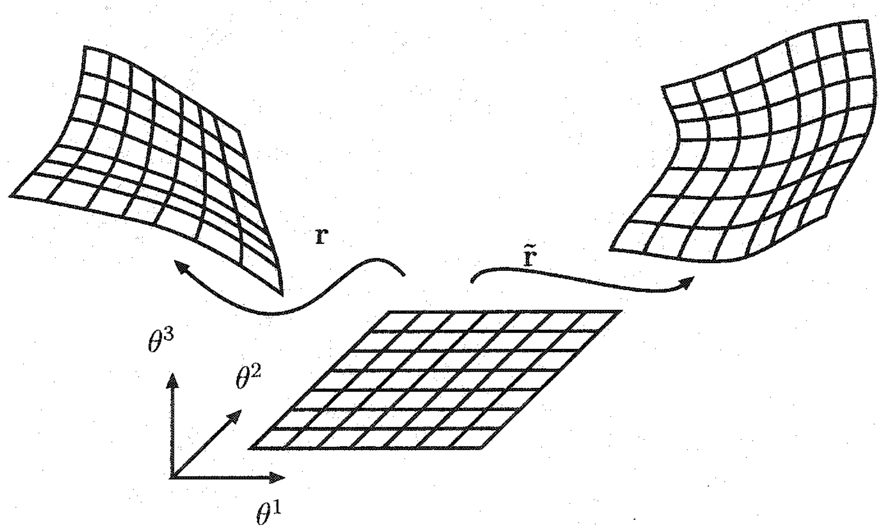


Figure 4.4: Parametric Mapping of Middle Surface

vectors are

$$\mathbf{a}_\alpha = \mathbf{x}_{,\alpha} \qquad \tilde{\mathbf{a}}_\alpha = \tilde{\mathbf{x}}_{,\alpha}. \tag{4.2}$$

The covariant components of the surface metric are defined by

$$a_{\alpha\beta} = \mathbf{a}_\alpha \cdot \mathbf{a}_\beta \qquad\qquad \tilde{a}_{\alpha\beta} = \tilde{\mathbf{a}}_\alpha \cdot \tilde{\mathbf{a}}_\beta. \tag{4.3}$$

The shell director in the reference configuration is defined to be normal to the middle reference surface over the element

$$\mathbf{a}_3 = \frac{\mathbf{a}_1 \times \mathbf{a}_2}{|\mathbf{a}_1 \times \mathbf{a}_2|}. \tag{4.4}$$

4.5 Strain

The covariant basis vectors in the reference and deformed configurations are:

$$\begin{aligned} \mathbf{g}_\alpha &= \frac{\partial \mathbf{r}}{\partial \theta^\alpha} = \mathbf{a}_\alpha + \theta^3 \mathbf{a}_{3,\alpha} \quad & \mathbf{g}_3 &= \frac{\partial \mathbf{r}}{\partial \theta^3} = \mathbf{a}_3 \\ \tilde{\mathbf{g}}_\alpha &= \frac{\partial \tilde{\mathbf{r}}}{\partial \theta^\alpha} = \tilde{\mathbf{a}}_\alpha + \theta^3 \tilde{\mathbf{a}}_{3,\alpha} \quad & \tilde{\mathbf{g}}_3 &= \frac{\partial \tilde{\mathbf{r}}}{\partial \theta^3} = \tilde{\mathbf{a}}_3. \end{aligned} \tag{4.5}$$

The covariant components of the metric tensors are

$$g_{ij} = \mathbf{g}_i \cdot \mathbf{g}_j \qquad\qquad \tilde{g}_{ij} = \tilde{\mathbf{g}}_i \cdot \tilde{\mathbf{g}}_j. \tag{4.6}$$

As before, the Green-Lagrange strain tensor is defined as the difference between the matrix tensors on the deformed and reference configurations of the shell

$$\epsilon_{ij} = \frac{1}{2}(\tilde{g}_{ij} - g_{ij}). \tag{4.7}$$

Let us define

$$\alpha_{ij} = \frac{1}{2}(\tilde{\mathbf{a}}_i \cdot \tilde{\mathbf{a}}_j - \mathbf{a}_i \cdot \mathbf{a}_j) \tag{4.8}$$

$$\beta_{\alpha\beta} = \frac{1}{2}\left(\tilde{\mathbf{a}}_\alpha \cdot \tilde{\mathbf{a}}_{3,\beta} + \tilde{\mathbf{a}}_{3,\alpha} \cdot \tilde{\mathbf{a}}_\beta - \mathbf{a}_\alpha \cdot \mathbf{a}_{3,\beta} - \mathbf{a}_{3,\alpha} \cdot \mathbf{a}_\beta\right). \tag{4.9}$$

Substituting into Equation 4.7 we find, to first order in thickness h ($(\theta^3)^2$ small)

$$\epsilon_{ij} = \alpha_{ij} + \theta^3 \beta_{ij}. \tag{4.10}$$

The membrane strains $\alpha_{\alpha\beta}$ of Equation 4.8, measure deformation in the plane of the surface. The bending strains $\beta_{\alpha\beta}$ of Equation 4.9, measure the change in curvature of the shell. The shearing of the director is measured by $\alpha_{\alpha 3}$, which gives rise to both the Kirchhoff-Love and Reissner-Mindlin theories of shells. Lastly, α_{33} measures the stretching of the director; this effect is often subsumed into plane-stress or plane-strain assumptions.

Kirchhoff-Love theory restricts the director of the deformed configuration to remain normal to the deformed middle surface:

$$\tilde{\mathbf{a}}_3 = \frac{\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2}{|\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2|}. \tag{4.11}$$

β may be algebraically simplified to

$$\beta_{\alpha\beta} = \tilde{\mathbf{a}}_{\alpha,\beta} \cdot \tilde{\mathbf{a}}_3 - \mathbf{a}_{\alpha,\beta} \cdot \mathbf{a}_3 = \tilde{\kappa}_{\alpha\beta} - \kappa_{\alpha\beta}. \tag{4.12}$$

4.5.1 Displacement Formulation

We define our deformation as a displacement of the middle surface in the deformed configuration from the reference

$$\tilde{\mathbf{x}}(\theta^1, \theta^2) = \mathbf{x}(\theta^1, \theta^2) + \mathbf{u}(\theta^1, \theta^2) \tag{4.13}$$

$$\tilde{\mathbf{a}}_\alpha = \mathbf{a}_\alpha + \mathbf{u}_{,\alpha}. \tag{4.14}$$

The covariant components of the surface and curvature vectors in terms of displacement are:

$$\alpha_{\alpha\beta} = \frac{1}{2}(\mathbf{a}_\alpha \cdot \mathbf{u}_{,\beta} + \mathbf{u}_{,\alpha} \cdot \mathbf{a}_\beta + \mathbf{u}_{,\alpha} \cdot \mathbf{u}_{,\beta}) \tag{4.15}$$

$$\beta_{\alpha\beta} = \mathbf{a}_{\alpha,\beta} \cdot \mathbf{a}_3 - (\mathbf{a}_{\alpha,\beta} + \mathbf{u}_{,\alpha\beta}) \cdot \frac{(\mathbf{a}_1 + \mathbf{u}_{,1}) \times (\mathbf{a}_2 + \mathbf{u}_{,2})}{|(\mathbf{a}_1 + \mathbf{u}_{,1}) \times (\mathbf{a}_2 + \mathbf{u}_{,2})|}. \tag{4.16}$$

4.6 Equilibrium

Equilibrium may be found by beginning with the continuous statement of virtual work, just as is Section 3.4:

$$\delta\Pi = \int_V \delta\epsilon_{ij}\sigma^{ij} - \delta\mathbf{u} \cdot \mathbf{f}\, dV = 0 \quad \forall \delta\epsilon_{ij}\, \delta\mathbf{u}. \tag{4.17}$$

Solving for equilibrium is equivalent to finding the zero of the elastic virtual work functional $\delta\Pi(\mathbf{u})$ where strain is given as a function of displacements. Conceptually the zero of the elastic virtual work functional may be found via an iterative method, such as Newton's method. Iterative numerical methods for solving nonlinear problems in solid mechanics will be discussed in in the proposal. Many numerical methods, including Newton's approximate the virtual work functional by a Taylor series. The first order Taylor series expansion of the virtual work functional is

$$\delta\Pi(\mathbf{u}+\delta\mathbf{u}) = \delta\Pi(\mathbf{u}) + \left.\frac{\delta\delta\Pi}{\delta\mathbf{u}}\right|_{\mathbf{u}} \delta\mathbf{u} + O(\delta\mathbf{u}^2). \tag{4.18}$$

The variational derivative of the elastic virtual work functional is

$$\frac{\delta\delta\Pi}{\delta\mathbf{u}}\delta\mathbf{u} = \delta^2\Pi = \int_V \delta\epsilon : \mathsf{C}\delta\epsilon + \delta^2\epsilon : \boldsymbol{\sigma}\, dV, \tag{4.19}$$

where C is the tangent constitutive modulus, defined as

$$\mathsf{C} = \frac{\delta\boldsymbol{\sigma}}{\delta\epsilon}. \tag{4.20}$$

As displacement $\mathbf{u}$ is a continuous field value over the domain of the body, there is little hope in general of solving for $\delta\Pi(\mathbf{u}) = 0$ with the conceptual framework presented here. Discretizing the body into finite elements reduces displacement from a continuous field to a discrete one, allowing the application of numerical techniques.

4.6.1 Membrane Strain

The membrane strain of the shell (changes in area) may be expressed in covariant components as

$$\epsilon_{ij} = \frac{1}{2}\left(\tilde{\mathbf{a}}_i \cdot \tilde{\mathbf{a}}_j - \mathbf{a}_i \cdot \mathbf{a}_j\right) \tag{4.21}$$

$$= \frac{1}{2}\left((\mathbf{a}_i + \mathbf{u}_{,i}) \cdot (\mathbf{a}_j + \mathbf{u}_{,j}) - \mathbf{a}_i \cdot \mathbf{a}_j\right) \tag{4.22}$$

$$= \frac{1}{2}\left(\mathbf{u}_{,i} \cdot \mathbf{a}_j + \mathbf{u}_{,j} \cdot \mathbf{a}_i + \mathbf{u}_{,i} \cdot \mathbf{u}_{,j}\right). \tag{4.23}$$

Membrane Strain First Variation

Following the derivation presented in Section 2.4, the first variation of the membrane strain follows as:

$$\delta\epsilon_{ij} = \frac{1}{2}\left(\delta\mathbf{u}_{,i}\cdot\mathbf{a}_j + \delta\mathbf{u}_{,j}\cdot\mathbf{a}_i + \delta\mathbf{u}_{,i}\cdot\mathbf{u}_{,j} + \delta\mathbf{u}_{,j}\cdot\mathbf{u}_{,i}\right) \tag{4.24}$$

$$= \delta\mathbf{v}_m \cdot \frac{1}{2}\left(N^m_{,i}\left(\mathbf{a}_j + \mathbf{u}_{,j}\right) + N^m_{,j}\left(\mathbf{a}_i + \mathbf{u}_{,i}\right)\right). \tag{4.25}$$

Membrane Second Variation

The second variations of membrane strain with respect to virtual displacements become

$$\delta^2\epsilon_{ij} = \frac{1}{2}\left(\delta\mathbf{u}_{,i}\cdot\delta\mathbf{u}_{,j} + \delta\mathbf{u}_{,j}\cdot\delta\mathbf{u}_{,i}\right) \tag{4.26}$$

$$= \delta\mathbf{v}_m \cdot \frac{1}{2}\left(N^m_{,i}\mathbf{1}N^n_{,j} + N^m_{,j}\mathbf{1}N^n_{,i}\right)\delta\mathbf{v}_n. \tag{4.27}$$

4.6.2 *Discretized Energy*

The formula for virtual work and its variation are expressed in terms of basis functions and displacements as:

$$\delta\Pi = \delta\mathbf{v}_m \cdot \int_V \frac{1}{2}\left[N^m_{,i}\left(\mathbf{a}_j + \mathbf{u}_{,j}\right) + N^m_{,j}\left(\mathbf{a}_i + \mathbf{u}_{,i}\right)\right]\sigma^{ij} - N^m\mathbf{f}\,dV \tag{4.28}$$

$$\delta^2\Pi = \delta\mathbf{v}_m \cdot \mathbf{K}^{mn}\mathbf{v}_n. \tag{4.29}$$

By Equation 2.31 the components of the membrane stiffness matrix may be expressed as

$$\mathbf{K}_{mn} = \int_V \frac{1}{4}\left[N^m_{,i}\left(\mathbf{a}_j + \mathbf{u}_{,j}\right) + N^m_{,j}\left(\mathbf{a}_i + \mathbf{u}_{,i}\right)\right]\otimes$$
$$\mathsf{C}^{ijkl}\left[N^n_{,k}\left(\mathbf{a}_l + \mathbf{u}_{,l}\right) + N^n_{,l}\left(\mathbf{a}_k + \mathbf{u}_{,k}\right)\right]$$
$$+\frac{1}{2}\left(N^m_{,i}\mathbf{1}N^n_{,j} + N^m_{,j}\mathbf{1}N^n_{,i}\right)\sigma^{ij}\,dV. \tag{4.30}$$

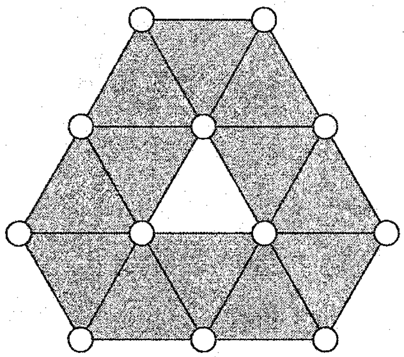
The membrane portion of the stiffness matrix may be readily computed from the preceding equation. The derivation for the bending part is more involved and is presented in the Appendix A.2.

4.7 Element Design and Evaluation

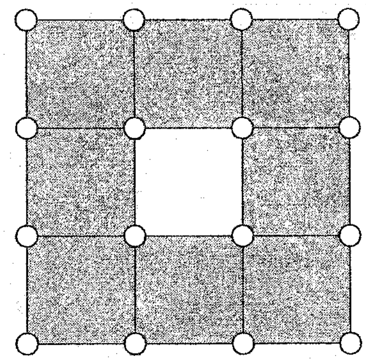
The preceding equations, and those given in Section A.2 give the general form of the stiffness matrix and vector of unbalanced forces in terms of the basis functions N^m, but do not specifically take into account the form of the subdivision basis functions, and the practical matters of using them in a finite element framework. By their nature, the subdivision basis functions are not as straightforward to work with as standard finite element basis functions; their domain extends over a neighborhood of elements, and their very definition depends on the local topology of the body they represent. This section describes the implementation of subdivision elements that not only makes use of local neighborhood connectivity, but allow for efficient navigation of a multilevel hierarchy.

Finite element approaches approximate a continuous domain by a mesh consisting of many discrete pieces. For thin shells, the mesh will represent a surface consisting of faces, edges and vertices. In traditional approaches the basis functions which are used to describe strain over each face in the mesh are associated only with nodes corresponding to the vertices of that face; that is the displacement field over any element is completely described by the values at the nodes which are part of the current face. In these approaches there exists a one-to-one correspondence between elements and mesh faces. As interpolating basis functions are commonly used, the mesh face provides a piecewise geometric representation of the model surface.

In a subdivision-based finite element, the basis functions which have support over an element are those which correspond to nodes in the neighborhood of the current face of the control mesh. Nodes in the neighborhood are those that are part of any face topologically adjacent to the current control mesh face. Example neighborhoods for both triangular and quadrilateral elements are illustrated in Figure 4.5. The white face in the center is denoted to be the *central* face. The central face represents topologically the area over which the basis functions of the element are parameterized. The gray areas themselves are not important, but the nodes which they connect have influence over the element. For a valid control mesh,



(a) Triangular Element



(b) Quadrilateral Element

Figure 4.5: Subdivision Element Neighborhoods

the geometry parameterized over each central face is disjoint from the geometry associated with all other central faces. The union of the parameterized geometry associated with each central face fully describes the surface of the model. Invalid control meshes include those that have degenerate faces, or otherwise form limit surfaces which are not orientable manifolds and are not considered in this work. A manifold, but not orientable model, such as a Möbius strip could be modeled as a non-closed geometry with constraints enforcing boundaries to align properly (constraints are discussed in Chapter 5).

The concept of a finite element needs to be expanded when non-local basis functions and extended neighborhoods are used. The term "finite element" traditionally refers to a discrete part of a whole. As the neighborhoods of nearby central faces overlap one another, it is natural to associate each element with it's central face; thus there is one element for each interior face in the control mesh even though the displacement field over the element is influenced by control values outside of the central face. From this point on in the document, the term "element" will refer to the central face over which the basis functions are parameterized.

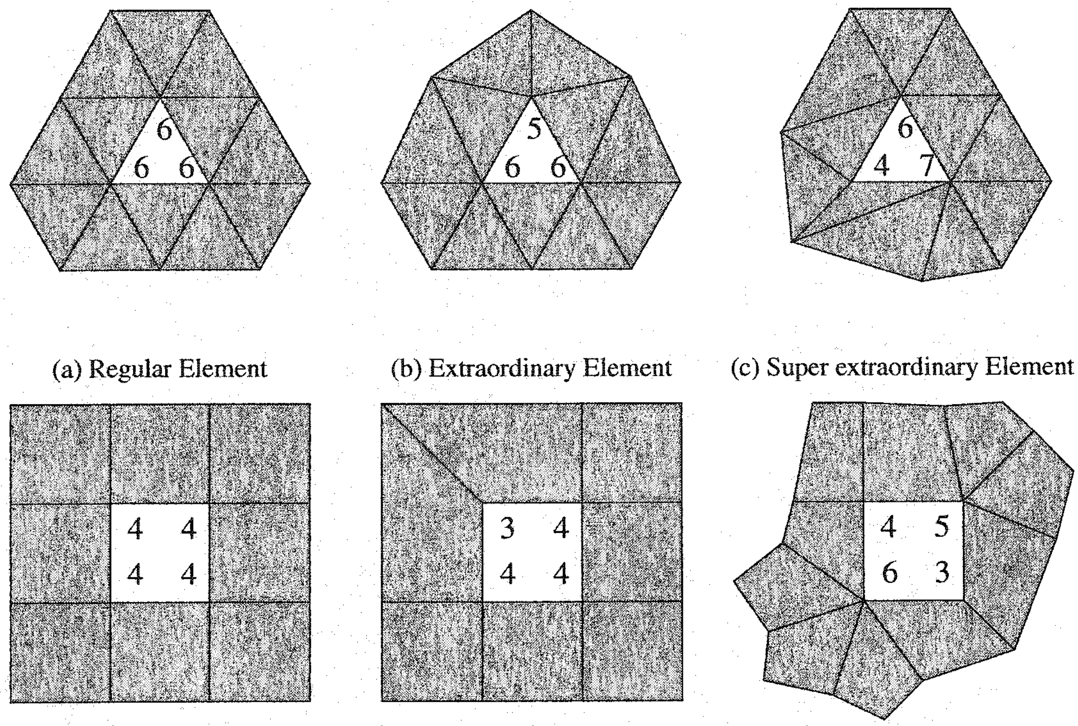
Evaluation of the global vector of unbalanced forces $\mathcal{Q}$ and tangent stiffness matrix $\mathcal{K}$ of Equation 2.30 can be accomplished very efficiently in a piecewise fashion. Each entry of $\mathcal{Q}$ and $\mathcal{K}$ involves products of basis functions corresponding to different nodes throughout the model. Because the basis functions weighting the nodes have compact support, the product of basis functions which have non-overlapping support are identically zero and may be ignored. The most straightforward manner to evaluate $\mathcal{Q}$ and $\mathcal{K}$ is to integrate locally over each element of the mesh. The global $\mathcal{Q}$ and $\mathcal{K}$ may then be assembled by summing the local element-wise calculations.

Determining the neighborhood of an element is an important step in determining the solution, and is repeated once for each element for each iteration of the solver. After the neighborhood of the current element has been found, computation of the local vector of unbalanced forces and tangent stiffness matrix for the current element may be restricted to only those basis functions corresponding to nodes within the neighborhood. The specific definition of the basis functions describing each element depend on the valence of the vertices of the current element. This contrasts to standard FEM formulations is which the basis functions are defined based on the type of element only without respect to the topology of neighborhoods. In fact, subdivision approaches can be thought to define an infinite set of specific elements, those that correspond to every possible configuration of nodal valences.

Schemes for evaluating the basis functions over each element depend on the specific connectivity of the associated control mesh face. Three distinct schemes are necessary depending on the connectivity of the face, and elements may be classified by the scheme used. Figure 4.6 show examples of the three types of connectivities for triangular and quadrilateral elements respectively. Regular connectivities ensure that each mesh vertex has a valence appropriate to tile a plane (6 for triangles, 4 for quadrilaterals). Extraordinary faces contain exactly one non-regular valence vertex, and super extraordinary faces contain two or more.

4.7.1 Regular Elements

Figure 4.6(a) shows a triangular element with regular connectivity. Each vertex of the central face has valence 6 (6 connected nodes). If each vertex of each triangle in a control mesh has valence 6, the mesh may repeatably tile a plane. Similarly quadrilateral elements



(a) Regular Element (b) Extraordinary Element (c) Super extraordinary Element

(d) Regular Element (e) Extraordinary Element (f) Super extraordinary Element

Figure 4.6: Types Of Triangular and Quadrilateral Elements

posses regular connectivity if each vertex has valence 4, as shown in Figure 4.6(d). These elements are known as regular elements. The basis functions over these elements are simply defined by known tensor product polynomials which do not change definition over the element, making them comparatively simple to evaluate.

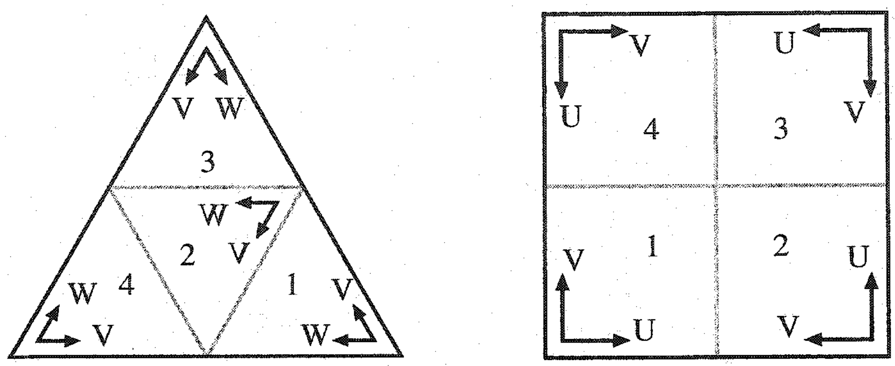
4.7.2 Extraordinary Elements

If the connectivity is not regular then the basis functions will be linear combinations of the regular tensor product basis functions. The specific linear combination depends on both the parametric location evaluated and the local connectivity of the face (see Section 2.2 and Section 2.2.2 for details about the subdivision basis functions and Stam's evaluation scheme). Elements that contain only one non regular node are called extraordinary; Figures 4.6(b) and 4.6(e) show representative extraordinary elements. Prior literature has shown how to evaluate the basis functions for the special case of the Barycenter of triangular elements [17]. More recently, Stam [58] has presented an efficient algorithm to evaluate the subdivision basis functions for faces which contain only one node with non-regular connectivity, such as the ones shown in Figure 4.6(b) and 4.6(e). This approach is used for extraordinary elements in this work.

4.7.3 Super Extraordinary Elements

Those elements that contain more than one non-regular node are called super extraordinary. Figures 4.6(a) and 4.6(d) show elements containing two or more non-regular valence vertices. The basis functions over these elements may be evaluated using a novel, extension to Stam's approach. One step of subdivision isolates non-regular vertices, ensuring that all faces in a once-subdivided control mesh may be evaluated using Stam's technique. The implementation in [17] required that all meshes be subdivided at least once ensuring that no super extraordinary faces be present. In a multilevel hierarchy scheme it is important that the coarsest level be efficient to solve on directly. Requiring that the coarsest level be once-subdivided can greatly reduce the efficiency of a multilevel solver, such as that described in Chapter 6, by increasing the complexity of the coarsest level. Thus the ability to evaluate super extraordinary faces is quite important, and worth the added cost of implementation.

Our implementation leverages the benefits of synthetic subdivision, inspired by the approach of Stam. Were the super extraordinary element subdivided once, the basis functions of each of its child elements will either be regular or extraordinary, and may be evaluated by the techniques described above. Instead of physically subdividing the mesh, a local subdivision matrix S may be created which relates control values of a super extraordinary element to the control values of its children. The important aspect of synthetic subdivision for evaluating basis functions is the mapping between the super extraordinary parent element and its children.



(a) Triangular Element (b) Quadrilateral Element

Figure 4.7: Super Extraordinary Element Parameterization

Figure 4.7 shows triangular and quadrilateral parent elements (outlined in black) which have been once subdivided each into four child elements. The parent element is assumed to be parameterized in the standard way with $u = 1, v = 0, w = 0$ in the bottom left corner for the triangle element, and $u = 0, v = 0$ in the bottom left corner for the quadrilateral element. The mapping for the child elements is shown pictorially in Figures 4.7(a) and 4.7(b) and described in more detail in tables 4.1 and 4.2 for triangular elements and quadrilateral elements respectively. In each case the parent element is parameterized with lowercase variables and child faces are parameterized with uppercase variables. The first column of the Tables 4.1 and 4.2 gives the number of the sub-region (child face) corresponding to Figures 4.7(a) and 4.7(b). The second column denotes the parametric sub-region of the parent element which each child element corresponds to. The third column denotes the parent element edge along which corresponds to the standard parameterization. LNext() is used to designate the next edge counterclockwise from the bottom edge of the parent element, and LPrev() the previous; this information is useful for implementation purposes. The fourth column gives a linear mapping between the parent and child parameterizations. The fifth column denotes the Jacobian of the transformation in the form $[\frac{\partial V}{\partial v}, \frac{\partial V}{\partial w}; \frac{\partial W}{\partial v}, \frac{\partial W}{\partial w}]$ for triangles and $[\frac{\partial U}{\partial u}, \frac{\partial U}{\partial v}; \frac{\partial V}{\partial u}, \frac{\partial V}{\partial v}]$ for quadrilaterals.

The mappings presented allow the basis functions over a super extraordinary element to

Table 4.1: Triangular Child Element Parameterization

Face	Region	Base Edge	Transformation	Jacobian
1	$v \geq \frac{1}{2}$	LNext()	$V \Leftarrow 2w,\ W \Leftarrow 2(1-v-w)$	$[0,2;-2,-2]$
2	otherwise	$(seetext)$	$V \Leftarrow 1-2v,\ W \Leftarrow 1-2w$	$[-2,0;0,-2]$
3	$w \geq \frac{1}{2}$	LPrev()	$V \Leftarrow 2(1-v-w),\ W \Leftarrow 2v$	$[-2,-2;2,0]$
4	$u \geq \frac{1}{2}$	Self	$V \Leftarrow 2v,\ W \Leftarrow 2w$	$[2,0;0,2]$

Table 4.2: Quadrilateral Child Element Parameterization

Face	Region	Base Edge	Transformation	Jacobian
1	$u < \frac{1}{2},\ v < \frac{1}{2}$	Self	$U \Leftarrow 2u,\ V \Leftarrow 2v$	$[2,0;0,2]$
2	$u \geq \frac{1}{2},\ v < \frac{1}{2}$	LNext()	$U \Leftarrow 2v,\ V \Leftarrow 2-2u$	$[0,2;-2,0]$
3	$u \geq \frac{1}{2},\ v \geq \frac{1}{2}$	LNext().LNext()	$U \Leftarrow 2-2u,\ V \Leftarrow 2-2v$	$[-2,0;0,-2]$
4	$u < \frac{1}{2},\ v \geq \frac{1}{2}$	LPrev()	$U \Leftarrow 2-2v,\ V \Leftarrow 2u$	$[0,-2;2,0]$

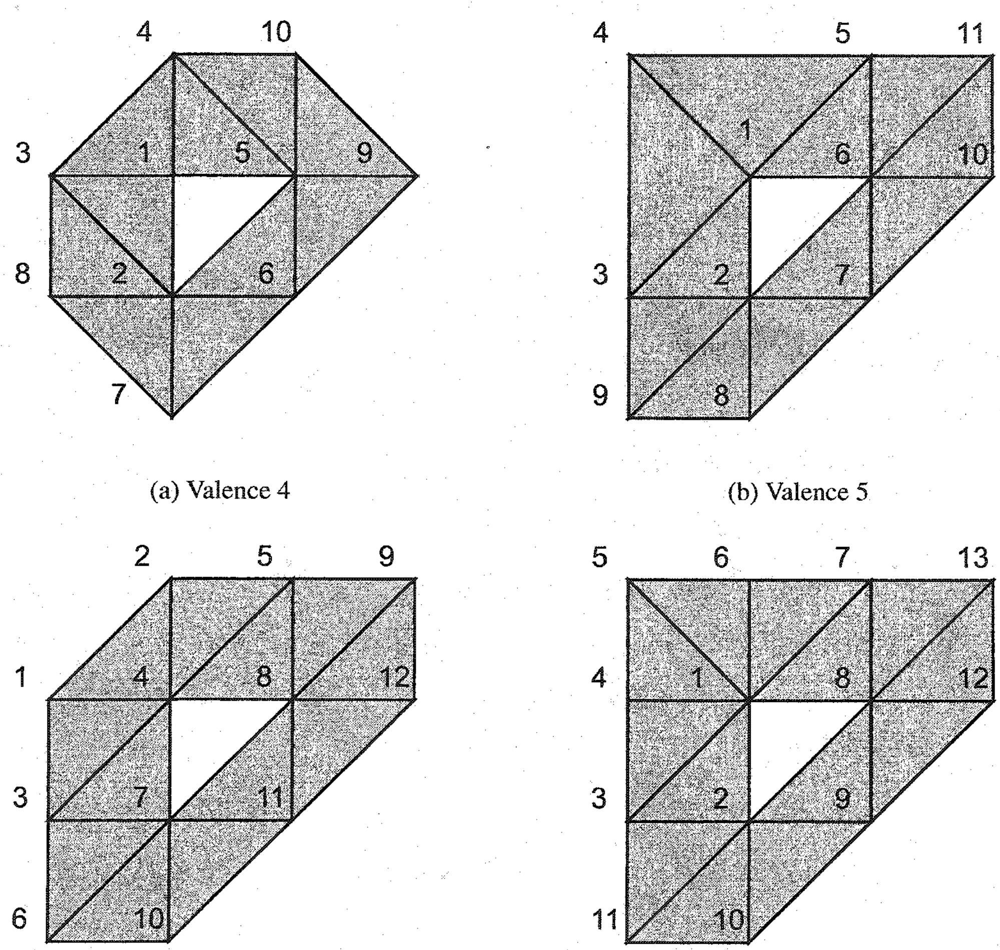
be implemented as either regular or extraordinary calculations on a once-subdivided mesh, which need not be physically subdivided in practice. Evaluating derivatives takes care and requires the Jacobian of the mapping. Because all the mappings are linear, the Hessian is trivially zero and is not required. Finite element basis functions N_i each correspond to a nodal degree of freedom $\mathbf{v}_i$. The synthetic subdivision process must be designed to ensure that this relationship remains appropriate.

4.8 Subdivision Basis Functions

This section presents the subdivision basis functions for a variety of configurations. Unlike more traditional finite element basis functions, the shape and number of subdivision basis functions depends on the local connectivity of the subdivision control mesh. Elements can be classified into three distinct types depending on their local connectivity. These types correspond to the technique required to evaluate the basis functions and their derivatives. Regular, extraordinary and super extraordinary elements are discussed in sections 4.7.1 4.7.2 and 4.7.3 respectively. As the total number of basis functions changes, the numbering of the basis functions does not remain constant for different connectivities. Numberings for several connectivities of the loop element are shown in Figure 4.8. These numberings are consistent with those of Stam [59].

The numberings for the extraordinary elements (valences 4,5,7) begin with the extraordinary vertex and then continue in a counter-clockwise fashion to number. This accounts for $n+1$ of the vertices, where n is the valence of the single extraordinary vertex. The total number of vertices in the neighborhood of an extraordinary element is always $n+6$. The remaining five vertices are then numbered in a consistent fashion. The regular element (valence 6) is numbered in a different manner and is consistent with prior convention. Because regular connectivities were understood far in advance of extraordinary cases, a different convention was adopted.

The basis functions for three extraordinary elements are shown in figures 4.9, 4.10, and 4.12. The basis functions for the regular element are shown in Figure 4.11. These arrangements are correspond to the ordering of Figure 4.8 are are numbered from left to right, top to bottom. For the Loop element there are always three basis functions of much larger magnitude than the others. These are the basis functions corresponding to the three vertices which make up the face of the innermost triangle. For the regular element this corresponds to control vertices/basis functions numbers 4, 7, and 8 of Figure 4.8(c). Because these control vertices are topologically closer to all points on the parameterized element, they exert more effect on the shape and thus are associated with basis functions of higher magnitude. The highest magnitude of any basis function is $\frac{1}{2}$; the basis functions do sum to unity at all points, but because the basis is not interpolating no basis function ever attains a unit value. The basis functions for super extraordinary elements are scaled combinations of the extraordinary basis functions, and are computed individually for each super extraordinary element as shown in Section 4.7.3.



(a) Valence 4

(b) Valence 5

(c) Valence 6

(d) Valence 7

Figure 4.8: Loop Basis Function Arrangement

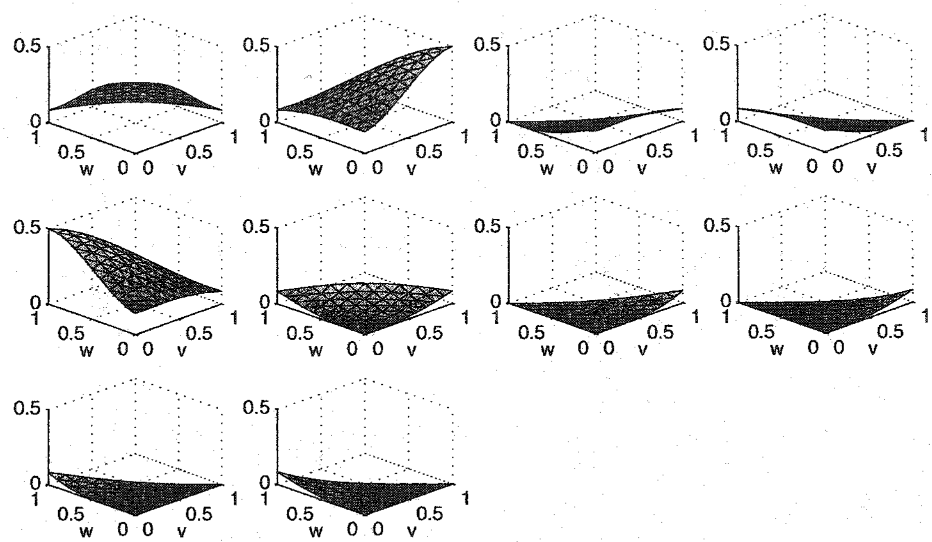


Figure 4.9: Loop Basis Functions, Valence 4

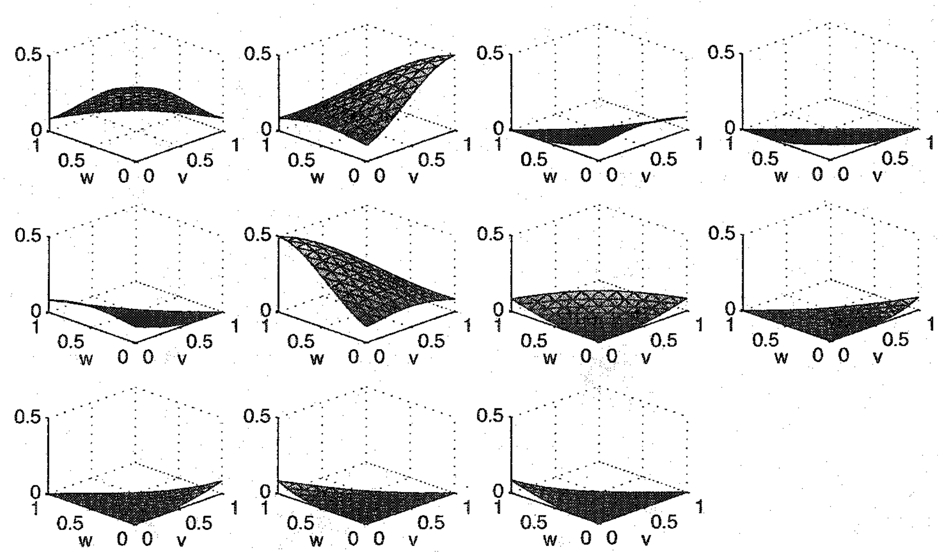


Figure 4.10: Loop Basis Functions, Valence 5

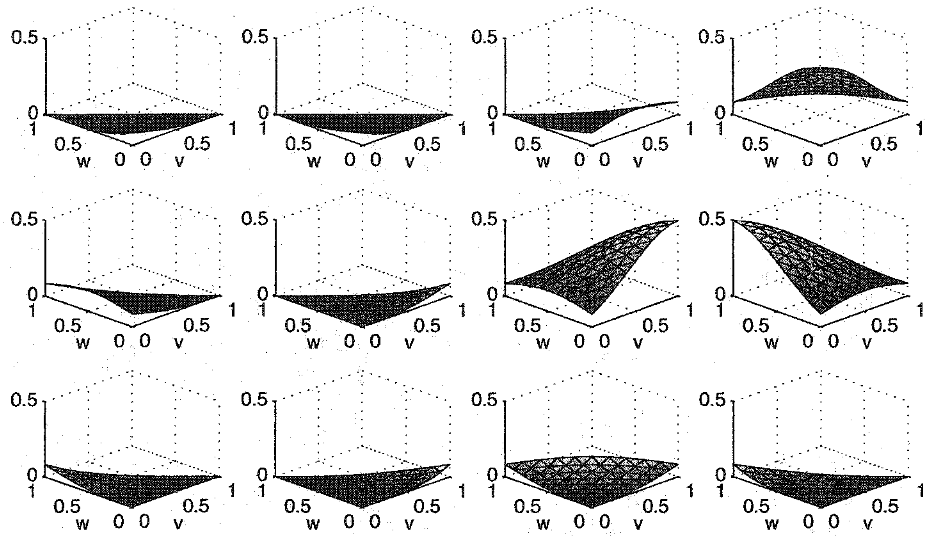


Figure 4.11: Loop Basis Functions, Valence 6

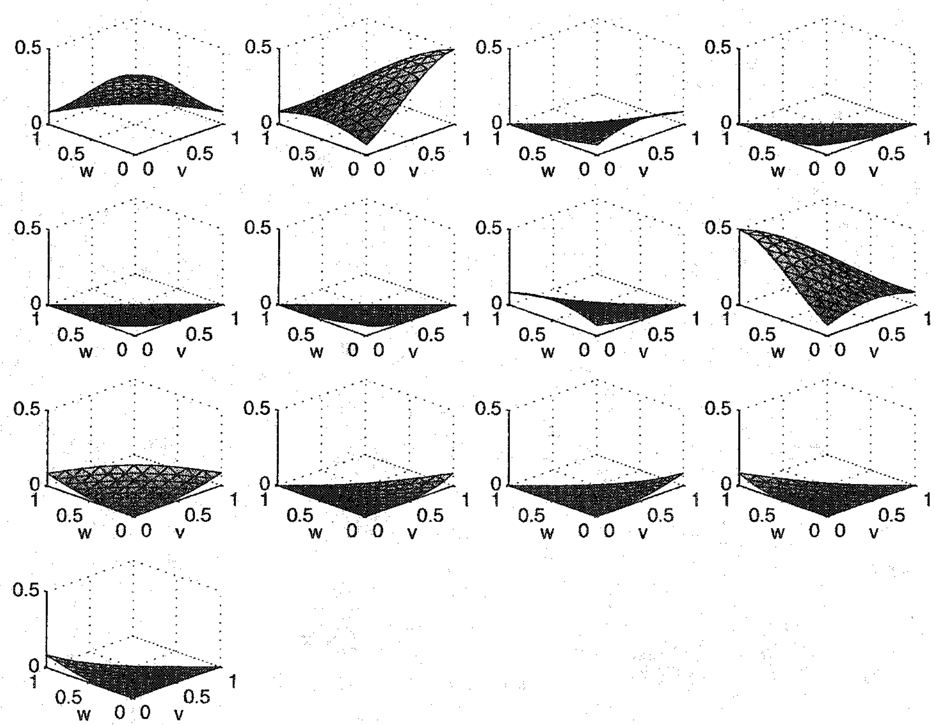


Figure 4.12: Loop Basis Functions, Valence 7

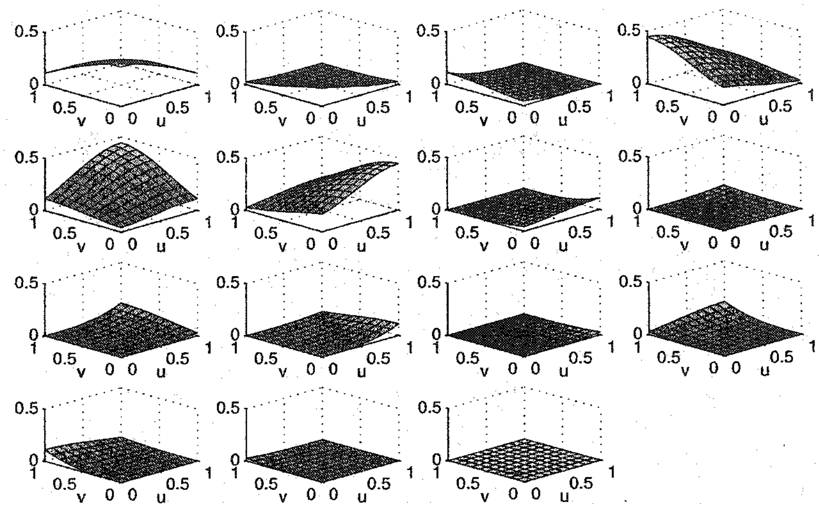


Figure 4.13: Catmull-Clark Basis Functions, Valence 3

The basis functions for the Catmull-Clark quadrilateral element are illustrated in figures 4.13, 4.14, 4.15 and 4.16 for valences 3,4,5 and 6 respectively. For these elements, valence 4 is regular, and there are a total of $n + 12$ vertices in the neighborhood of the element. Similar to the Loop basis functions, the four control vertices corresponding to the innermost quadrilateral are associated basis functions of the largest magnitude. The numbering scheme for these elements is also similar to the case of the Loop elements, and is consistent with that of Stam [58]. It is worth noting that the preceding figures for both quadrilateral and triangular elements are straightforward plots of the actual basis functions, and are not projected onto the eigenbasis as in Stam's work [58, 59].

4.9 Element Distribution

The ratio of regular elements to non-regular elements increases as a mesh is subdivided. Subdividing a control mesh once introduces four times (modulo boundary effects) the number of vertices and faces into the model for both triangular and quadrilateral schemes. This effect is shown in Figure 4.17. An initial control mesh may consist of an arbitrary number of regular and non-regular vertices, which leads to an arbitrary initial number of regular and

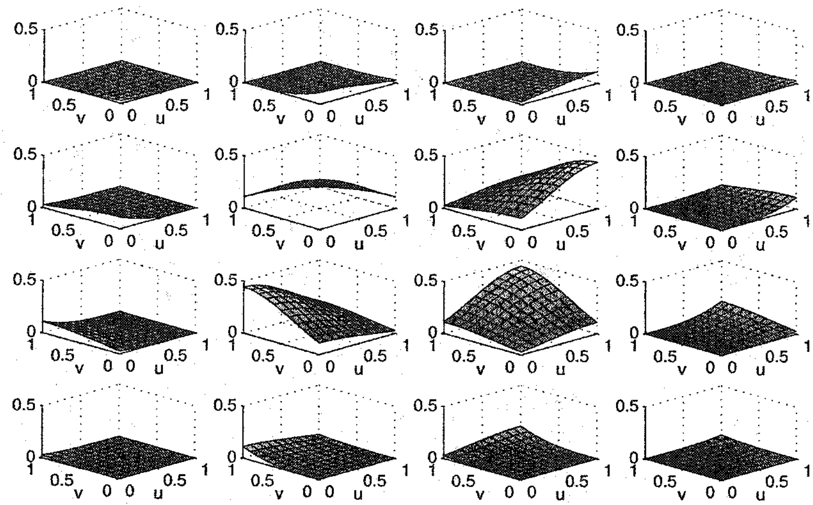


Figure 4.14: Catmull-Clark Basis Functions, Valence 4

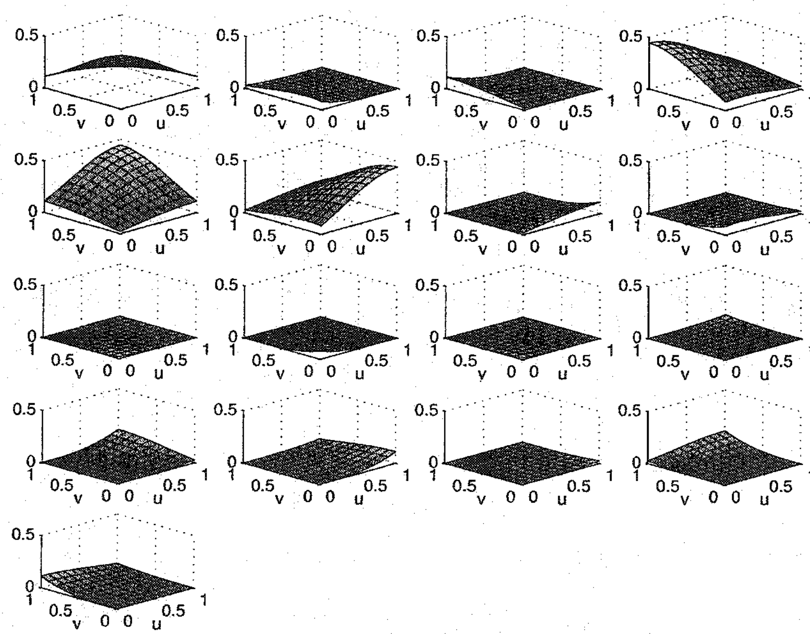


Figure 4.15: Catmull-Clark Basis Functions, Valence 5

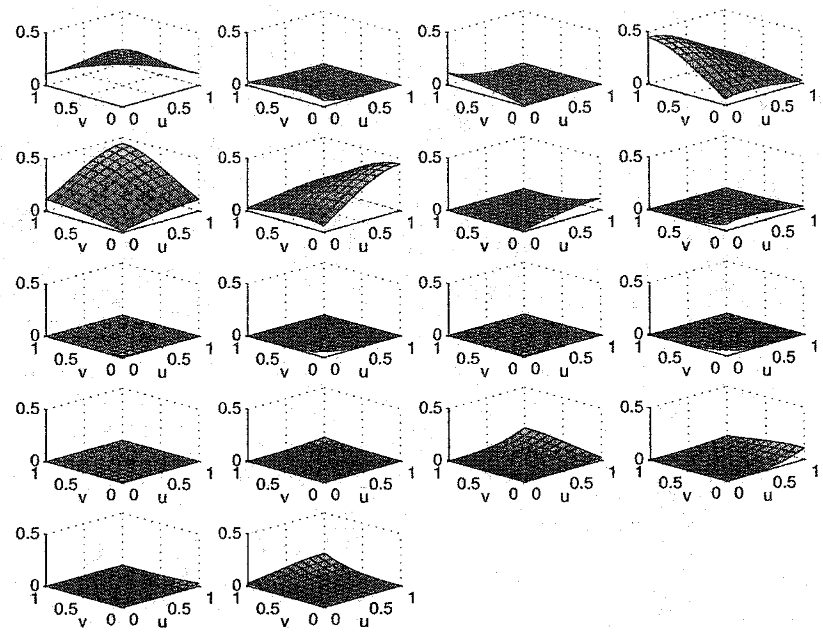
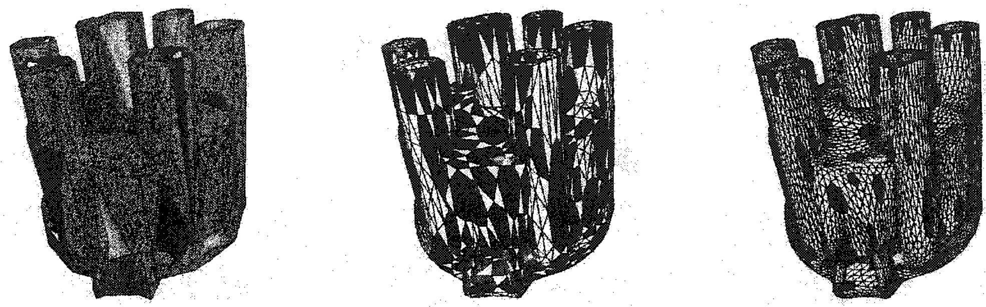


Figure 4.16: Catmull-Clark Basis Functions, Valence 6

non-regular faces. Figure 4.17(a) shows a subdivision model of a distributor cap with faces colored corresponding to element type. Extraordinary faces are colored black and contain exactly one extraordinary vertex. Super extraordinary faces are colored green (gray); most faces of this mesh are super extraordinary. Regular faces, of which there are very few, are colored white. Figure 4.17(b) shows the same control mesh after one level of subdivision. The mesh now consists of several white regular faces and black extraordinary faces. The subdivision process guarantees that super extraordinary faces no longer exist after one level of subdivision. Figure 4.17(c) shows the same control mesh after a second subdivision. Again this new control mesh consists only of regular faces and extraordinary ones. The regular faces greatly outnumber the extraordinary, and repeated subdivision would only enhance this disparity.



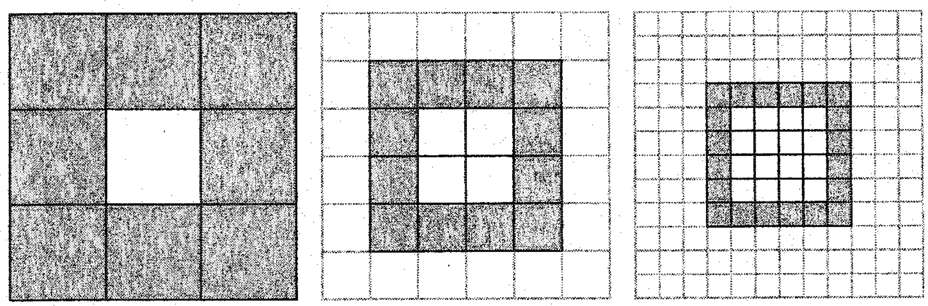
(a) Initial Mesh (b) One Subdivision (c) Two Subdivisions

Figure 4.17: Element Type Distribution

4.10 Multilevel Elements

The preceding sections of this chapter have been devoted to describing a thin-shell subdivision element and its use in representing the geometry, deformation, and mechanics of thin bodies. It is one thing to state the discretized equations of motion for a system, quite another to solve them. The accuracy of a numerical simulation is a function of the quality of the discretization of the system. Simulations using coarse discretizations with just a few degrees of freedom can be solved quickly, but may yield poor results. Fine discretizations are often more accurate but there is an associated increase in computational cost. Multiresolution solvers can mitigate the cost associated with increased accuracy by considering a hierarchy of refinements of the model instead of only a single resolution. The idea behind multiresolution (or multi-level) approaches are to accelerate the convergence of numerical simulations by propagating knowledge between coarsely discretized versions of a model where solutions may be obtained quickly and finely discretized representations where solutions may be obtained more accurately. A well designed multiresolution algorithm may achieve a significant savings in computational effort for a given accuracy.

Chapter 6 is devoted to describing a Multigrid-Preconditioned Conjugate Gradient algorithm for efficiently solving subdivision surface finite element problems. The key to this approach is an efficient means to propagate information between neighboring levels. In this section and Section 4.11, we describe a multi-level extension of the subdivision element suitable for describing a hierarchy of model discretizations.

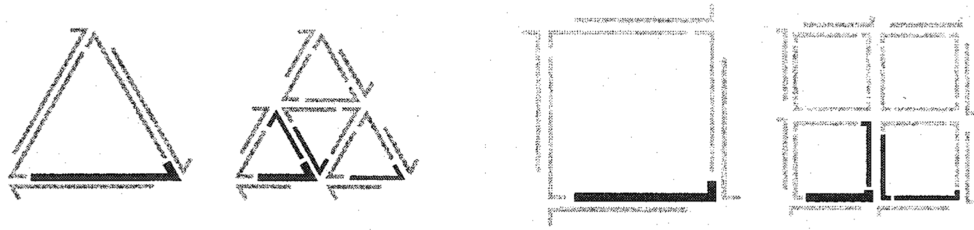


(a) Initial Mesh (b) One Subdivision (c) Two Subdivisions

Figure 4.18: Quadrilateral Element Multilevel Hierarchy

In a multilevel hierarchy, element neighborhoods must be related to their corresponding regions in coarser and finer discretizations of the model. Figure 4.18 shows a simple multilevel subdivision hierarchy for a quadrilateral element. The central element (white) and its neighborhood (gray) are shown in Figure 4.18(a). Figure 4.18(b) shows the four child central elements (white) and their combined neighborhood (gray). Each child central element is also a neighbor for other central elements; no one color scheme is fully appropriate for this image. Faces outlined in gray are children of the neighbor faces of the coarsest level, but do not contribute to the neighborhoods of any of the child central elements. Figure 4.18(c) illustrates this process extended once more. The white faces represent children of the central faces of the level above, grandchildren of the coarsest central face.

It is important that the neighborhood computation process be efficient; a pair-wise search for neighborhoods would limit the order of the algorithm to no better than $O(N^2)$, a serious detriment. An efficient data structure to describe local neighborhoods across levels of a hierarchy is described in Section 4.11. This data structure also aids in constructing the local subdivision matrices and parent child relationships for the evaluation of super extraordinary elements.



(a) Triangular (b) Quadrilateral

Figure 4.19: Hierarchy Representation

4.11 Hierarchy Traversal

Mesh traversal is implemented using a novel multilevel extension to the quad-edge data structure. The quad-edge is a powerful structure for storing boundary manifolds of arbitrary topology [34]. It allows constant time access to all immediate neighbors of a vertex, face, or edge by storing both the manifold edge graph and it's dual. We have modified the structure to allow for the explicit storage of subdivision hierarchies by adding a parent and child field to each directed edge. Each non boundary directed edge is related to four children, and each child is assigned a single parent.

The child edge that shares the same origin and direction as its parent is known as the primary child, and all other child edges may be derived from it in constant time, thus requiring only one child pointer. Figure 4.19 shows a triangle of a mesh, formed of half-edges, before and after one step of subdivision; the parent edge and the primary child are indicated in extra-bold, the non primary children in bold. The convention for the primary/non-primary children was chosen because it is applicable to both triangular and quadrilateral 4:1 subdivision schemes.

The primary child is always a directed edge of the child element in the lower-left. The three remaining child edges are obtained easily using the Quad-Edge data structure as

- Edge().PrimaryChild().LNext()

- Edge().PrimaryChild().LNext().Sym()

- Edge().Sym().PrimaryChild().Sym()

for both the triangular and quadrilateral subdivision schemes. The union of the children of the parent element will be the union of the children of all of the parent elements. The children of each edge of the parent element are unique and are not shared between parent edges.

The similarity between figures 4.19 and 4.7 should not be overlooked. The chosen hierarchy scheme allows for easier evaluation of super extraordinary shell elements because of the similarity between the parent-child relationship of the hierarchy operator and that of the super extraordinary parameterization.

This hierarchy scheme is useful for generating the global subdivision operator $\mathrm{S}_{\mathrm{j}}^{\mathrm{j+1}}$ in constant time per vertex. Traditionally the global subdivision operator is used as an analysis tool and is rarely explicitly formed in practice for producing subdivision geometry, in favor of walking the mesh as it is generally needed only once [20]; however, we find it advantageous to explicitly represent this operator in a sparse matrix data structure as it is applied to the model several times in executing the multigrid preconditioned cojugate gradient algorithm described in Chapter 6.

4.12 Summary

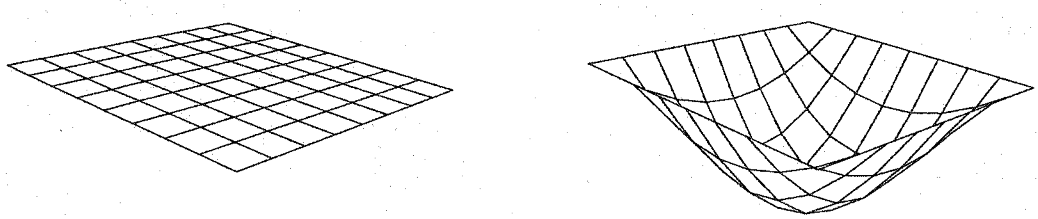
This chapter describes a multilevel rotation-free thin shell finite element. The derivation of the mechanics of thin shells and the equations of equilibrium are shown for the subdivision element. We classified elements into three distinct types corresponding to the differing evaluation techniques needed and shown how the element may be evaluated in each of these situations. We created a unified data structure for meshes with both quadrilateral and triangular connectivity; this approach allows either formulation to be used without minimal additional implementation effort. Our data structure allows for efficient information transfer between levels of a multiresolution hierarchy. The accuracy of this element is demonstrated in Chapter 5, and its efficiency in Chapter 6.

Chapter 5

CONSTRAINTS

5.1 Introduction

This chapter presents a new method for enforcing boundary conditions within subdivision surface finite element simulations of thin shells. The proposed framework is demonstrated to be second order accurate for displacements with respect to increasing refinement for simply-supported and clamped boundary conditions. Second order accuracy on the boundary is consistent with the accuracy of subdivision based approaches for the interior of a body. Our proposed framework is applicable to both triangular and quadrilateral refinement schemes, and does not impose any topological requirements upon the underlying subdivision control mesh. Several examples from the Belytschko *et. al.* [9] obstacle course of benchmark problems are used to demonstrate the convergence of the scheme.



(a) Reference (b) Deformed

Figure 5.1: Simply Supported Flat Plate Under Gravity Load

Figure 5.1 shows a discretized approximation to a simply supported flat plate under transverse load in its reference and deformed configurations. The simply supported boundary conditions require that all four edges of the plate do not change position during deformation. Closed form solutions rarely exist for general thin shell boundary value problems (the above example being a rare exception), and numerical techniques such as finite element analysis are required to find solutions. Numerical methods must quickly converge to a solution to avoid excess computational expense. Convergence of any numerical method is limited by the weakest link in the chain of numerical approximations. Thin shell subdivision element simulations using Loop or Catmull-Clark bases are capable of converging with quadratic accuracy in the displacement error. It is important that the boundary conditions used also converge with at least quadratic accuracy in order to not inhibit the overall convergence of the simulation.

We propose a new formulation for applying boundary conditions to thin shell subdivision element models. Subdivision element simulations are capable of converging with second order accuracy in the displacement error. Previously proposed boundary conditions for thin shell models converge at a less favorable rate. The constraints we describe within this work exhibit second order convergence in displacement. Our proposed framework builds on the ideas first presented by Halstead *et. al.* [35] for fairing subdivision surfaces. These constraints may be applied to any point on the boundary or interior of the body with no restrictions on the connectivity or representation of the discretized model with demonstrated second order accuracy.

The remainder of this chapter is organized as follows. Section 5.2 discusses methods of boundary representation for thin shell subdivision element simulation, prior approaches to boundary modeling and our new framework for defining boundary conditions. Convergence results are presented in Section 5.3. Relevant information, including a listing of parameters for each of the benchmark problems are given in the Appendices A.3, and A.4.

5.2 Boundary Conditions

This section describes both a general formulation and specific instances of commonly used boundary conditions and their implementation for subdivision modeling. It is important that the convergence rates on the interior and along the boundary of a body be equal for a chosen solution scheme, otherwise one or the other will limit the total solution accuracy and become the "Achilles heel" of the method as a whole. Subdivison elements are capable of converging with quadratic accuracy in displacement over the interior of thin bodies. Our proposed formulation retains this order of convergence along the boundary of a model. Verification of the convergence properties of our framework applied to a series of benchmark problems is provided in Section 5.3.

The solution of the boundary value problem corresponding to the discretization of Equation 7.3, requires imposing appropriate boundary conditions. The most commonly used boundary conditions in plate and shell mechanics restrict either the displacement or rotation (or both) of a thin body along an edge. Boundary conditions which prevent displacement along an edge are known as simple supports. Those which constrain both displacement and rotation to be zero are labeled clamped boundary conditions. Other boundary conditions, such as symmetry conditions may be derived from these primitives.

5.2.1 Specification of Boundaries

Within the thin shell subdivision element framework the subdivision basis functions serve several purposes including representing both the geometry and the mechanics of the thin shell models. The subdivision basis functions extend over a neighborhood of influence beyond the topological face to which they correspond, unlike traditional finite elements in which the boundary of the element and the boundary of the domain of support of the basis functions associated with the element coincide. The definition of the basis functions used is dependent on the mesh connectivity of the face to which they correspond. The concept of element neighborhoods requires a carefully reasoned and compatible definition of boundaries for properly specifying boundary value problems.

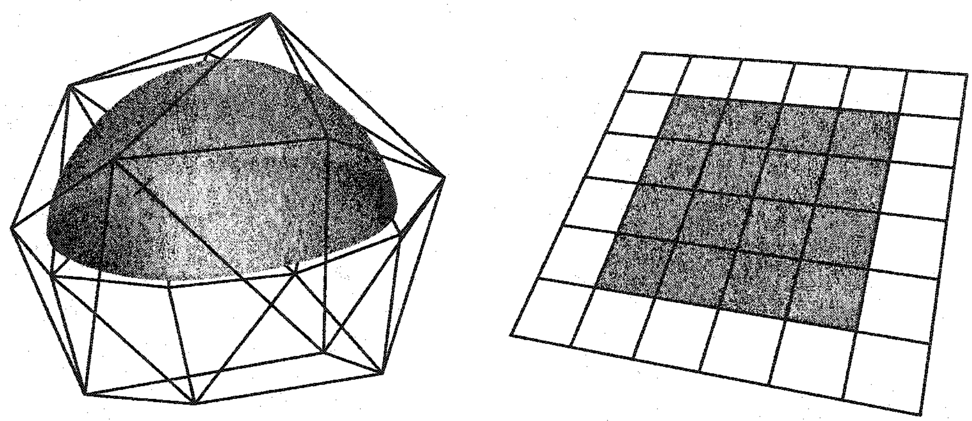
Explicit Boundary Modeling

The use of neighborhood elements allows for extra freedom in defining the geometric boundary of models. One method of specifying the boundary of the model is to not associate parameterized geometry with faces on the boundary of control mesh. This technique will be denoted as "explicit boundary modeling." The control mesh of the model (and any interior holes) are surrounded by a ring of extra faces over which geometry is not parameterized. In keeping with prior literature, these faces are denoted as "ghost faces," and control mesh boundary vertices as "ghost vertices." Although the ghost faces are not associated directly with parameterized geometry, the ghost vertices still affect the geometry parameterized over the interior faces which include them in their neighborhood. It is the responsibility of the designer in this paradigm to manipulate the control mesh, including the ghost faces, to create the intended geometry along the boundary of the limit surface.

Examples of open meshes with ghost boundary faces are shown in Figure 5.2. The control mesh is drawn as a set of connected edges surrounding the shaded limit surface. The use of boundary faces to describe interior geometry is best illustrated by the flat plate model on the right. Because of the regular geometric ordering of the control mesh, interior faces appear filled while ghost faces are not. Control meshes which are closed, such as those illustrated in Figure 5.3 have no boundary, and require no ghost faces or boundary rules of any kind.

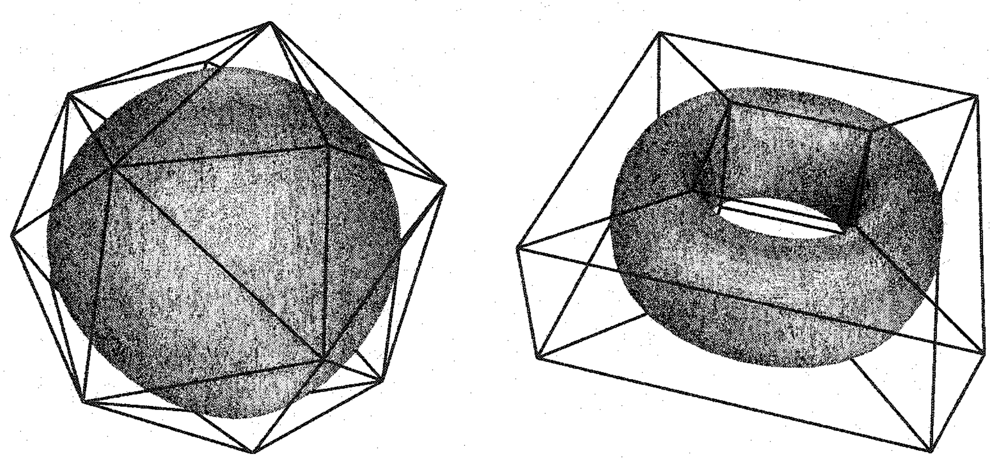
Implied Boundary Modeling

Another approach for describing boundaries, presented in [17], does not require the model designer to explicitly create ghost faces. Instead limit geometry is parameterized over all faces on the control mesh. This technique will be denoted as "implied boundary modeling." The limit geometry associated with faces which lie on the boundary of the control mesh, and therefore do not have complete one neighborhoods, is defined by special subdivision rules. These rules enforce that the boundaries of the limit surface be defined by the same curves



(a) Open Triangular Control Mesh (b) Open Quadrilateral Control Mesh

Figure 5.2: Open Subdivision Models



(a) Closed Triangular Control Mesh (b) Closed Quadrilateral Control Mesh

Figure 5.3: Closed Subdivision Models

that would result from one-dimensional subdivision applied to the boundary. Schweitzer and Duchamp [53] have shown that these conditions may easily be enforced by creating a ring of ghost faces that are derived from the control vertex positions on the boundary of the mesh. In this design paradigm the geometry of the ghost faces is implied and their use is purely of convenience. This technique applies to boundary vertices with regular valence only, forcing modelers to meet this restriction.

These two methods of specifying boundaries for subdivision geometry lead to different approaches for specifying boundary conditions for mechanical simulation. Boundary conditions from prior literature which were developed for control meshes with implied boundaries are discussed in Section 5.2.2. Novel boundary conditions and evaluation techniques for explicit boundary models are described in Sections 5.2.3 - 5.2.5.

5.2.2 Prior Work

Cirak *et. al.* [17] have proposed boundary conditions for thin shell finite element problems with triangular control meshes which incorporate the implied ghost faces into the analysis. Figure 5.4 shows two typical faces of a control mesh meeting along a boundary edge. Four control mesh vertices are shown with their corresponding displacements are labeled $\mathbf{u}_1 \ldots \mathbf{u}_4$. The region colored gray indicates the interior of the model, and the white region indicates ghost faces. The ghost faces must be constructed with the rules given by Schweitzer and Duchamp [53], which requires that all faces along the boundary possess regular valence (number of neighbors).

The boundary conditions proposed in [17] are summarized in Table 5.1. These relations are specifically formulated to maintain a positioning of the ghost faces *after* deformation that continue to meet the restrictions posed by Schweitzer and Duchamp [53]. It is noteworthy that a constraint must be satisfied even if both rotation and translation are free.

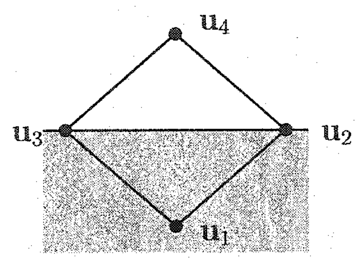


Figure 5.4: Boundary Geometry

Table 5.1: Cirak *et. al.*Boundary Conditions

Displacement	Rotation	Boundary Condition
Free	Free	$u_4 = u_2 + u_3 - u_1$
Fixed	Free	$u_4 = -u_1, \quad u_2 = u_3 = 0$
Fixed	Fixed	$u_1 = u_2 = u_3 = u_4 = 0$

Convergence and Accuracy

The constraints described in Section 5.2.2 are straightforward and simple to implement. They are especially appropriate for applications in graphics where the definition of the ghost faces is compatible with fast rendering methods and general practice. In the field of solid mechanics this constraint framework is not optimal. Cirak *et. al.* [17] demonstrate convergence for a variety of test problems with increasing refinement demonstrating the sufficiency of their method. Although sufficient, these boundary conditions proposed are not necessary and lower the rate of convergence of the subdivision element method as a whole.

The reasons behind the low convergence rate of the constraints in table 5.1 are most easily illustrated in the case of a boundary in which several successive edges are constrained such that both rotations and displacements are fixed. This case is illustrated in Figure 5.5. Referring to Table 5.1 the displacements of all of the vertices are constrained to be zero. The edge between vertices 5 and 6 is highlighted in extra bold in Figure 5.5. Displacement

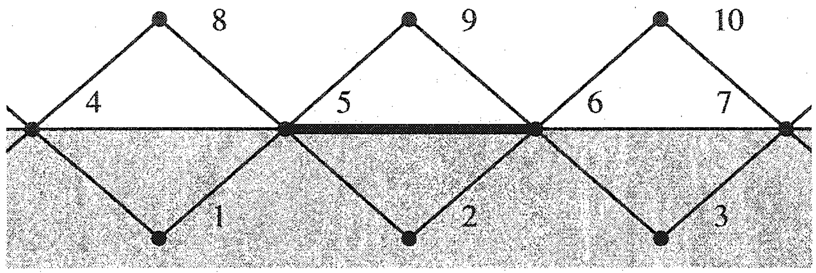


Figure 5.5: Clamped Edge Neighborhood

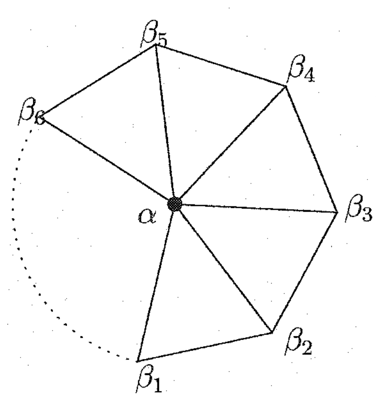
along this edge is fully specified by the 10 vertices shown in the figure. Displacement in this region will be a linear combination of the local subdivision basis functions weighted by their corresponding coefficients, all of which are zero. Therefore the displacement along this edge is identically zero, as are *all* of its derivatives.

The boundary condition applied to the edge in the above example was designed to limit displacement and rotation (slope) only, however it also effectively inhibits deformations involving higher derivatives such as curvature. Traditional definitions of the clamped boundary condition allow for finite curvatures at a boundary while specifying that slopes and displacements remain fixed. Furthermore this implementation also imposes constraints that inhibit in-plane modes of deformation. As the displacement and all of its derivatives are constrained to be zero along an edge, membrane (in-plane) strains are also forced to vanish. This coupling further impedes convergence when both bending and membrane strains are present. Although we have shown this effect only for the specific case of a clamped boundary region, similar reasoning may easily be applied to show that these effects are inherent to all of the boundary conditions described in [17]; in the case of free rotations, the requirement that $\mathbf{u}_1$ and $\mathbf{u}_4$ in Figure 5.4 be equal and opposite imposes itself on bending and membrane deformation alike, effectively restricting membrane strains along the boundary edge to be zero. All of these constraints are sufficient in that they do indeed constrain the modes of deformation they are designed to enforce, but they unnecessarily inhibit other modes of deformation that should be allowed, leading to slower than possible convergence.

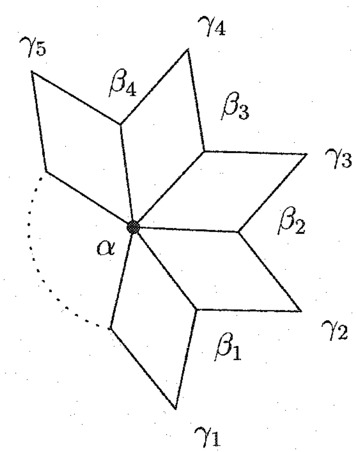
As refinement proceeds, the areas over which the boundary conditions are applied shrinks such that results do converge with increasing refinement, but at a slower rate than the discretized solution of the Kirchhoff-Love thin shell equations of motion on the interior of the body. The boundary conditions used are the limiting factor in the total convergence rate of the solution. In the following sections of this work we describe a new constraint formulation and demonstrate convergence rates of $\mathcal{O}(N^2)$.

5.2.3 *Vertex Position Constraint*

The limit location and displacement of any vertex of the control mesh may be found simply by using a limit mask. Limit masks are a set of values that weight the geometry of a neighborhood of the control point of interest on the coarse mesh which are derived from an eigen-analysis of the local subdivision matrix. Example limit masks are show in Figure 5.6. The values for α, β and γ are dependent on the valence of the central vertex. Two local tangent vectors may also be found by a similar procedure, with different values for α, β and γ. The explicit definition of these coefficients are given in Appendix sections A.3.1 and A.3.2.



(a) Loop Mask



(b) Catmull-Clark Mask

Figure 5.6: Limit Masks

Limit masks may also be used to enforce the limit displacement of a point of interest. The use of limit masks to create constraints at nodal locations was first put forth by Halstead *et. al.*in the context of creating fair interpolating surfaces [35]. We have extended this approach to create boundary conditions appropriate for thin shell finite element analysis using subdivision surfaces. The limit displacement of any control mesh vertex may be written using the limit mask as a linear combination of the displacements of its neighbors. This relation takes the form

$$\mathbf{u}_{limit} = \sum_{i=1}^{k} \beta^i \mathbf{u}_i, \tag{5.1}$$

where the $\mathbf{u}_i$ correspond to displacements of control mesh vertices in the neighborhood of the vertex of interest, and the β^i now generically represent the appropriate value of α^i, β^i or γ^i from the mask and k is the size of the neighborhood. Because the form of Equation 5.1 is linear in displacements $\mathbf{u}_i$, the constraint may be written as

$$\mathbf{C}\mathbf{u} = \mathbf{g}, \tag{5.2}$$

where $\mathbf{C}$ is the vector of vertex weights from the limit mask, $\mathbf{u}$ is the vector of control point displacements, and $\mathbf{g}$ is the value of desired displacement (the zero vector for zero displacement). Three separate constraints will be produced, one for each of the Cartesian axes. In the implementation of finite element solvers it is common to unwind vectors of displacements into matrices using Voight's notation whereupon the size of $\mathbf{C}$ will be 3xk, $\mathbf{u}$ kx1, and $\mathbf{g}$ 3x1.

Most often in finite element simulations boundary conditions are applied at nodal locations, but they need not be in general. If the point of interest does not lie at a nodal location, but instead within the interior of a face, the evaluation technique of Stam [58, 59] may be used to evaluate a set of coefficients which weight the values of all vertices that are in the neighborhood of the given face. The method described by Stam is only defined for faces which possess at most one extraordinary point; however this technique may be extended to faces with any number of extraordinary vertices as shown in Green *et. al.* [32].

5.2.4 *Vertex Direction Constraint*

Limit vertex displacements may also be constrained from moving in a certain direction. Displacement is prohibited in a direction $\mathbf{d}$ by requiring that

$$\mathbf{u}_{limit} \cdot \mathbf{d} = 0. \tag{5.3}$$

By the definition of the limit displacement of a control mesh vertex of Equation 5.1 and linearity of the limit mask,

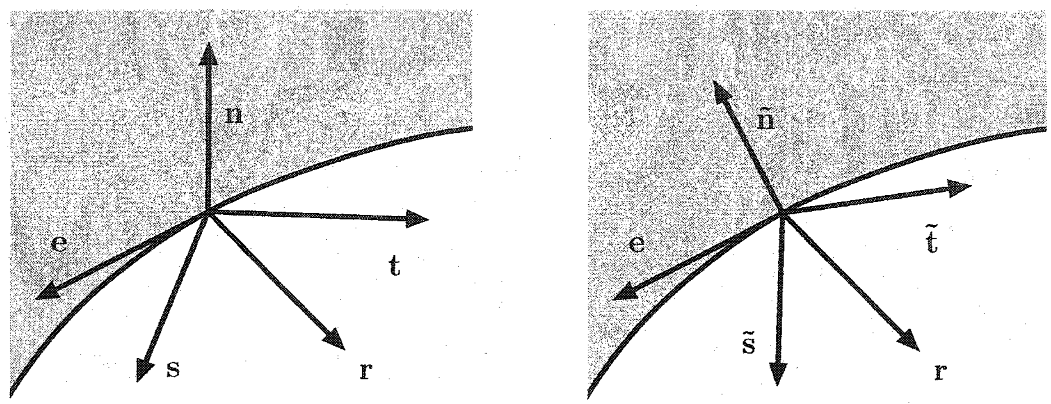
$$\left(\sum_{i=1}^{k} \beta^i \mathbf{u}_i\right) \cdot \mathbf{d} = \sum_{i=1}^{k} \beta^i \left(\mathbf{u}_i \cdot \mathbf{d}\right) = 0. \tag{5.4}$$

Equation 5.4 is a scalar-valued constraint which is linear in displacements $\mathbf{u}_i$ and may be written in the form $\mathbf{Cu} = \mathbf{g}$. Each vertex may be constrained from moving in up to three independent directions. In the case that $\mathbf{d}$ is chosen to be each of the Cartesian basis vectors in turn, the resulting system of constraints will be equivalent to the displacement constraint described in Section 5.2.3.

5.2.5 *Rotation and Clamped Constraint*

Clamped boundary conditions are common to engineering analysis of plates and shells. These boundary conditions prescribe that both the position and rotation of a body along the boundary must be fixed. Rotation constraints are useful in their own right and may also be combined with a displacement constraint, such as the one described in section 5.2.3, to create a clamped boundary condition which inhibits both displacement and rotation.

The rotation at any point on the surface of a body may be specified by constraining changes to the local normal. It is tempting to define rotation constraints as not allowing any changes to the unit normal of the surface after deformation, however this is problematic as discussed at the end of this section. Instead a rotation constraint specifically inhibits changes to the normal in a direction perpendicular to a boundary edge but allows free rotations in the direction of the edge itself.



(a) Undeformed (b) Deformed

Figure 5.7: Clamped Boundaries

Let the local tangent plane to an undeformed surface be spanned by two vectors $\mathbf{s}$ and $\mathbf{t}$, both orthogonal to the normal vector $\mathbf{n}$, as shown in Figure 5.7(a). Another vector $\mathbf{e}$ is shown, which is tangent to the boundary curve of the surface, and lies in the plane spanned by $\mathbf{s}$ and $\mathbf{t}$. If no rotation about the axis $\mathbf{e}$ is allowed, then the deformed normal $\tilde{\mathbf{n}}$ may not move outside the plane spanned by $\mathbf{e}$ and $\mathbf{n}$. We may define for convenience a vector

$$\mathbf{r} = \mathbf{n} \times \mathbf{e} \tag{5.5}$$

that is orthogonal to both the boundary edge and the undeformed normal. The deformed configuration is illustrated in in Figure 5.7(b) where tildes denoted deformed quantities. $\mathbf{r}$ and $\mathbf{e}$ are not recomputed after deformation. Inhibiting rotation of the deformed normal $\tilde{\mathbf{n}}$ about the undeformed edge direction $\mathbf{e}$ may be written conveniently as

$$\mathbf{r} \cdot \tilde{\mathbf{n}} = 0. \tag{5.6}$$

Implementation

The tangent vectors $\mathbf{s}$ and $\mathbf{t}$ may be found at nodal locations by applying limit tangent masks to the deformed configuration of the body. Computing tangent masks at control mesh vertices is highly desirable, as constraints may be specified directly at nodal locations. If the constraint is imposed on the interior of a mesh face the Stam evaluation technique may be used to evaluate a weight vector for each of the surface tangents as a function of the one neighborhood of the current face. In the following discussion w_s^i and w_t^i will be used to represent the weights of the two tangent masks regardless of how they were computed.

The two limit tangent vectors $\mathbf{s}$ and $\mathbf{t}$ are determined by weighting the appropriate neighborhood of nodal values by the weights w_s^i and w_t^i respectively. The corresponding limit tangents in the deformed configuration of the body, $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$ may be found in a similar manner as displacements from the undeformed tangents, as in Equation 5.7

$$\tilde{\mathbf{s}} = \mathbf{s} + \tilde{\mathbf{ds}} \qquad\qquad \tilde{\mathbf{t}} = \mathbf{t} + \tilde{\mathbf{dt}}, \tag{5.7}$$

where

$$\tilde{\mathbf{ds}} = \sum_i w_s^i \mathbf{u}_i \qquad\qquad \tilde{\mathbf{dt}} = \sum_i w_t^i \mathbf{u}_i. \tag{5.8}$$

The deformed and undeformed normals are related to their tangent vectors respectively as

$$\mathbf{n} = \mathbf{s} \times \mathbf{t} \qquad\qquad \tilde{\mathbf{n}} = \tilde{\mathbf{s}} \times \tilde{\mathbf{t}}. \tag{5.9}$$

The vector $\mathbf{r}$ may be computed from known quantities by Equation 5.5. The constraint defined in Equation 5.6 may be written in terms of nodal displacements as follows,

$$\begin{aligned}
0 &= \mathbf{r} \cdot (\tilde{\mathbf{s}} \times \tilde{\mathbf{t}}) \\
&= \mathbf{r} \cdot ((\mathbf{s} + \tilde{\mathbf{ds}}) \times (\mathbf{t} + \tilde{\mathbf{dt}})) \\
&= \mathbf{r} \cdot (\mathbf{s} \times \mathbf{t} + \tilde{\mathbf{ds}} \times \mathbf{t} + \mathbf{s} \times \tilde{\mathbf{dt}} + \tilde{\mathbf{ds}} \times \tilde{\mathbf{dt}}) \\
&= \mathbf{r} \cdot (\tilde{\mathbf{ds}} \times \mathbf{t} + \mathbf{s} \times \tilde{\mathbf{dt}} + \tilde{\mathbf{ds}} \times \tilde{\mathbf{dt}}) \\
&= \mathbf{r} \cdot (\tilde{\mathbf{ds}} \times \mathbf{t} + \mathbf{s} \times \tilde{\mathbf{dt}}) + \mathcal{O}(u_i^2) \\
&= \tilde{\mathbf{ds}} \cdot \mathbf{t} \times \mathbf{r} + \tilde{\mathbf{dt}} \cdot \mathbf{r} \times \mathbf{s} + \mathcal{O}(u_i^2)
\end{aligned} \tag{5.10}$$

The last two lines of the preceding derivation keep only terms which are linear in the displacements $\mathbf{u}$. It is straightforward to add the appropriate terms back in for a full nonlinear implementation. Using Equation 5.8, the final form of the linearized constraint in terms of displacements becomes

$$\sum_i \mathbf{u}_i \cdot (w_s^i(\mathbf{t} \times \mathbf{r}) + w_t^i(\mathbf{r} \times \mathbf{s})) = 0. \tag{5.11}$$

Equation 5.11 is a single scalar equation in terms off all of the degrees of freedom of the neighborhood of the area of interest, which inhibits rotation about a boundary edge. Combining the scalar constraint of Equation 5.11 with the displacement constraint of Section 5.2.3 fully clamps points on the boundary.

As previously noted, it is tempting to write a clamped boundary constraint by simply constraining the deformed normal not to change direction at all. Along clamped boundary edges where both rotation and displacement are inhibited the normal will in fact not change direction. However combining constraints that fix the direction of the deformed normal *and* constrain translations leads to an over-determination of the constraint, as rotation perpendicular to an edge is prohibited by both constraints. Some solvers can deal with over-determined systems and find minimum energy solutions, but it is far better to avoid the potential difficulties associated with over-determined constraints by applying a properly reasoned formulation that lead to non-singular systems of equations.

A constraint on the direction of the normal is straightforward to implement and may be easily derived from the relation that the deformed surface tangent vectors must remain perpendicular to the undeformed surface normal, as expressed below in Equation 5.12.

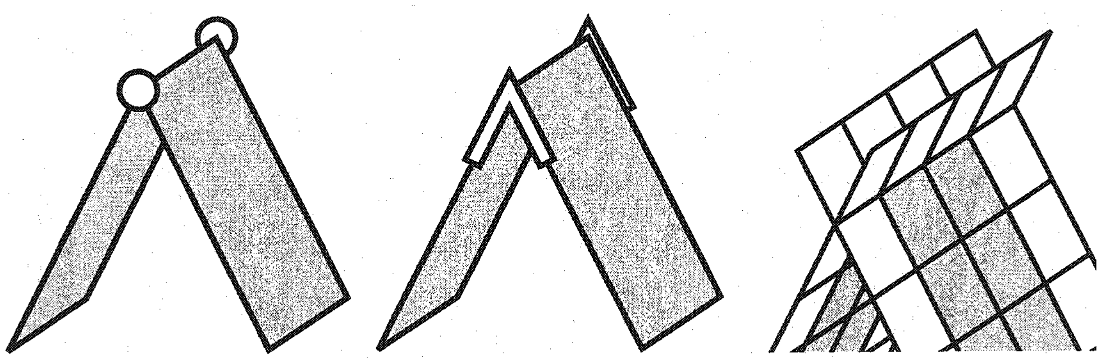
$$\mathbf{n} \cdot \tilde{\mathbf{s}} = 0 \qquad\qquad \mathbf{n} \cdot \tilde{\mathbf{t}} = 0 \tag{5.12}$$

A constraint of this form was first described by Halstead *et. al.* [35] and is useful in its own right, for instance in ensuring tangency between two mating surfaces. However it should not be confused with a rotation constraint which isa different concept.

There is also some freedom in choosing a definition for the edge tangent $\mathbf{e}$, especially at corners. At corners e may be chosen to be the average of two tangents with generally favorable results.

5.2.6 *Internal Pinned and Clamped Connections*

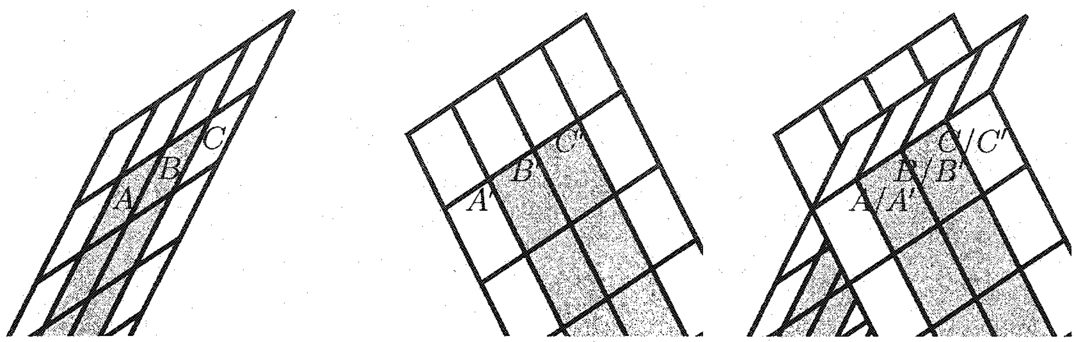
Internal connections are frequently encountered in structural mechanics problems and may be effectively handled within our boundary condition formulation. Figures 5.8(a) and 5.8(b) illustrate the joining of two flat shells, connected by pinned and clamped connectors respectively, along their top edges. A pinned constraint along a boundary edge requires that for each point along the boundary the displacement of mating points from each side be equal. A clamped constraint requires that the rotations be equal as well as displacements. Internal boundaries may be modeled by mating two halves of a shell body together using constraints. Figure 5.8(c) shows two quadrilateral subdivision control meshes, surrounded by ghost faces, meeting along an internal boundary.



(a) Pinned (b) Clamped (c) Subdivision Model

Figure 5.8: Models with Internal Boundaries

The process of creating an internal boundary is illustrated in Figure 5.9. The left portion of the roof example is shown in Figure 5.9(a) with vertices A, B and C along the top boundary. Figure 5.9(b) shows the right portion of the roof with corresponding vertices A', B' and C' along the top boundary. These two halves are shown mated together in



(a) 'Left" Control Mesh (b) 'Right" Control Mesh (c) Mated Control Mesh

Figure 5.9: Application of Rigid Internal Edge Constraints

Figure 5.9(c). Internal pinned connections are formulated similar to the simple supports described in Section 5.2.3. Limit masks are used to enforce compatibility of displacements between two mating vertices.

$$\begin{gathered} \mathbf{u}_{limit} = \mathbf{u}'_{limit} \\ \therefore \\ \sum_{i=1}^{k} \beta^i \mathbf{u}_i = \sum_{j=1}^{k} \beta^j \mathbf{u}'_j \end{gathered} \tag{5.13}$$

where displacements $\mathbf{u}_i$ and $\mathbf{u}'_j$ are the generalized displacements of each mated half respectively. The constraint is linear in the displacements $\mathbf{u}_i$ and $\mathbf{u}'_j$. Internal clamped constraints may also be formulated in a similar fashion to those described in Section 5.2.5. Instead of setting rotations equal to zero, the rotations at two mating nodes are set equal to each other. It is assumed to be the responsibility of the designer of the control mesh to create internal boundaries which make sense for the application of constraints. The boundaries of each mated half must coincide to a numerical tolerance consistent with simulation accuracy.

5.3 Verification

In this section we demonstrate the second order accuracy of our constraint formulation by applying it to a variety of example problems and measuring the resulting error with respect to known solution values. Belytschko *et. al.* [9] have assembled an obstacle course of problems with known elasticity solutions for comparison. These problems are designed to provide complex stress states that provide stringent tests of solution strategy.

The order of convergence of a solution technique is determined by measuring the increase in accuracy of a solution with respect to increasing refinement. Our chosen error metric is the relative error in displacement of a chosen point on the model with respect to a known value for comparison with the displacement results presented in [9]. The solution convergence may be conveniently measured by examining the slope of a plot of error versus number of degrees of freedom on a plot with logarithmic axes. A solution scheme which converges at second order will decrease error by a factor of four with every doubling in the number of degrees of freedom used to represent the system. We also compare the convergence properties of our proposed boundary condition formulation to those previously demonstrated in the literature for select problems and demonstrate improved convergence.

5.3.1 Flat Plate

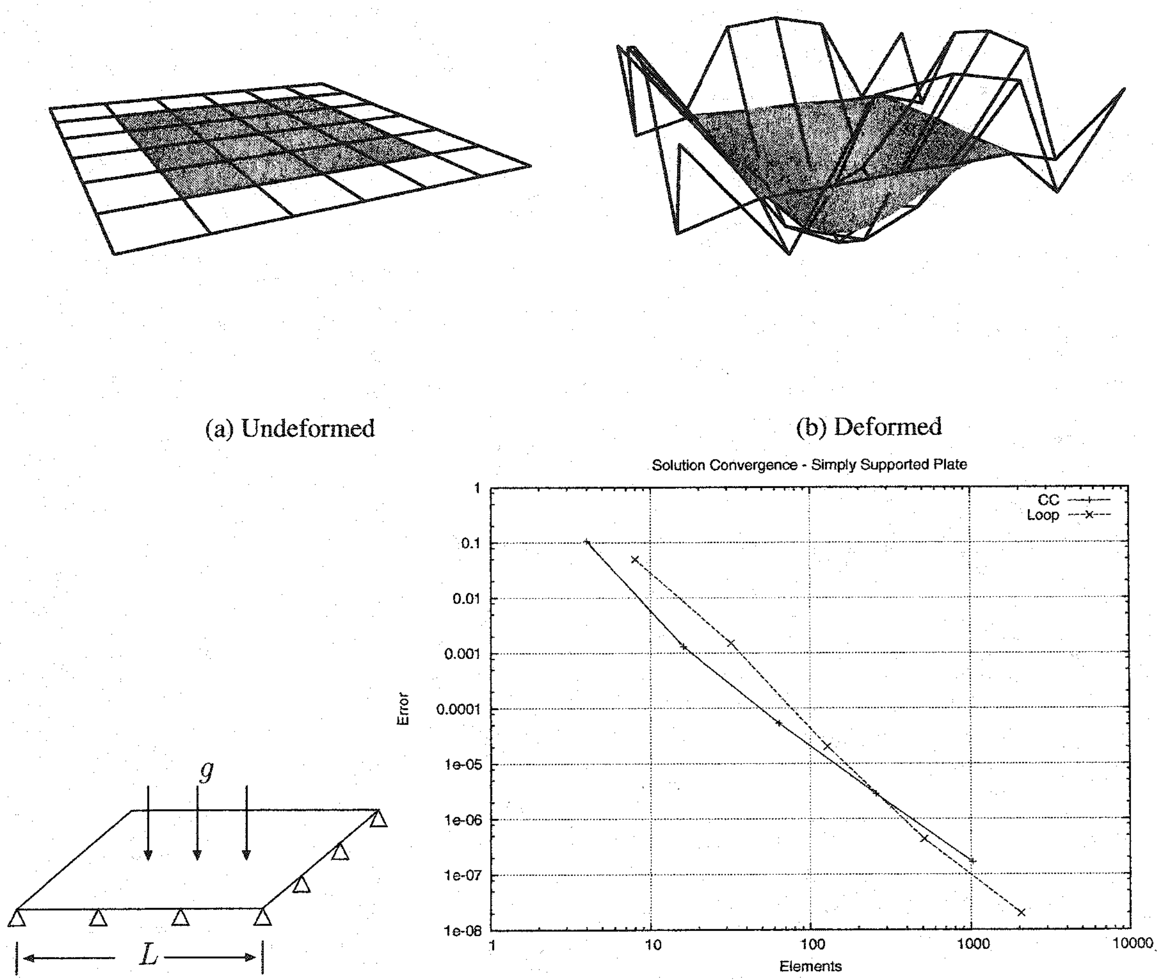
As a first example we consider the case of a square plate under uniform gravitational loading conditions. Although geometrically simple, a flat plate model is a useful example of a thin shell boundary value problem in which bending energies dominate the solution. Two different boundary conditions are examined, simple supports which constrain the outside edges of the plate to have zero displacement, and clamped supports which also enforce that the outside edges have zero slope. In each case, the plate is subjected to a uniform gravitational load. Error is defined to be the normalized difference between the computed displacement of the center of the plate, and the analytical solution. Due to the symmetric geometry, loading, and boundary conditions of the model; the material properties, size of the plate and loading, represent only a scaling factor and do not affect the overall shape of the solution.

Simply Supported Boundaries

A schematic for the simply supported square plate is shown in Figure 5.10(c). The initial and undeformed configurations of the model (with deformations greatly exaggerated) are shown in Figures 5.10(a) and 5.10(b) respectively. In each of these figures the edges of the control mesh are outlined and surround the shaded limit surface. It is interesting to note that parts of the control mesh, especially the ghost faces undergo large displacements while the boundaries of the limit surface remain still, as enforced by the simple support.

Figure 5.10(d) shows convergence in displacement error for a simply supported plate using the boundary conditions discussed in section 5.2.3. The vertical axis represents the error measured for the deflection of the center of the plate. The exact elasticity solution was calculated via a trigonometric series solution to a high degree of accuracy [62]. The horizontal axis shows the number of of elements used to represent the model for simulation. Two geometric models of the thin plate were created using Loop and Catmull-Clark subdivision respectively and convergence results for each of these two models are plotted in Figure 5.10(d). Both the Loop and Catmull-Clark models exhibit quadratic accuracy as indicated by a slope slightly steeper than -2 with respect to refinement, demonstrating the $\mathcal{O}(N^2)$ accuracy of our approach for this problem.

The character of the results for the Loop and Catmull-Clark representations are quite similar. In the limit of increasing refinement a model using Loop subdivision will have 2 elements per amortized degree of freedom whereas the Catmull-Clark scheme has only one.



(a) Undeformed (b) Deformed

(c) Schematic (d) Convergence

Figure 5.10: Plate with Simply Supported Boundaries

Clamped Supports

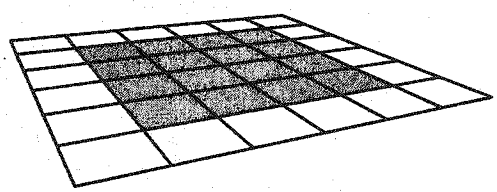
The clamped square plate model is similar to the simply supported model with the exception of the boundary conditions. Clamped plates are stiffer than those with only simple supports, and the loading of this simulation has been increased by approximately a factor of 3 to put the overall displacements on par with those of the simply supported plate. The resulting limit surface in the deformed configuration differs in character from its simply-supported counterpart in that the boundary edges are required to have zero slope. This example allows us to effectively demonstrate the improved accuracy of our formulation compared to prior approaches.

Accuracy results for five levels of discretization are shown in Figure 5.11(d). Four trends are shown simultaneously. The top two lines, labeled "Implied Boundary Loop" and "Implied Boundary CC" represent the convergence of equivalent Loop and Catmull-Clark subdivision models using the boundary conditions described in [17] for Loop subdivision. Although reference [17] did not discuss Catmull-Clark subdivision, for purposes of comparison, we have implemented a Catmull-Clark boundary condition in the same spirit as was done for the Loop subdivision elements. The slope each of these lines is much shallower than -1 indicating that the solution converges more slowly than $\mathcal{O}(N)$ accuracy in that formulation.

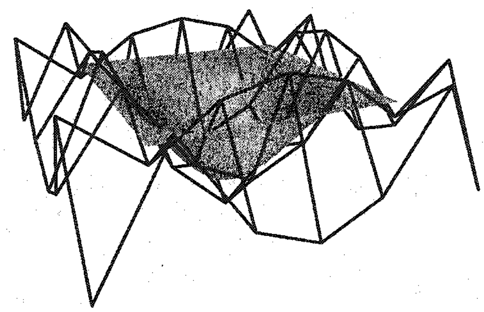
The bottom two data sets of Figure 5.11(d), labeled "Loop" and "CC" respectively, represent the convergence given by our newly proposed boundary conditions discussed in section 5.2.5. Both of these plots converge in displacement error with a slope of -2, indicating that this boundary condition formulation converges to the true solution with order $\mathcal{O}(N^2)$ with respect to the number of degrees of freedom.

5.3.2 *Hemispherical Shell*

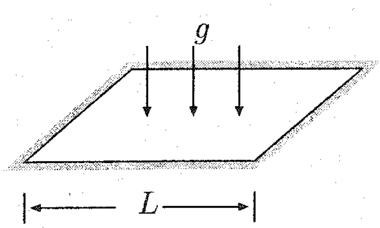
Belytschko *et. al.*describe the hemispherical shell model as a "challenging test of an element's ability to represent inextensional modes." The problem consists of a hemisphere



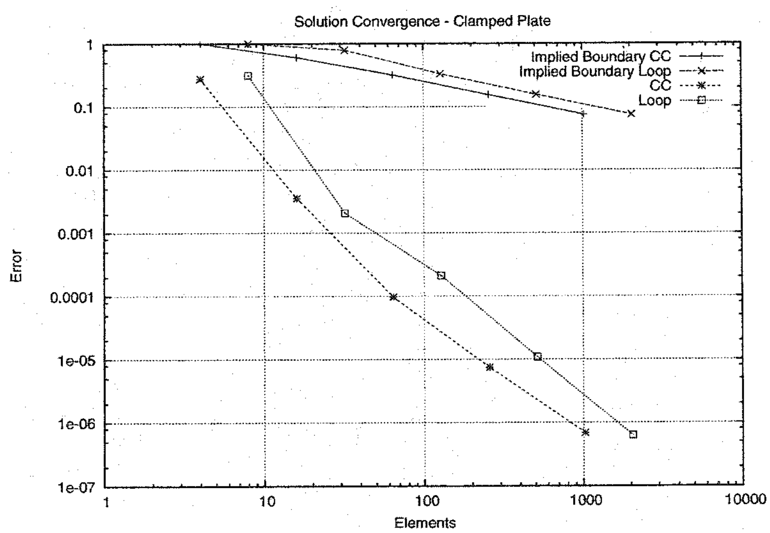
(a) Undeformed



(b) Deformed



(c) Schematic



(d) Convergence

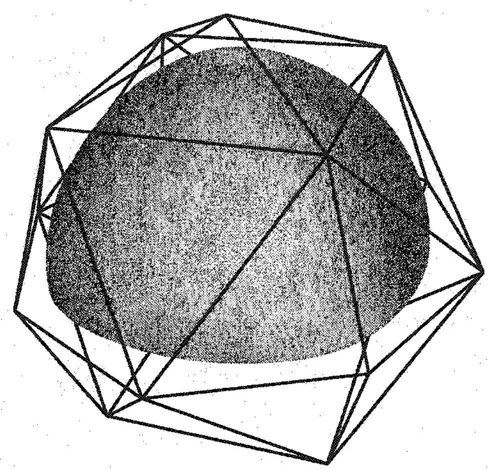
Figure 5.11: Plate with Clamped Boundaries

with four alternating point loads applied around the equator. This model is illustrated in Figure 5.12(c). The shaded limit surface surrounded by its coarse control mesh in the initial and deformed configurations of the model are shown in Figures 5.12(a) and 5.12(b) respectively. Boundary conditions are applied simply to remove rigid body modes from the system. A position constraint and a tangent plane constraint are applied to the pole of the model, constraining 5 of the 6 rigid body modes. Lastly "twist" is constrained by fixing one degree of freedom of one point around the equator. Solution convergence is shown in Figure 5.12(d). Error for this problem is defined as the normalized difference between calculated displacement at a point of load application along the boundary and an analytical displacement given in [9].

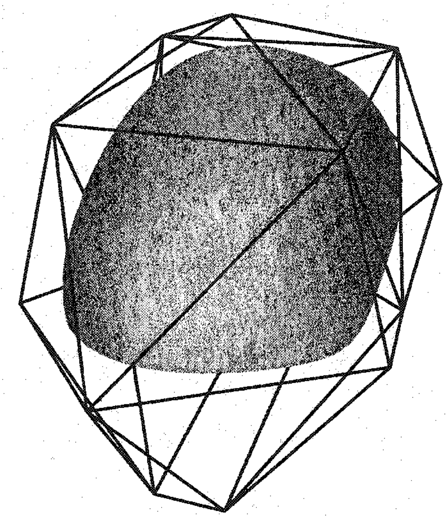
Convergence is illustrated in Figure 5.12(d). Error decreases at a rate of $\mathcal{O}(N^2)$ with respect to the number of degrees of freedom, just as the solution error in the flat plate problems does. This result adds insight to the relative convergence of the subdivision element approach and our formulation for boundary conditions. The constraints applied to the hemispherical shell are designed to constrain rigid body degrees of freedom only and do not introduce any reaction forces into the system (due to the symmetry of the loading). The rate of convergence of the system should therefore be governed by the convergence properties of the element alone and not the boundary conditions. The convergence properties of the Loop subdivision basis functions have been proven rigorously by Arden [3].

The result that rates of convergence of displacement error are the same for problems with constraints that contribute energy to the system and those that do not, shows that the discretization error contributed to the system by the constraints described in sections 5.2.3 and 5.2.5 converges at a rate at least great as that of the element itself. Thus our proposed formulation for boundary conditions does not introduce discretization error which impedes the accuracy of solution any more than the subdivision shell element itself does. The boundary conditions described in [17] do not possess this property, and in fact limit the rate of convergence of the entire system, as shown in Figure 5.11(d).

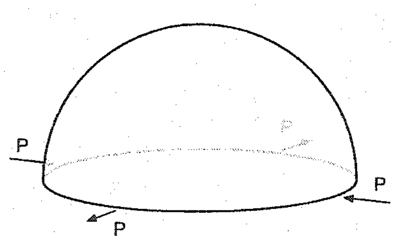
Unlike the flat plate models, a subdivision hierarchy cannot exactly model spherical and



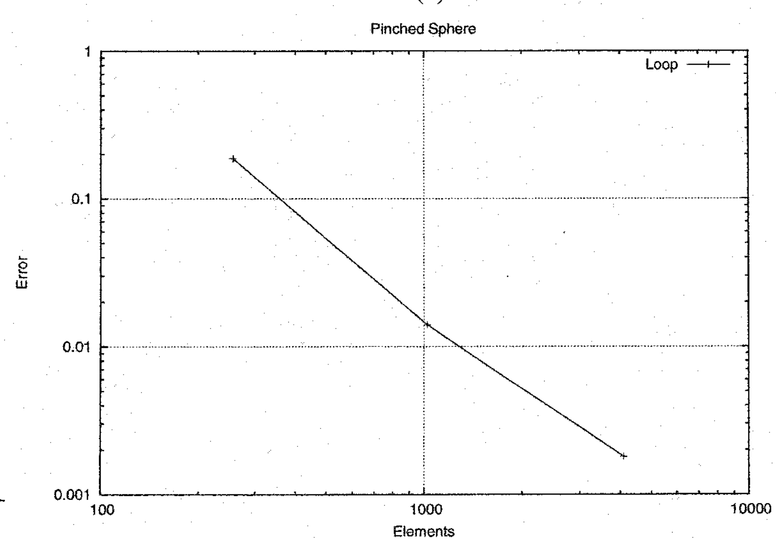
(a) Undeformed



(b) Deformed



(c) Schematic



(d) Convergence

Figure 5.12: Hemispherical Shell

cylindrical shapes, due to the limitations inherent in the underlying cubic basis functions. The hemisphere was modeled by starting with an octahedral control mesh. After each level of subdivision, control mesh vertices were moved via a simple optimization process to best represent a hemispherical shell. Because the connectivity of the model is retained and the perturbations are small, the multi-level acceleration scheme discussed in section 6.6 remains an efficient solution approach for this model.

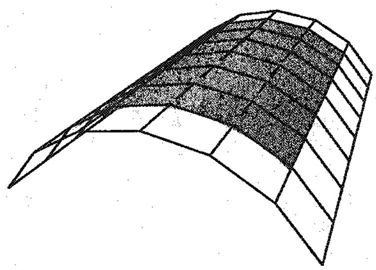
5.3.3 Scordelis-Lo Roof

The Scordelis-Lo roof problem provides a rigorous test of an element's ability to represent inextensional bending and complex states of membrane strains. The rest and deformed states of the model are illustrated in Figure 5.13 (with deformations greatly exaggerated for purposes of illustration).

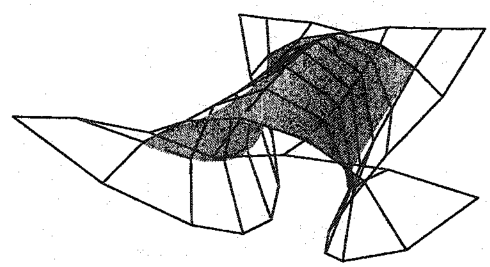
The geometric and material properties relevant to this problem are given in Appendix section A.4.1. Convergence results for this test are presented in Figure 5.13(d). Relative error in vertical displacement at the mid-side of the free edge is shown to converge with second order accuracy with increasing refinement. For this problem, we measure convergence with respect to a "true" displacement value measured on a very finely discretized version of our mesh. For the material properties listed in the Appendix we measure a maximal mid-side displacement of 0.3005, about 0.6% lower than the value given in [9]. A more rigorous study of the Scordelis-Lo roof problem is presented in [5] which notes a 3% variation in elasticity results for different formulations of this problem. As with the hemisphere problem of section 5.3.2 the vertices of the body were slightly repositioned to best represent a cylinder at the given level of discretization.

5.3.4 Pinched Cylinder

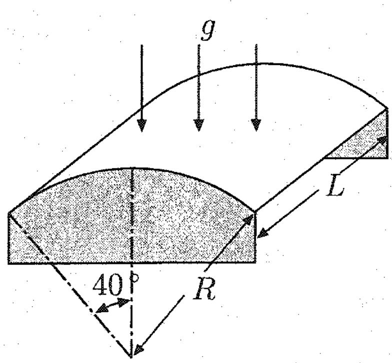
Figure 5.14(c) illustrates the benchmark problem of a cylindrical shell supported by two rigid diaphragms under load from two opposing concentrated forces. This problem tests



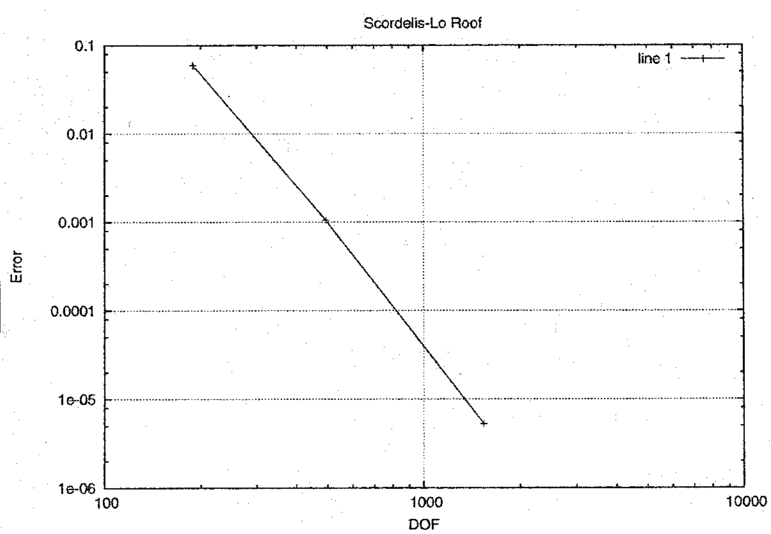
(a) Undeformed



(b) Deformed



(c) Schematic



(d) Convergence

Figure 5.13: Scordelis-Lo Roof

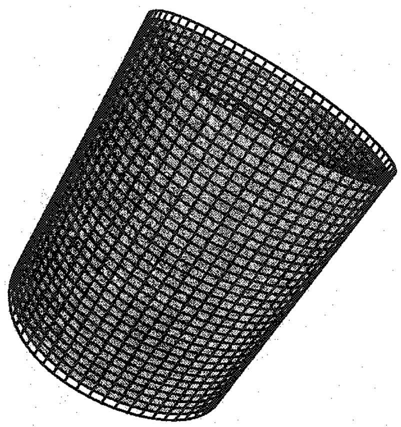
the ability of a simulation to represent complex bending and membrane stress states. Convergence is illustrated in Figure 5.14(d), which demonstrates clear second order convergence for our solution framework. The number of degress of freedom listed are for the full cylinder model.

5.3.5 Integration

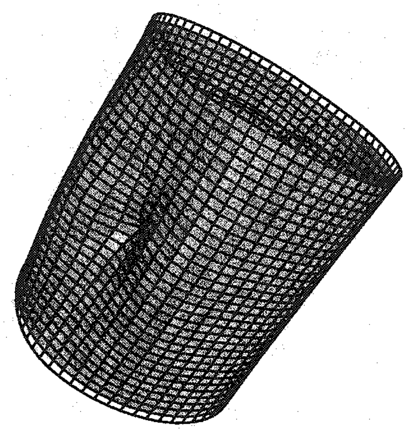
Integration of the stiffness matrix for all models was performed using traditional Gauss-Legendre Quadrature methods [65]. Given the cubic nature of the underlying spline basis functions for the Catmull-Clark scheme, a 3x3 quadrature scheme was chosen and proved sufficient in practice. For triangular meshes (quartic basis functions) a 7 point scheme was used. Stam's technique was used to evaluate each of the basis functions at the Gauss points.

5.4 Summary

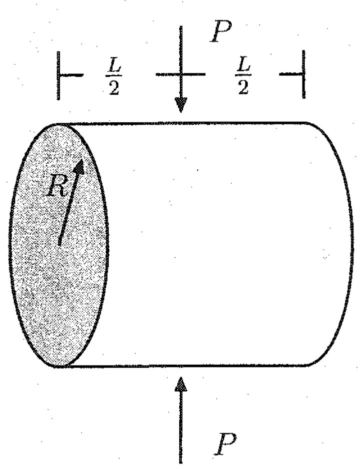
In this chapter we presented a new framework for applying constraints to subdivision surface element simulations of thin shells. We formulated constraints for representing commonly encountered boundary conditions in thin shell engineering design: simple and clamped edge supports. We demonstrated second order accuracy in the displacement with increasing refinement for a rigorous obstacle course of test problems. The accuracy of our constraint framework has been shown to be consistent with the accuracy of subdivision elements for thin shell simulation. Our proposed framework for boundary conditions does not introduce discretization error that impedes the accuracy of solution. Our proposed framework does not impose any requirements on the topology of the underlying subdivision control mesh. Furthermore, we adapted the multigrid-preconditioned solver described in Chapter 6 and Green *et. al.* [32] to provide an efficient framework for solving large-scale problems with constraints.



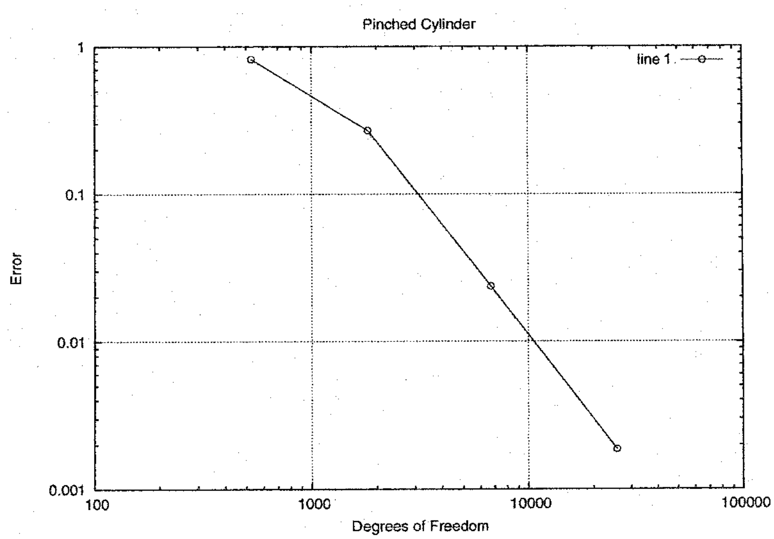
(a) Undeformed



(b) Deformed



(c) Schematic



(d) Convergence

Figure 5.14: Pinched Cylinder Roof

Chapter 6

MULTILEVEL SIMULATION AND EFFICIENT SOLUTION STRATEGIES

The subdivision surface thin-shell elements of Chapter 4 have been shown to be an appropriate and successful approach for simulating the mechanical deformation of thin shells within a finite element framework. By construction, the subdivision shape functions provide the necessary C^1 continuity requirements (and H^2 integrability) for representing the solution of the fourth-order equilibrium equations governing the behavior of thin shells. These subdivision model descriptions also provide the rather elegant property of representing both the geometry and the physics of the deformation with the same mathematical description.

However, when coupled with standard solvers such simulations do not scale well computationally. Given the fourth order nature of the governing equations, the condition number of the underlying stiffness matrices involved grows as $O(\frac{1}{h^4})$, or $O(N^2)$, where h is a nominal element length scale and n is the number of degrees of freedom in the discretized model. Using common preconditioned conjugate gradient solvers, the number of iterations grows, at best, linearly with problem size $O(N)$, and run time grows as $O(N^2)$ even with the most efficient sparse representations. In the context of high-resolution simulations involving large meshes with $n \gtrsim 10^5$ elements, the run time costs become prohibitive and present serious practical limits to the effective use of simulation in engineering analysis.

In this chapter, an algorithm that exploits the hierarchical, multilevel structure of subdivision surfaces to accelerate the convergence of solution strategies for thin shell simulations is described. The use of the standard subdivision matrices allows the inter-level propagation of solution values, and leads to methods where the number of iterations is nearly independent of problem size, resulting in run time that grows only linearly. Moreover the strategies developed are not only useful on large scale meshes but are shown to be also advantageous even for relatively small problem sizes. The subdivision framework is used not only for representing the geometry of the solid and the mechanics of the simulation, but also for preconditioning the numerical solution. The framework allows practical simulations that are effective on a broad range of problems and problem sizes.

6.1 Multilevel Schemes

The accuracy of a mechanical simulation is a function of the quality of the discretization of the system. Simulations using coarse discretizations with just a few degrees of freedom can be solved quickly, but may yield poor results. Fine discretizations are often more accurate but there is an associated increase in computational cost. At the heart of many solution algorithms are linear approximations that result in a number of simultaneous equations proportional to the number of unknowns, N. Inverting a dense set of equations requires $O(N^3)$ work. If the system matrix is sparse, this can be reduced to $O(N^2)$. For large problems even $O(N^2)$ algorithms simply takes too long. Ideally one would like to find a solution algorithm that is $O(N)$ in the number of degrees of freedom. Multiresolution solvers can mitigate the cost associated with increased accuracy by considering a hierarchy of refinements of the model instead of only a single resolution [63]. A well designed multiresolution algorithm may achieve solution times of $O(N)$.

Multiresolution solvers accelerate the convergence of numerical simulations by propagating solution knowledge between coarsely discretized versions of a model where solutions may be obtained quickly and finely discretized representations where solutions may be obtained more accurately. The two (or as we will see later, many) representations complement each other in ways that can be exploited to achieve a reduction in total computational effort. This process requires a hierarchical representation of multiple model resolutions with an ability to transfer information between neighboring levels (both coarse-to-fine and fine-to-coarse) as illustrated in Figure 6.1. This geometry shown in this image is deliberately simplified for purposes of illustration; real geometries need not be flat, nor regular.

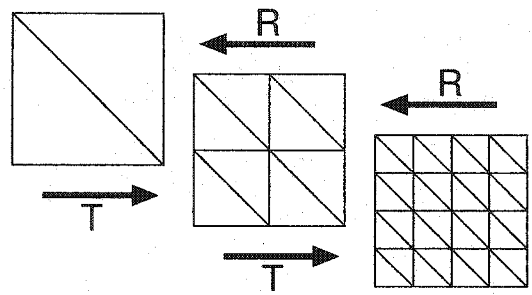


Figure 6.1: An Example Multiresolution Hierarchy

6.1.1 Multigrid

Multigrid (MG) methods are a family of multiresolution methods that leverage the *smoothing principle*, which applies to many physical systems [63]. Many classical iterative methods, such as Gauss-Seidel, have a smoothing effect on the error of discrete elliptical problems. The rate at which these iterative methods reduce error is related to the spatial frequency of the error components. Error components with a high spatial frequency are eliminated quickly, while those with a low spatial frequency reduce more slowly. This effect can be shown by the use of Fourier methods which relate the rate of error reduction to the eigenvalues of the system.

Multigrid methods leverage this smoothing property by noting that the troublesome smooth error components on a detailed model may be well represented on a coarser model of the problem. Multigrid solvers possess two main ingredients: *smoothing* and *coarse grid correction*. The idea is to use an iterative scheme to remove high frequency error components of the solution on a finely discretized version of the problem by smoothing. Low frequency components of error may be eliminated more efficiently on a more coarsely discretized model where computation is less expensive. This process may be implemented recursively over many levels. Subdivision-based finite element meshes form a natural hierarchy that can be used for successive coarse grid corrections.

6.1.2 Multigrid Algorithm

Multigrid makes use of two operations for inter-grid transfer of information. *Prolongation* (T) and *restriction* (R) transfer from coarse to fine and fine to coarse levels respectively. These operators are discussed in detail in sections 6.1.3 and 6.1.4. Pseudo code for the multigrid algorithm is shown below in Algorithm 6.2, adapted from [14] for solving the system $Ku = f$. For a system with an exact solution $\hat{u}$ and an current approximate solution u, the *residual* is defined to be $r = f - Ku$, the *error* is $e = \hat{u} - u$, and they are related as $r = Ke$. Were the error known, we could compute an exact solution, but solving for the error explicitly is equivalent to solving our original problem.

The algorithm proceeds by using the smoothing step to eliminate high frequency error from u. Next the residual r is calculated and then transferred to the next coarser level as $\tilde{r} = R_l r$. The error e, on the current level, is estimated by first solving for the error $\tilde{e}$ on the next coarser level. The coarser error estimate is then transferred to the current level and used to update the solution u. The error on the next coarser level is solved for using a recursive call to the multigrid algorithm, except on the coarsest level (denoted by $l = 0$) where it is solved for directly.

The process of repeatedly refining the error is known as a *W-Cycle*. Each call to the multigrid solver that fully visits each level is known as a *V-Cycle*. Each smoothing step may also require several internal iterations. The number of V and W-Cycles used, and the number of internal smoothing iterations, have a great effect on the performance of the multigrid algorithm.

procedure: multigrid(u, f, l)

for each V-Cycle **do**

  if $l = 0$ **then**

    $u \Leftarrow K_0^{-1} f$

  else

    for each W-Cycle **do**

      $u \Leftarrow \text{presmooth}(K_l, u, f)$

      $r \Leftarrow f - K_l u$

      $\tilde{r} \Leftarrow R_l r$

      $\tilde{e} \Leftarrow \text{multigrid}(\tilde{r}, \mathbf{0}, l-1)$

      $e = T_l \tilde{e}$

      $u \Leftarrow u + e$

      $u \Leftarrow \text{postsmooth}(K_l, u, b)$

    end for

  end if

end for

Figure 6.2: General Recursive Multigrid Algorithm

6.1.3 Prolongation

Information defined on coarse grids must be *prolongated* to finer levels via interpolation. Similarly, values defined on fine grids must be *restricted* to coarser levels, which results in a loss of information.

The choice of prolongation method is part of the design of any multigrid algorithm. We have chosen our prolongation operator between levels j and $j+1$ to be the *global subdivision matrix* $\mathrm{S}_{\mathrm{j}}^{\mathrm{j+1}}$ itself. This choice retains the intuitive feature that a prolongated limit displacement field corresponds exactly to the original limit displacement field of the coarser mesh. Thus, we have now incorporated the subdivision basis in three ways into our solution scheme: it describes the geometry, the displacement, and the connection between levels in our multiresolution hierarchy. The choice of prolongation operator governs the choice of restriction operator via work and energy considerations as discussed below.

6.1.4 Restriction

One way to derive the restriction operator is to require that the work done by the applied forces be equivalent for each level of the hierarchy. This energy requirement results in restriction operator being the transpose of the prolongation operator. It is well known that this transpose relation holds for interpolating basis functions generally used in finite element analysis [48, 21]. We show here that this relation holds true for the choice of the subdivision operator for prolongation, which is not interpolating in general. The work done by the generalized forces F_i moving through generalized displacements U_i is given by

$$W = \mathbf{U}^T\mathbf{F}. \tag{6.1}$$

We must emphasize the use of the term *generalized* displacements. Unlike interpolating bases which define the nodal displacements to be the actual displacement of a point on the body at the nodal location, Loop subdivision basis functions are non-interpolating and the nodal displacements do not possess such a simple physical interpretation. Displacements on a finer level of the mesh are given by

$$\mathbf{U}^{j+1} = \mathbf{S}_j^{j+1}\mathbf{F}^{j+1}. \tag{6.2}$$

Forces are transferred between levels by the restriction operator $\mathbf{R_j^{j+1}}$. Requiring that the work be equal on two levels for arbitrary $\mathbf{U}$ using the subdivision operator for restriction yields:

$$\begin{aligned}
\left(\mathbf{U}^{j+1}\right)^T \mathbf{F}^{j+1} &= \left(\mathbf{U}^j\right)^T \mathbf{F}^j \quad \forall \mathbf{U}^j \\
\left(\mathbf{S}_j^{j+1}\mathbf{U}^j\right)^T \mathbf{F}^{j+1} &= \left(\mathbf{U}^j\right)^T \mathbf{F}^j \quad \forall \mathbf{U}^j \\
\left(\mathbf{U}^j\right)^T \left(\mathbf{S}_j^{j+1}\right)^T \mathbf{F}^{j+1} &= \left(\mathbf{U}^j\right)^T \mathbf{F}^j \quad \forall \mathbf{U}^j \\
&\therefore \\
\left(\mathbf{S}_j^{j+1}\right)^T \mathbf{F}^{j+1} &= \mathbf{F}^j.
\end{aligned}$$

This demonstrates that forces are transferred from fine to coarse levels using the transpose of the subdivision operator.

The coefficient matrix for the coarse level approximation may also be derived by requiring that the strain energies be equal for various levels in the hierarchy. This results in a coefficient matrix that is expressed as $\mathbf{A}_j = \mathbf{S^T K_{j+1} S}$. However implementation of this relation would require a series of expensive matrix multiplies. It is shown in [21] that using the stiffness matrix described by the coarse mesh elements works well and is easier to compute ($\mathbf{A}_j \approx \mathbf{K}_j$) and we follow this approach here.

6.2 Conjugate Gradient

The conjugate gradient algorithm (CG) is an effective solution strategy for positive definite systems of the form $\mathbf{Ax} = \mathbf{b}$. Reviews of the conjugate gradient method may be found in [7, 54]. The conjugate gradient algorithm is optimized for solving sparse systems as it uses only matrix-vector multiplies for each iteration, and no factorization is performed. Because of the lack of factorization, no additional storage space is required. CG uses

procedure: PCG Preconditioned Conjugate Gradient

$\mathrm{r}_0 \Leftarrow \mathrm{b} - \textbf{Multiply}(\mathrm{A}, \mathrm{x}_0)$

for i in $1, 2, \ldots$ **do**

$\quad \mathbf{z}_{i-1} \Leftarrow$ **Precondition(** M^{-1}, r_{i-1} **)**

$\quad \rho_{i-1} \Leftarrow \mathbf{r}_{i-1}^T \mathbf{z}_{i-1}$

$\quad$ **if** $i = 1$ **then**

$\quad\quad \mathbf{p}_1 = \mathbf{z}_0$

$\quad$ **else**

$\quad\quad \beta_{i-1} \Leftarrow \rho_{i-1} / \rho_{i-2}$

$\quad\quad \mathbf{p}_i \Leftarrow \mathbf{z}_{i-1} + \beta_{i-1} \mathbf{p}_{i-1}$

$\quad$ **end if**

$\quad \mathbf{q}_i \Leftarrow \textbf{Multiply}(\mathrm{A}, \mathrm{p}_i)$

$\quad \alpha_i \Leftarrow \rho_{i-1} / (\mathbf{p}_i^T \mathbf{q}_i)$

$\quad \mathbf{x}_i \Leftarrow \mathbf{x}_{i-1} + \alpha_i \mathbf{p}_i$

$\quad \mathbf{r}_i \Leftarrow \mathbf{r}_{i-1} - \alpha_i \mathbf{q}_i$

end for

Figure 6.3: Preconditioned Conjugate Gradient Algorithm

only $O(N)$ additional memory, as opposed to factor based methods which require $O(N^2)$ storage. A preconditioned conjugate gradient algorithm is provided in Figure 6.3 (adapted from [7]).

The implementation presented here makes use of a preconditioner. Without a preconditioner, the number of CG iterations required to solve a problem to a specified accuracy is proportional to the condition number of the matrix A. The condition number of the stiffness matrix grows unfavorably with the number of unknowns as shown in Figure 6.4, which plots the condition number of the stiffness matrix for a clamped flat plate versus the number of degrees of freedom used. CG benefits greatly from the use of a preconditioner; an extra step which lowers the condition number of the system to a more manageable quantity.

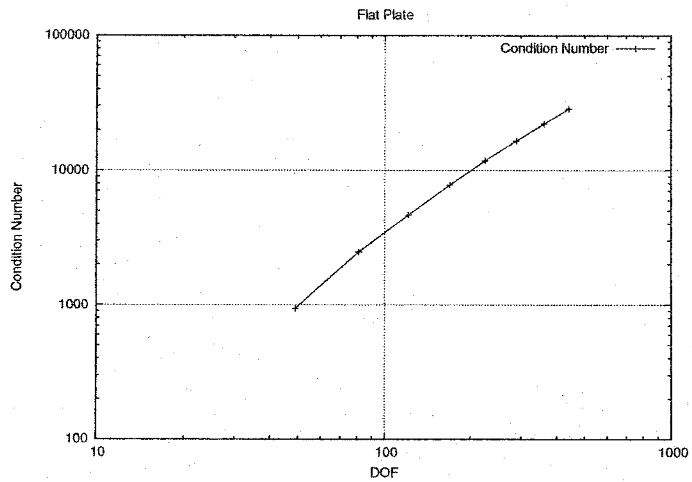


Figure 6.4: Condition Number vs. Refinement

The effectiveness of a preconditioner $\mathbf{M}$ is determined by the condition number of $\mathbf{M}^{-1}\mathbf{A}$. The preconditioner $\mathbf{M}$ must approximate $\mathbf{A}$ sufficiently to improve convergence enough to make up for the cost of computing the product $\mathbf{M}^{-1}\mathbf{r}_i$ once per iteration. One especially effective preconditioner is a multigrid solver. The next section describes the use of a multigrid solver to precondition conjugate gradient iterations.

6.3 Multigrid-Preconditioned Conjugate Gradient

The multigrid method provides a useful framework for solving problems hierarchically, yet achieving optimal performance can be difficult. The algorithm relies on a handful of tuning parameters, including the number of V and W cycles to perform and the number and kind of smoothing steps to apply per iteration. Proper tuning of these parameters remains difficult. Optimal performance of the multigrid algorithm by itself can be difficult to realize in practice.

Fortunately, multigrid concepts can be easily integrated into other strategies for solving linear systems. Multigrid has been shown to be significantly more effective when used to precondition the conjugate gradient algorithm (CG). The conjugate gradient algorithm is specifically designed to takes advantage of the positive definiteness of the systems encountered in finite element simulations. Multigrid preconditioned conjugate gradient strategies (MGPCG) have been shown effective in structural mechanics and other applications [21, 45, 4].

A preconditioner functions by transforming the system $\mathbf{Ku} = \mathbf{f}$ into $\mathbf{MKu} = \mathbf{Mf}$. If the preconditioner $\mathbf{M}$ approximates $\mathbf{K}^{-1}$ well, the new system will have a lower condition number and will thus be easier to solve numerically. Efficient algorithms need not explicitly form the preconditioner $\mathbf{M}$. Our MGPCG implementation uses a forward Gauss-Seidel pre-smoothing step and a backwards Gauss-Seidel post-smoothing step, which retains the symmetry property necessary for preconditioning CG. A single V-Cycle and one iteration of pre and post smoothing per MGPCG iteration is used for our implementation.

6.4 Results

We considered a variety of examples to verify and test the performance of our method, including a distributor cap and fan housing (courtesy Hugues Hoppe), a square flat plate, a hollow coiled tube, and a model with non-trivial genus inspired by the ancient Egyptian "Ankh" symbol. Each of these models and their simulations are shown in figures 6.5 and 6.6 (the fan housing is shown separately in Figure 6.10). The left columns of Figure 6.5 and 6.5 show the coarsest level control meshes used to define the models, the middle columns show sample loading conditions used for testing, and the right columns shows the calculated deformed shape. The two rightmost columns are rendered after two levels of subdivision each.

For comparison, we have tested our own MGPCG solver against two other solution techniques, unpreconditioned conjugate gradient and SuperLU [22]. Unpreconditioned CG was selected to compare scaling behavior and contrast the effectiveness of using a multigrid preconditioner in our solver. SuperLU is an efficient general sparse direct matrix solver

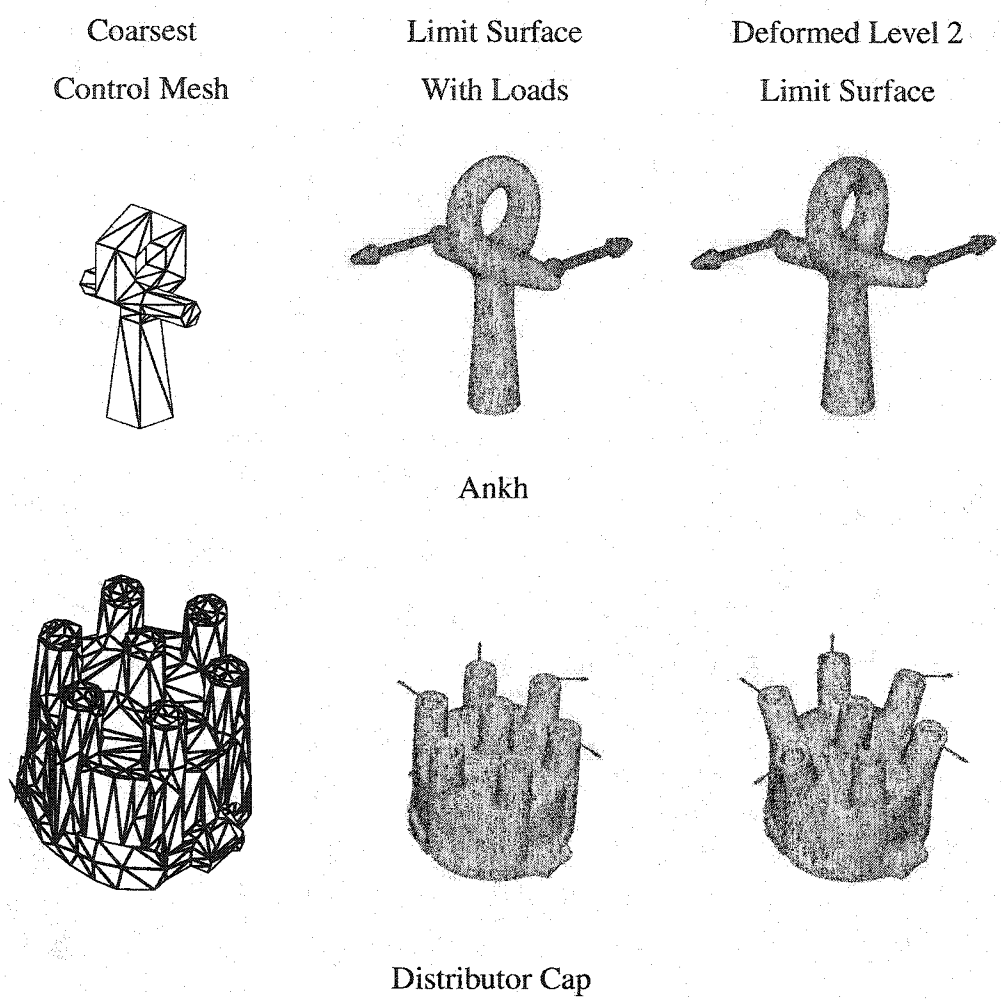


Figure 6.5: Models

that is freely available for academic use. Both MGPCG and CG were run until the solution residual was reduced by a factor of 10^5 ($r_n/r_0 < 10^{-5}$).

Each level of subdivision increases the number of elements and degrees of freedom of the model four-fold (modulo boundary effects). Because of this, setup times for the entire model hierarchy are not significantly larger than for the finest level alone. Each model was simulated for a few different levels of subdivision refinement. The MGPCG solver always used the coarsest level for a base mesh and the number of total levels in the hierarchy

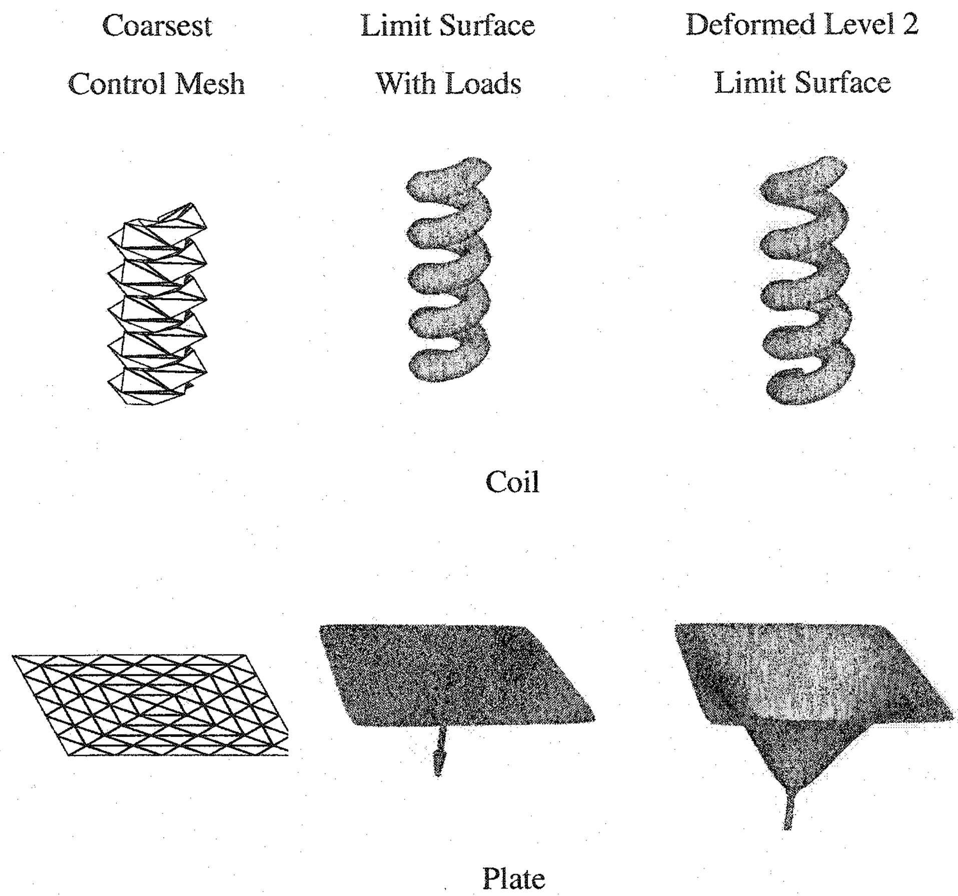


Figure 6.6: Models

was equal to the given level of subdivision. Logarithmic plots of solution time versus the number of model degrees of freedom is shown for selected experiments to demonstrate efficiency and scaling behavior; an algorithm of order $O(N^p)$ will show a slope of p on a log-log plot. A full listing of all results is given in table 6.1 at the end of this section. For the simplest models, the very finest discretizations may be considered overkill; the experiments were performed to best show the scaling behavior of the technique for very large problem sizes. The order of the solution algorithm is estimated by plotting solution time versus number of degrees of freedom of the problem and measuring the slope.

6.4.1 Plate Models

While flat plate models may be of limited geometric interest, flat plates remain of significance in engineering analysis [21] and provide test cases where results can be compared to known analytical solutions. Figure 6.5 shows a flat plate simulated with a central concentrated load. We have also simulated plates with distributed loads and inhomogeneous material properties; in this last case the elastic modulus of one quarter of the plate was increased by a factor of 10. Figures 6.7 and 6.8 plot the time to solution for various levels of subdivision refinement for the concentrated loading and inhomogeneous cases respectively.

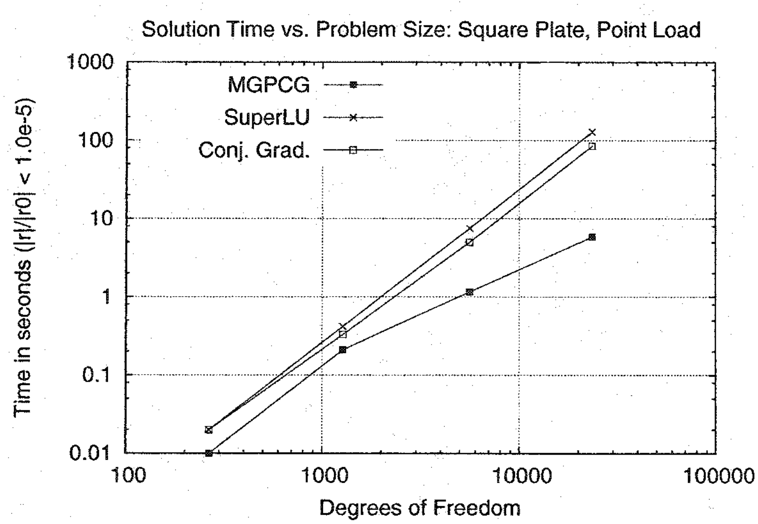


Figure 6.7: Homogeneous Plate, Concentrated Load

Each of these plots demonstrate the superior scaling behavior of our MGPCG solution strategy. In both examples, MGPCG scales at nearly $O(N)$ while SuperLU and CG scale at nearly $O(N^2)$. There is little difference between the plots for the homogeneous square plate under either concentrated or distributed loading (not shown), indicating that it is only the system stiffness matrix, not the load vector which governs solver performance. While CG and SuperLU show nearly identical scaling rates in each case, the material inhomogeneity of the latter experiment greatly increases the condition number of the system and adds a

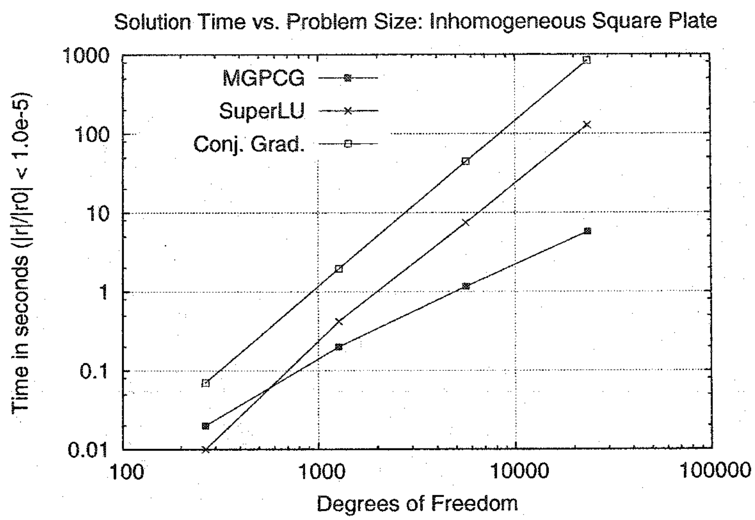


Figure 6.8: Inhomogeneous Plate, Distributed Load

large constant factor to the CG solution time. This trend is not seen with MGPCG, showing its effectiveness. Being a direct solver, SuperLU is completely unaffected by conditioning.

6.4.2 Distributor Cap

The distributor cap is the most complex model tested. The coarsest level mesh has approximately 500 vertices, 1000 faces, and 1500 edges, several of which are sharp. An enlarged image of the solution times are again plotted versus levels of model refinement in Figure 6.9. Here MGPCG shows a marked speedup at all levels of refinement. Scaling improvement is not as pronounced with this simulation. Sharp features make the hierarchy more approximate than for smooth models, but this appeared to have little effect on the convergence of the method. The discretization also contains a number of very high aspect ratio elements which contribute to ill-conditioning. The smooth coil and ankh models showed very favorable scaling and speedup results similar to the plates. The fan model also gained significant speedup from MGPCG.

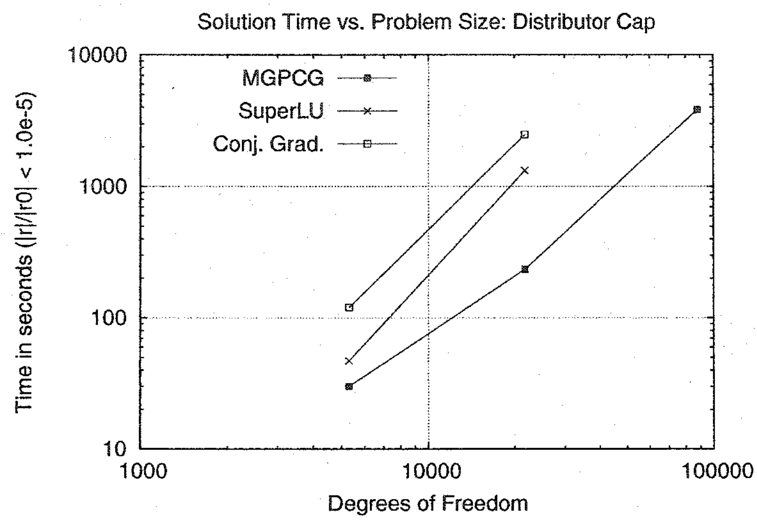


Figure 6.9: Distributor Cap Solution Times

6.4.3 Notes

SuperLU is optimized for very high performance; our MGPCG implementation, while optimal algorithmically, has not been tuned to this level and a re-implementation would likely result in significant constant factor speed increase. Being a direct solver, SuperLU also can consume very large amounts of memory during execution (two or more gigabytes for large problems) and was not used for the very largest simulations. Timings were performed on a 1.2GHz Athlon CPU with 1.5Gb of RAM. Missing entries in Table 6.1 are due to lack of memory for SuperLU or patience for large simulations when MGPCG clearly outperforms other methods. MGPCG requires at least two levels to form a hierarchy, thus no results are listed for the coarsest discretization in the MGPCG column.

6.5 Semi Sharp Features

Sharp features may be represented in the subdivision framework by modifying the subdivision rules along the edge of a sharp crease [38]; unfortunately infinitely sharp edges violate the strict curvature continuity requirements of thin shell theory, and are difficult to evaluate

Table 6.1: Timing Results for MGPCG Benchmark Tests

Model	Subdi-visions	DOF	Time		
			MGPCG	SLU	CG
Coiled Tube	0	336	.	0.06 sec	0.59 sec
	1	1344	0.45 sec	0.69 sec	8.66 sec
	2	5376	2.59 sec	18.05 sec	1.27 min
	3	21504	11.01 sec	11.07 min	9.76 min
	4	86016	50.67 sec	43.46 min	2.46 hr
Square Plate, Distributed Load	0	51	.	0.01 sec	<.01 sec
	1	267	0.02 sec	0.02 sec	0.03 sec
	2	1275	0.22 sec	0.43 sec	0.31 sec
	3	5595	1.19 sec	7.51 sec	5.14 sec
	4	23451	5.77 sec	2.14 min	1.41 min
Distributor Cap	0	1269	.	1.97 sec	3.01 sec
	1	5301	30.12 sec	47.13 sec	2.00 min
	2	21717	3.90 min	22.04 min	41.42 min
	3	88029	1.07 hr	.	.
Fan Disk	0	378	.	0.08 sec	0.68 sec
	1	1302	3.28 sec	1.62 sec	8.12 sec
	2	4803	18.87 sec	31.63 sec	1.86 min
	3	18393	3.07 min	14.25 min	.
Inhomogeneous Square Plate	0	51	.	<.01 sec	<.01 sec
	1	267	0.02 sec	0.01 sec	0.07 sec
	2	1275	0.20 sec	0.42 sec	1.96 sec
	3	5595	1.16 sec	7.49 sec	44.52 sec
	4	23451	5.76 sec	2.14 min	14.01 min
Ankh	0	174	.	0.02 sec	0.18 sec
	1	693	0.41 sec	0.25 sec	2.43 sec
	2	2757	2.50 sec	4.60 sec	19.89 sec
	3	10989	11.69 sec	1.18 min	2.43 min
	4	43869	45.69 sec	33.87 min	.
	5	175293	2.77 min	.	.
Square Plate, Point Load	0	51	.	0.01 sec	<.01 sec
	1	267	0.01 sec	0.02 sec	0.02 sec
	2	1275	0.21 sec	0.42 sec	0.33 sec
	3	5595	1.16 sec	7.50 sec	4.99 sec
	4	23451	5.84 sec	2.15 min	1.42 min

analytically. DeRose *et. al.* [23] introduced the idea of semi-sharp creases which allow selectively sharp corners to be modeled. Semi sharp edges may be created by applying a finite number of sharp subdivision steps, then reverting to the smooth rules thereafter. In our implementation sharp rules are used when subdividing the mesh, and mechanical properties are evaluated using smooth basis functions at the current level; thus a model's sharpness is determined by the current level of subdivision. Like the geometric representation itself, sharp features are refined by subdivision.

Two levels of an example hierarchy for an engineering model of a fan housing are demonstrated in Figure 6.10. The left column shows the coarsest level and the right column shows the model after two levels of subdivision. The top-left image shows the coarsest level control mesh with sharp edges marked, the top-right image shows the same model after two levels of subdivision. The middle-left image shows the very smooth limit surface derived from the coarse mesh with sharp features suppressed; this is the geometry which is defined by the coarsest model. The middle-right image shows the smooth limit surface that is generated after two levels of subdivision using sharp rules; the model still has smooth, finite curvature everywhere, but has much more distinct features. The bottom-left image shows the coarse model loading conditions; the arrow represents a concentrated load applied to the tip of the model. The bottom-right image shows the model after deformation. The computation was done on the model with two levels of subdivision. Simulating the coarsest level does not provide accurate results as the control mesh has too few degrees of freedom, and the very smooth limit surface generated (middle-left) does not model the design features of the model well; however, it can capture the gross motion of the body well and that information is leveraged at finer levels in the hierarchy.

6.6 Constrained Systems

In the previous sections of this chapter it has been assumed that boundary conditions may be expressed in such a way that the system stiffness matrix may be reduced easily, removing constrained degrees of freedom. For general Lagrangian constraints, such as those described in sections 5.2.3 and 5.2.5 this operation is prohibitively expensive. Instead it is customary to create to a block system of the form

$$\begin{bmatrix} \mathbf{K} & \mathbf{C} \\ \mathbf{C}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}, \tag{6.3}$$

where $\mathbf{K}$ is the stiffness matrix of the system and $\mathbf{C}$ is the matrix of constraint derivatives with respect to generalized displacements $\mathbf{u}$. For linear constraints the matrix $\mathbf{C}$ is constant.

The vector of generalized forces $\mathbf{f}$ and the goal value of the constraints $\mathbf{g}$ appear on the right hand side. Lagrange multipliers $\mathbf{v}$ are created as new variables and may be interpreted as the reaction forces required of the constraints to maintain equilibrium.

Equation 6.3 does not lead to a positive definite system of equations and therefore cannot be solved as-is by a multigrid-preconditioned conjugate gradient approach. Bramble and Pasciak have proposed a solution framework which converts symmetric semidefinite block systems to a symmetric positive definite form via a special inner product [13]. They further show how this inner product may be used to implement a conjugate gradient solution algorithm which allows for preconditioning of the block matrix $\mathbf{K}$ to accelerate convergence. We have combined this strategy with the techniques described in [32] to create an efficient multi-level solution scheme for constrained problems.

6.6.1 Lagrange Multiplier Scaling

Using the inner product described in [13] allows conjugate gradient to be applied to the entire system of Equation 6.3, with all block manipulations internal to the matrix multiplication algorithm. A question arises as to how best to determine the termination criteria for the algorithm. In the block system of Equation 6.3, the residual vector consists of both displacements $\mathbf{u}$ and of Lagrange multipliers $\mathbf{v}$. If the relative magnitudes of these individual entries differ greatly the total magnitude of the residual vector will be biased either towards displacements or Lagrange multipliers and the resulting answer will be inaccurate. The block system may be scaled to put the magnitude of the Lagrange multipliers $\mathbf{v}$ on even footing with the displacements $\mathbf{u}$ by the introduction of a scaling parameter β

$$\begin{bmatrix} \mathbf{K} & \beta\mathbf{C}^T \\ \beta\mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v}/\beta \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \beta\mathbf{g} \end{bmatrix}. \tag{6.4}$$

Finding an optimal value of β is not simple for a given system; it may require the problem to be solved several times, using previous (less accurate) solutions to extrapolate a best value. Fortunately for models defined by a multilevel hierarchy, a simple heuristic may be used to find a good value for β. The magnitude of the entries of the displacement vector $\mathbf{u}$ change little between levels of the hierarchy, as results for displacements remain similar regardless of the level of discretization. The elements of $\mathbf{v}$ are Lagrange multipliers, which have physical interpretations as reaction forces. If constraints are specified along edges of the model and implemented at vertices, the following observation may be made for subdivision hierarchies: for each level of subdivision parent edges are split into two child edges for both Catmull-Clark and Loop subdivision schemes. As the number of vertices along an edge doubles, the number of constraints providing reaction forces also doubles, and the expected reaction force per constraint is expected to halve. Given this hypothesis, if β were known for any level of the mesh, it may be easily extrapolated to any other. A direct solution for β_0 at the coarsest level of the mesh may be computed inexpensively via a direct solution technique. Once β_0 is known, β_p for any level p in the hierarchy may be estimated as

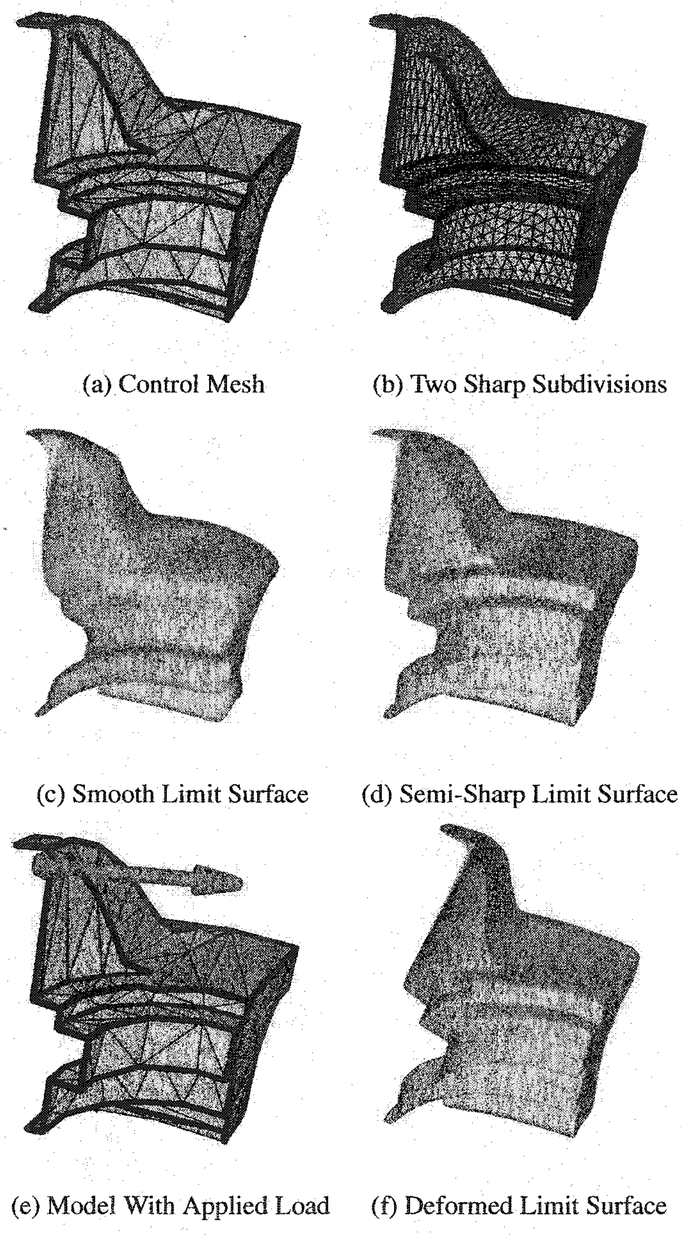
$$\beta_p = 2^p \beta_0. \tag{6.5}$$

This technique works well in practice for constraints which are specified along edges. Overall this solution scheme shows super-linear scaling in the number of unknowns.

6.7 Conclusion

This chapter presented a multigrid-preconditioned conjugate gradient solution scheme for models described by subdivision element hierarchies. When coupled with standard solvers, subdivision simulations do not scale well, and computational cost increases unfavorably as accuracy and the number of degrees of freedom are increased. We presented a technique which exploits the natural hierarchy of a subdivision model to solve problems in demonstrated near $O(n)$ time, that is an amount of time proportional to the number of unknowns in the system. Subdivision approaches for thin shell modeling have been shown previously to be both an elegant and robust description for models of arbitrary topology. In this chapter we have shown that they are efficient for simulation purposes as well. The subdivision operator to be an effective description for inter-level transfer of information in a hierarchical model. Specifically we use the global subdivision matrix as the restriction and prolongation operator in a multigrid-preconditioned conjugate gradient solver implementation.

Subdivision surface basis functions now form a triumvirate of representation techniques for thin shell simulation: they serve to define the model's geometry, deformation and hierarchical description. Subdivision surface modeling shows great promise as a unified framework for describing the geometry, deformation and hierarchical representation of thin plate models.



(a) Control Mesh (b) Two Sharp Subdivisions

(c) Smooth Limit Surface (d) Semi-Sharp Limit Surface

(e) Model With Applied Load (f) Deformed Limit Surface

Figure 6.10: Representation and Deformation of a Hierarchical Fan Housing Model

Chapter 7

BLOOD CELL MEMBRANE SIMULATION

7.1 Introduction

Artificial organ design challenges engineers to create devices that function in a way that is compatible with the workings of the human body. Human blood is susceptible to damage from rough handling which can lead to bruised or burst blood cells. Reducing blood damage to a minimum is critical to the performance of artificial organs which interact with the human vascular system. Currently the artificial organ design cycle requires physical testing at each iteration of the design cycle which is inconvenient, time consuming and expensive. The availability of robust numerical methods for simulation would reduce design cycle times and allow designers insights into behaviors which are currently not possible to achieve.

Mechanical models of human red blood cells are a necessary component for microstructural simulation of blood flow. Red blood cells occupy about 50% of total blood volume and contribute to the flow properties of blood in the human body. When at rest, human red blood cells, known as erythrocytes, assume a symmetrical biconcave shape with a mean size of 8μm x 2μm x 2μm and a membrane thickness of about 10nm; however, these cells distort significantly during flow. Figure 7.1 shows an electron micrograph of a series of erythrocytes flowing through a 12μm arteriole.

Theoretical and numerical treatments of blood as a homogeneous fluid have not been successful at predicting damage to blood cells in flow. Microstructural models are necessary to resolve individual cell deformations and motion, and their interactions with the surrounding fluid plasma. Because of the computational difficulties of resolving tens of

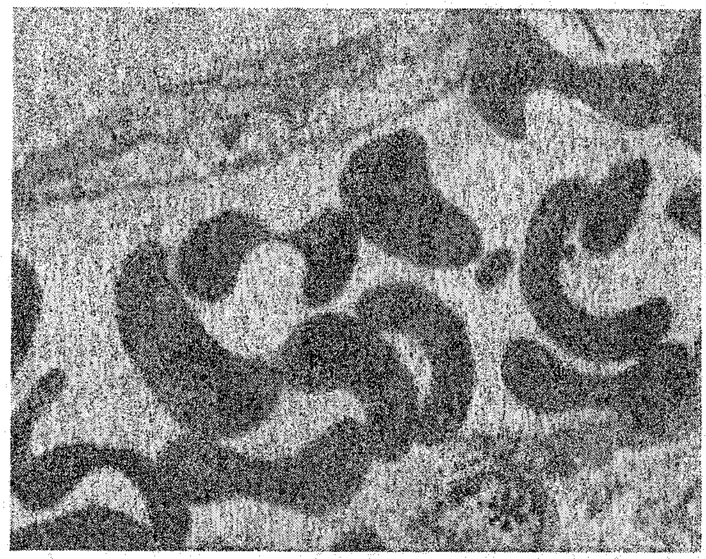


Figure 7.1: Electron Micrograph of Red Blood Cells (reprinted from [2])

thousands of dynamically deforming cellular interfaces, no one to date has simulated realistic blood flows at the microstructural level. We propose the use of subdivision surface based finite element techniques for numerical simulation of red blood cells. Subdivision surfaces have been shown to be a flexible geometrical representation scheme which retain continuity under arbitrary deformations [20, 38, 23]. Furthermore, subdivision surface based finite element schemes provide unified framework for simulation, providing a description of geometry, physics and numerical preconditioning as described in this document and [32, 17].

Figure 7.2 shows a blood cell model described using subdivision surface geometry. Three different configurations of the same model are shown. The ability of subdivision surface thin shell finite elements to compactly represent smooth geometry is a significant asset. Only 42 control points are needed to represent each of the configurations shown. Subdivision models inherently retain geometric continuity under large deformations, no creases or cracks develop in the model as it is distorted.

This chapter is organized as follows, Section 7.2 provides a literature review of red

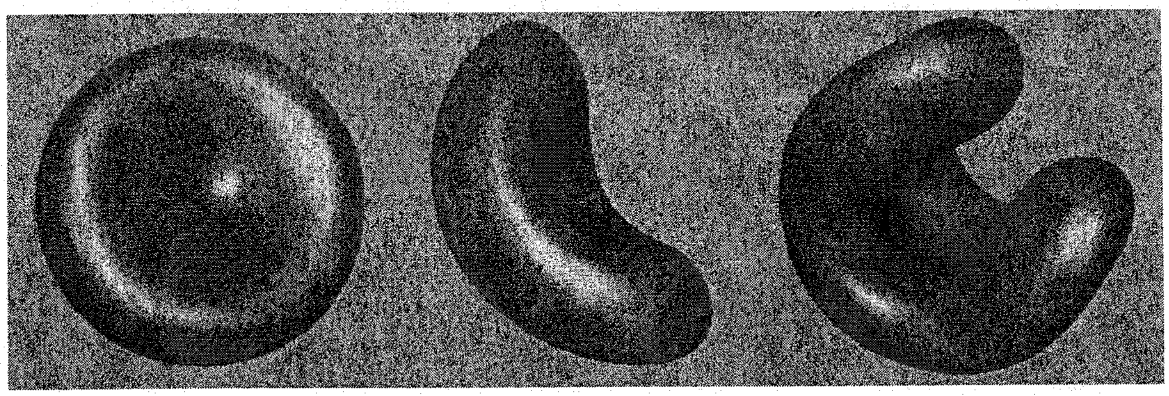


Figure 7.2: Deforming Subdivision Cell Model

blood cell material models, and the remainder of the chapter is devoted to subdivision-based techniques for red blood cell simulation. The process of creating an individual red blood cell subdivision model is described in Section 7.3. The next two sections describe two individual experiments meant to demonstrate the power of subdivision modeling for cell simulation. Our framework for the numerical simulation of micropipette aspiration is presented in Section 7.4. A description of our simulation techniques for cell deformation under point load application are presented in Section 7.5. A summary is provided in Section 7.6.

7.2 Blood Cell Material Models

A material model for red blood cells is a necessary component for numerical simulation. Blood cells are far too small to have their mechanical properties measured directly by conventional means. Instead experiments may be performed which measure global effects on the body of the cell and material properties can be calculated by hypothesizing a relation between the local deformation of the cell membrane and the global properties measured by experiment. Because of the small size of individual cells and their complicated mechanisms of deformation, no single material model has yet proved to be definitive. Several models exist today and are used by various authors. Figure 7.3 shows an artist's conception

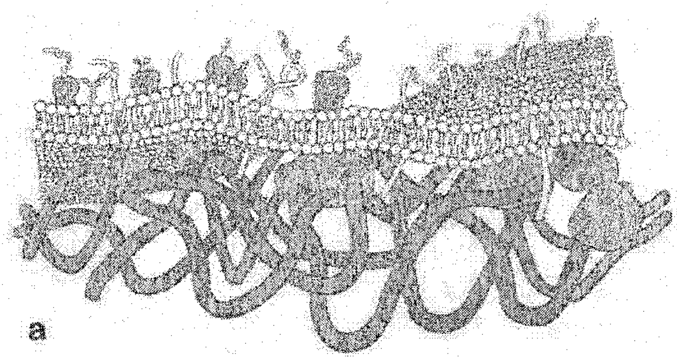


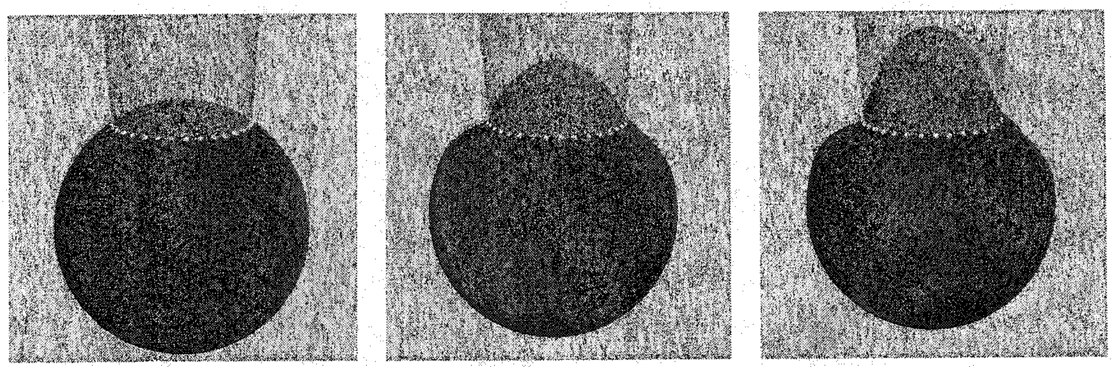
Figure 7.3: Cross Section of Red Blood Cell Membrane (reprinted from [46])

of a nano-scale slice through a red blood cell membrane. The bottom layer is a polymer material, called spectrin. This material is responsible for much of the mechanics of the cell membrane. The cross-linked spectrin network is responsible for the strain-hardening material behavior of cell membranes. A review of material models from current literature follows.

7.2.1 Rand and Burton Model

The first experiments aimed at determining the mechanical properties of red blood cells are those performed by Rand and Burton [52, 51]. Rand and Burton performed micropipette aspiration of red blood cells to measure their mechanical properties. In these experiments, a portion of a cell membrane was drawn into a micropipette by means of a pressure drop. A simulation of this experiment is shown in Figure 7.4; a full description of this simulation process which produced these figures is provided in Section 7.4. For a variety of pressure differences, Rand and Burton measured the length of the "tongue" of the cell, the portion of the cell drawn into the micropipette.

To determine material properties a simple geometric form for the deformed shape of the shell as a function of tongue length was assumed. Rand and Burton furthermore modeled the cell as an elastic membrane with resistance to both stretching and bending enclosing



(a) Rest State (b) Low Pressure (c) High Pressure

Figure 7.4: Simulation of Micropipette Aspiration of a Red Blood Cell

an incompressible fluid center. Using an axisymmetric thin shell model coupled with their enforced kinematic shape and experimental data, they determined the elastic modulus of approximately 0.1 to 10 MPa. (For comparison purposes, Steel has a Young's modulus of 200,000MPa, Wood: 7,000MPa, Rubber: 0.7MPa to 4MPa.) One of the most important results of their work is that it showed the cell membrane to be everywhere materially homogeneous. This overturned the common belief that the biconcave shape of the rest state of the cell was due to differences in material properties across the surface of the cell membrane.

Rand and Burton also noted some failure mechanisms for the cell membrane. They found that cells that were initially in the crenated, or biconcave state, such as those shown in Figure 7.2, deformed considerably during micropipette aspiration. Hypotonically swollen cells (those blown up like a balloon to a spherical shape) deformed only slightly before bursting under pressure. As a sphere is already a minimal surface, any deviation in shape introduces an increase in membrane area. Rand and Burton noted that erythrocytes membranes can sustain large bending strains, but comparatively small extensional deformation.

7.2.2 Boey, Boal and Discher Model

Boey, Boal and Discher [11] have created the most comprehensive cell material model for simulation yet. Their model is hybridized between traditional continuum mechanics ap-

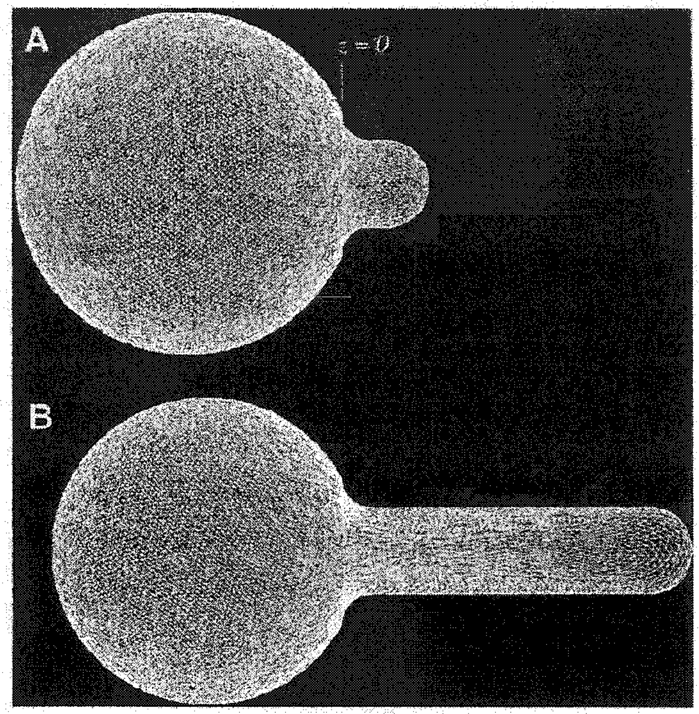


Figure 7.5: Discher *et. al.*Simulation (Reprinted from [24])

proaches and discrete polymer network simulation. The bulk of this approach involves simulating the individual spectrin strands which make up the inner layer of the cell membrane. The proposed energy functional is computed by Monte-Carlo integration of competing attraction-repulsion energies of nodes, located at vertices in a triangulated mesh, and polymer strands which lie along the edges. Each polymer chain is represented by a series of hard beads linked together by tethers with a given maximum extension. The particular choice of tether length, combined with the repulsion between nodes, enforces self-avoidance of the chains and the authors claim that this gives rise to an effective bending resistance for each chain. An approximate curvature energy is formed by computing the deviation between normals of connected triangles.

Discher *et. al.* [24] demonstrate the power of their material model by applying it to a simulation of the micropipette aspiration experiment of Rand and Burton. Figure 7.5

shows two steps from their simulation. In this simulation all points of the cell membrane are constrained to lie on a surface defined by the union simple primitives; a sphere for the body of the cell, and cylinder for the portion of the cell inside of the micropipette, topped with a hemispherical cap.

A simulation of the micropipette aspiration experiment using subdivision surface finite elements is presented in Section 7.4. Our simulation differs from that of Discher *et. al.*in two notable ways: firstly the resulting geometry is emergent from the simulation; and secondly many fewer elements and a simpler material model are used to predict motion. The simulation techniques used in [24] are tractable for simulating a single cell, but are far too complicated to be used for simulating many cells at one time.

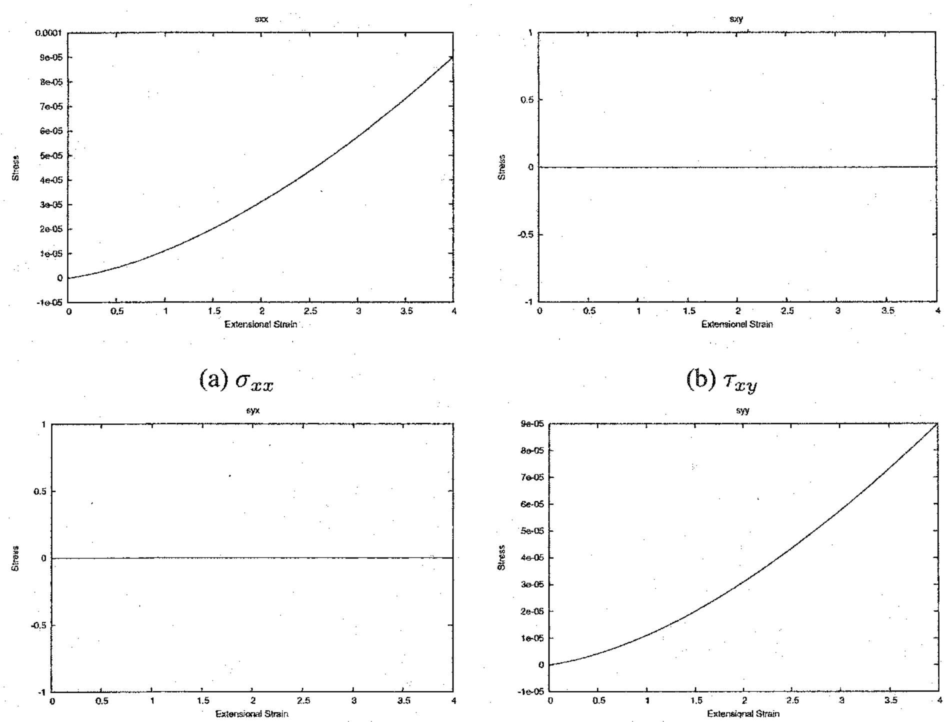
7.2.3 Evans and Skalak Model

Evans and Skalak proposed what is now the most widely accepted elastic model for red blood cell membranes [28]. It is expressed as an axisymmetric energy of the form

$$W_{rbc} = \frac{1}{2}K_{rbc}\left(\lambda_s\lambda_\phi - 1\right)^2 + \frac{1}{2}\mu_{rbc}\left(\frac{\lambda_s}{\lambda_\phi} + \frac{\lambda_\phi}{\lambda_s} - 2\right) + \frac{1}{2}k_{crbc}\left(\frac{1}{R_s} + \frac{1}{R_\phi}\right)^2, \tag{7.1}$$

where λ_s and λ_ϕ are the principal extensions in the azimuthal and meridional directions; R_s and R_ϕ are the principle radii of curvature, and K_{rbc}, μ_{rbc} and k_{crbc} represent the dilational, extensional (shear), and bending moduli respectively [6]. From micropipette aspiration data, these values are given as $K_{rbc} = 500$ dynes/cm, $\mu_{rbc} = 6.3 \cdot 10^{-3}$ dynes/cm, and $k_{crbc} = 0.85 \cdot 10^{-12}$ dynes cm.

This energy form has been generalized to non axisymmetric problems by replacing the terms λ_s and λ_ϕ with general principal extensions [50]. The first term of Equation 7.1 penalizes changes in the area (the product of the extensions) of the cell membrane. The second term penalizes shearing by discouraging differences in the ratio of the principal extensions. The last term is the standard linearized bending term for thin shells, although here only changes in mean curvature are considered, due to the original application to axisymmetric bodies. These terms each have the flavor of those found in traditional continuum



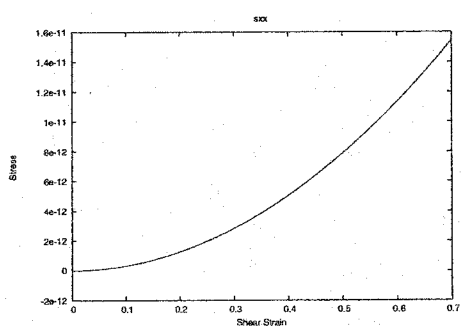
(a) σ_{xx} (b) τ_{xy}

(c) τ_{yx} (d) σ_{yy}

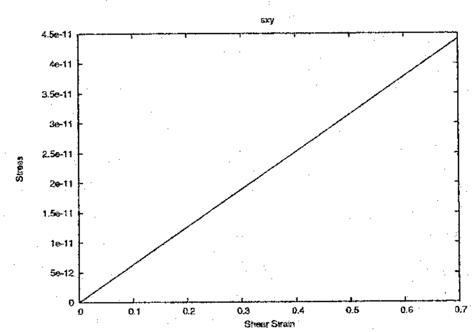
Figure 7.6: Evans-Skalak Material Model for Extensional Strains

mechanics literature, but do not strictly match traditional definitions for these quantities.

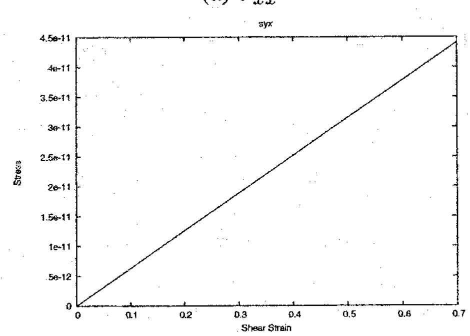
The Evans-Skalak model exhibits strain hardening behavior, as expected to simulate the spectrin layer of polymers, which becomes increasingly difficult to deform as strain is increased. Resultant stresses for the Evans-Skalak model for uniaxial extension and pure shear are shown in figures 7.6 and 7.7 respectively. All of these figures show plots of Cauchy stresses in Cartesian coordinates versus Lagrangian strain. Figure 7.6 plots the four Cartesian membrane stresses under uniaxial extension in the x direction. Both σ_{xx} and σ_{yy} increase and harden as extensional strain ϵ_{xx} increases. Due to the pure area conservation term, K_{rbc}, σ_{xx} and σ_{yy} are equal to one another for uniaxial extension; a feature that is rarely exhibited in engineering materials. Figure 7.7 shows the relationship between Cartesian stresses and pure shearing deformations. Shear stresses vary linearly with applied



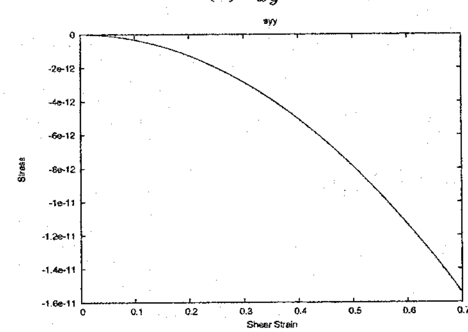
(a) σ_{xx}



(b) τ_{xy}



(c) τ_{yx}



(d) σ_{yy}

Figure 7.7: Evans-Skalak Material Model for Shear Strains

deformation. For small shears, extensional stresses are negligible; for large shears these stresses resist further shearing and can dominate the shear stresses in magnitude.

7.2.4 Discussion

As the extensional energy of the cell membrane is large compared to its bending energy, bending resistance is often ignored in many simulations of blood flow. Instead many authors treat the cell membrane as an inextensional surface with zero bending resistance bounding a fixed volume. Although this simplifies the analysis of the mechanics of the shell greatly, it leads to numerical instabilities as simulated cell membranes often fold or crease or self intersect; something real cell membranes don't do. This fact led Eggleton and Popel [26] to conclude a recent paper with the cautionary warning: "Bending stiffnesses must be included in order to simulate [red blood cells]."

Although the Evans-Skalak model includes bending, the relative magnitudes of the area-conserving, shear and bending terms are a cause for concern. Most classic engineering materials possess a shear modulus roughly one half of the extensional modulus, a fact that Discher *et. al.* [24] note for their material model. For linear engineering materials, these terms are normally related to one another through the Poisson's ratio of the material. It is apparent that the Evans-Skalak model does not follow standard ideas of the relation between extension and dilation. Furthermore the bending energy constant of this model k_{crbc} is 15 orders of magnitude smaller than the extensional constant K_{rbc}. Although these constants possess different units and cannot be compared directly, such a vast difference in magnitudes can cause numerical problems during simulation.

Other material models based on more traditional engineering material models, such as hyper-elasticity are gaining in popularity. Recent work by Pozrikidis in the numerical simulation of blood cells deformed by shear flow uses a hyper-elastic model with success [49].

7.3 Construction of Subdivision Surface Finite Element Blood Cell Models

Red blood cells are flexible and change shape in the blood stream due to the chemical composition and dynamics of the surrounding blood plasma. During rapid flow blood cells assume a bullet or bullet-like shape; when flow stops cells become biconcave disks, indicating a preferred rest state. If the chemical concentration of the surrounding blood plasma is changed appropriately, red blood cells will swell osmotically and become increasingly spherical. Irrespective of the amount of swelling, red blood cells at rest are characterized by having a near perfect radial symmetry. Red blood cells may be hypothesized to have a preferred rest state, and return to this shape when all deformation-causing loads are removed. The exact shape of the rest state is a function of the chemical composition of the environment in which the cell is immersed. The shape of the red blood cells used for experiment are discussed in Section 7.3.1

A blood cell model for numerical simulation requires a description of the geometry and material properties of each cell. Our process for creating human red blood cell models for simulation consists of four steps:

- Sampling the surface geometry of a known cell membrane configuration
- Constructing a subdivision surface representation of cell geometry from the surface samples
- Adding material properties
- Applying loads and forces of constraint

7.3.1 Red Blood Cell Shape

Evans and Fung performed experiments to measure the shape of a red blood cell for varying chemical concentrations (tonicities) of surrounding media [27, 30]. They proposed a

(a) 300 mosmol (b) 217 mosmol (c) 131 mosmol

Figure 7.8: Surface Sample Points

mathematical model of cross-sectional thickness t as

$$t(r) = \sqrt{1 - \left(\frac{r}{R_0}\right)^2} \left[C_0 + C_2 \left(\frac{r}{R_0}\right)^2 + C_4 \left(\frac{r}{R_0}\right)^4 \right], \tag{7.2}$$

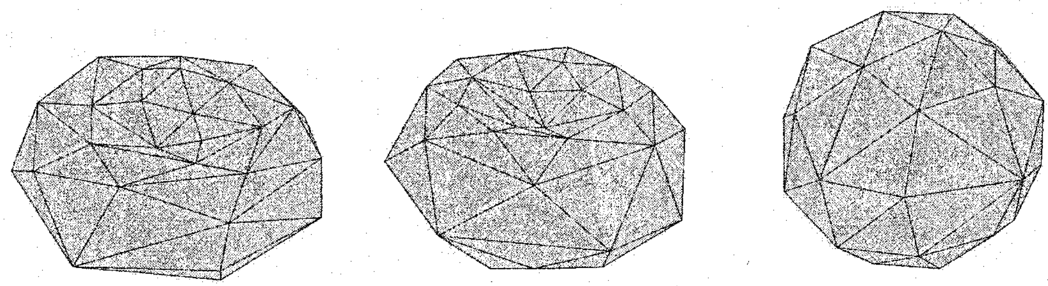
where the radius, r is the distance from the axis of symmetry and C_0, C_2, C_4 and R_0 are numerical coefficients determined from experiment. Using this formula for the thickness of a red blood cell cross section, a dense point sampling of the cell surface may be obtained, as shown in Figure 7.8. Each of these point samplings corresponds to a given tonicity of the surrounding media measured in mosmol (saline concentration). At 300 mosmol the solution is considered isotonic. Approximately 1500 surface samples were used in each example.

7.3.2 *Subdivision Control Mesh Generation*

We apply the technique of Hoppe et. al. [38] to the sampled cell surface data to create a smooth subdivision surface representation of the surface of the cell. The reconstruction process involves three distinct steps.

1. Creation of a fine triangulated surface from the sample data.

2. Creation of a coarse triangulation from the output of step 1.



(a) 300 mosmol (b) 217 mosmol (c) 131 mosmol

Figure 7.9: Coarse Subdivision Control Meshes

3. Creation of an optimal subdivision surface control mesh from the output of step 2.

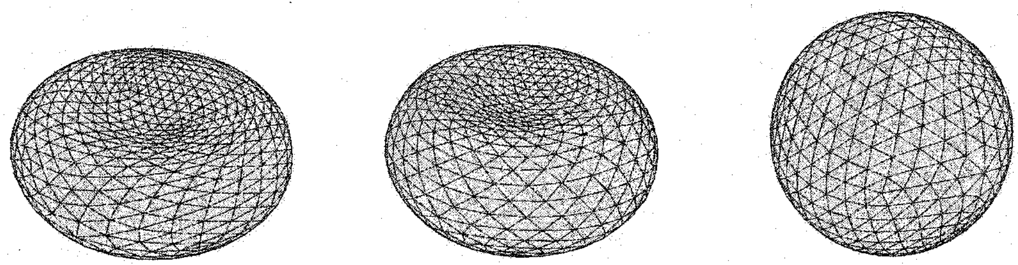
An automatic approach to steps 1 and 2 are described by Hoppe et. al. in [39, 40]; their approach to step 3 of the process is detailed in [38].

The results of this process applied to our surface sample data sets are the coarse subdivision control meshes shown in Figure 7.9. Each of the illustrated control meshes contain approximately 100 faces. The control meshes themselves do not interpolate the sample data points; instead they serve as input to a refinement process by which a smooth subdivision surface is created. Refined subdivision surfaces for each of our example models are shown in Figure 7.10. These refined meshes interpolate the surface sample data to a high degree of accuracy even though the control meshes which generate them contain relatively few faces.

7.3.3 Graded Subdivision Control Mesh Generation

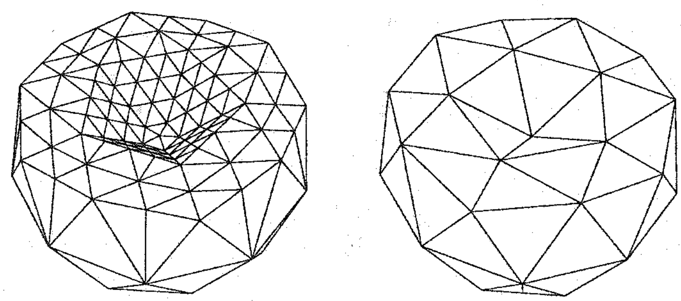
The meshes shown in Section 7.3.2 are the result of a uniform sampling of the surface. As a result each of the triangulated elements have approximately the same area. For specialized applications, a graded mesh in which smaller elements are concentrated in desired locations will increase simulation accuracy and efficiency.

A graded mesh may be generated by creating a coarse triangulation with desired sampling properties and substituting this result for steps 1 and 2 of the process of Hoppe et. al.



(a) 300 mosmol (b) 217 mosmol (c) 131 mosmol

Figure 7.10: Refined Subdivision Surface Reconstructions

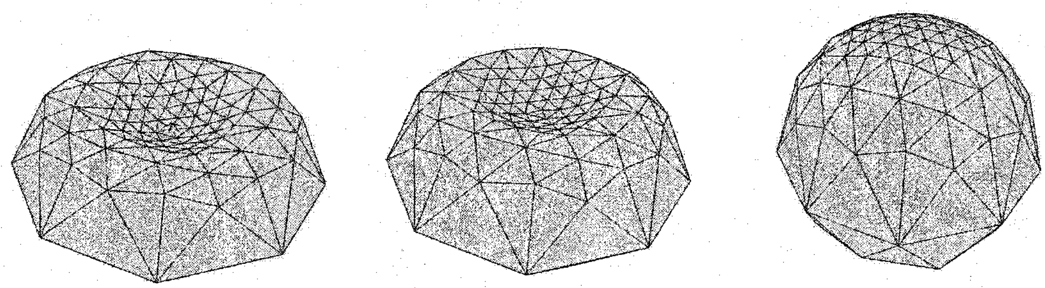


(a) Top View (b) Bottom View

Figure 7.11: Graded Faceted Mesh

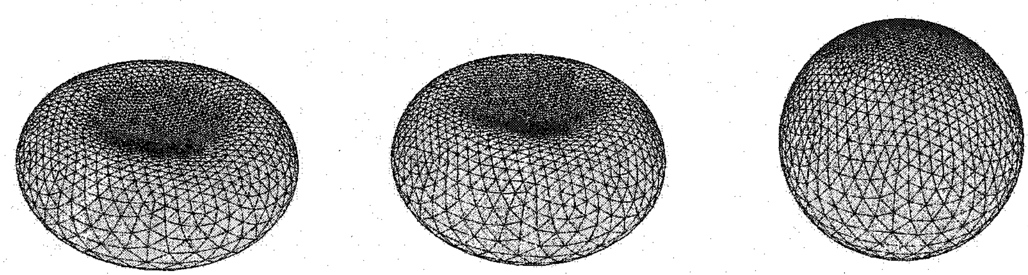
For simulation of cell micropipette aspiration the central region of the face of one side of cell is of special interest. A faceted mesh with a region of high sampling around the center was created by hand and is shown in Figure 7.11.

This mesh has only the desired arrangement of elements, but does not serve as an accurate representation of the surface of the cell. To produce an optimal subdivision surface control mesh, the vertices of the mesh illustrated in Figure 7.11 are first projected onto the surface surface data set, resulting in a better overall fit. This new mesh is used as input to step 3 of the process of Hoppe et. al. The result is a subdivision control mesh that produces a limit surface which tightly fits the surface sample data, while retaining the desired



(a) 300 mosmol (b) 217 mosmol (c) 131 mosmol

Figure 7.12: Graded Coarse Subdivision Control Meshes



(a) 300 mosmol (b) 217 mosmol (c) 131 mosmol

Figure 7.13: Refined Graded Subdivision Surface Reconstructions

element distribution chosen a priori. Examples of graded subdivision control meshes are shown in Figure 7.12. The limit surfaces produced by these control meshes are shown in Figure 7.13.

7.4 *Micropipette Aspiration Simulation*

Human red blood cells consist of an outer membrane which surrounds a fluid interior. Experiments suggest that the interior of the cell may be easily modeled as a homogeneous fluid exerting a hydrostatic pressure, while the behavior of the outer membrane is governed by the solid cytoskeleton which resists both stretching and bending deformations [52]. Our experiment numerically simulates the micropipette aspiration of red blood cells with various initial shapes. A schematic overview is shown in Figure 7.14. The simulation starts with the cell membrane and pipette wall in an initial configuration. Next loads are added to simulate the pressure drop inside of the pipette and constraints are added to prohibit the cell membrane from piercing the walls the pipette. Many steps are taken, starting from zero load and increasing to the full amount desired. Once all of the loads and constraints are in place, a new equilibrium condition can be solved for. Once the new configuration is known, material properties can be updated and the process may be repeated for a higher value of load.

In order to the behavior of the cell we make the following hypotheses for the purposes of simulation:

1. The membrane is an elastic thin shell with a finite resistance to changes in area and curvature.

2. The biconcave membrane state is the zero stress state.

3. The cell is in static equilibrium at all times.

4. The cell membrane surrounds an inviscid incompressible fluid which does not flow across the membrane.

5. No friction exists between the cell membrane and the pipette mouth.

6. The cell membrane remains in contact with the circular mouth of the micropipette at all times.

Hypotheses 1, 2 and 3 lead to a weak form of equilibrium for thin shells:

$$\delta\Pi = \int_S \delta\boldsymbol{\kappa} : \mathbf{m} + \delta\boldsymbol{\epsilon} : \boldsymbol{\sigma} - \delta\mathbf{u} \cdot \mathbf{f}\, dS = 0 \quad \forall\, \delta\boldsymbol{\kappa}, \delta\boldsymbol{\epsilon}, \delta\mathbf{u}, \tag{7.3}$$

where $\delta\boldsymbol{\kappa}$ is the admissible virtual strain due to changes in curvature between the deformed and undeformed configurations of the body, $\mathbf{m}$ is the corresponding internal moment tensor, $\delta\boldsymbol{\epsilon}$ and $\boldsymbol{\sigma}$ are the virtual membrane (in-plane) admissible strain and stress tensors respectively, $\delta\mathbf{u}$ and $\mathbf{f}$ are the virtual displacement and applied surface load vectors representing external work, and ":" represents tensor scalar contraction [55].

The effect of cell plasma is described by hypothesis 4. In the absence of viscosity or mass transfer, the fluid center of the cell acts as a volume preservation constraint; the cell membrane may change shape and area, but the volume it surrounds must remain constant. Mathematically we represent this simply as

$$\delta V = 0. \tag{7.4}$$

Friction is assumed to be negligible in our simulation per hypothesis 5. At each step of the simulation the cell is assumed to be in equilibrium with its surroundings. Strictly speaking our hypotheses state that the cell deforms instantly when acted upon by outside forces. While this assumption is not realistic when loads are varied quickly, it provides a useful approximation of cell behavior when loads are applied slowly with respect to any dynamic effects of the cell material and surrounding media. The last hypothesis 6 incorporates the boundary contact requirement necessary for the aspiration of the cell.

7.4.1 Discretization

The hypotheses given in the preceding section give laws for a continuous representation of the cell at all time. Closed form solutions to Equation 7.3 cannot be found for arbitrary starting geometries, let alone include the constraint of contact with the pipette mouth. To create a computationally tractable simulation we discretize the geometry, loading, material behavior and external constraints of the cell.

We choose to represent the cell as a finite element model using smooth subdivision surface basis functions. We also geometrically discretized the mouth of the pipette by modeling it as as a sequence of positional and tangential constraints which allows the cell

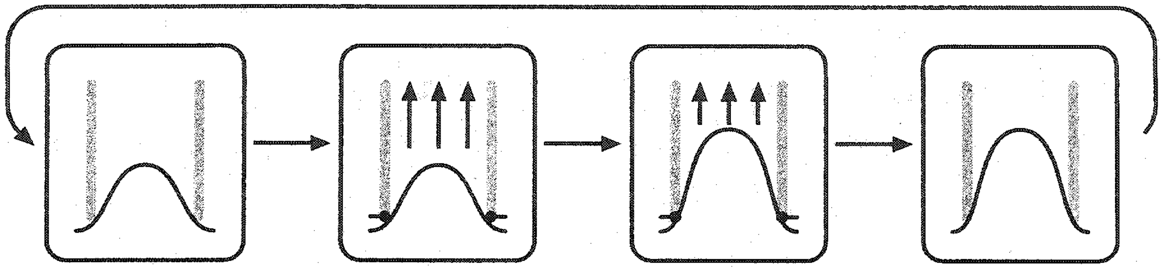


Figure 7.14: Simulation Overview

body to slide past but not penetrate the wall of the pipette. Applied suction loads are also discretized. Our simulation consists of the application of a number of small load increments, each corresponding to an equilibrium pose of the cell. The load, initially zero, is increased in small steps. At each load step the equilibrium shape of the body is computed and used as the starting point for the next iteration. This formulation allows us to easily simulate non-linear geometric properties with a stepwise linear material model.

7.4.2 Cell Membrane Model

We model the cell as a Kirchhoff-Love thin shell which includes both stretching and bending deformations with a statement of virtual work corresponding to Equation 7.3. The membrane of the cell is modeled geometrically as a collection of subdivision surface elements which define the geometry and mechanical properties of the cell membrane.

7.4.3 Cell Interior Model

The incompressible nature of the fluid interior of the cell is modeled via a single volume preservation constraint. The volume preservation constraint is simple to calculate because every degree of freedom of a subdivision finite element model is a generalized displacement. Computing volume integrals directly on volumes bounded by subdivision surfaces is difficult because the bounds of integration are complicated. The divergence theorem states

$$\int_V \nabla \cdot \mathbf{F}\, dV = \int_{\partial V} \mathbf{F} \cdot \mathbf{n}\, dA, \tag{7.5}$$

for any vector function $\mathbf{F}$ where V is the volume of the body, ∂V is the surface, $\mathbf{n}$ is the normal to the surface and dV and dA in infinitesimal volume and surface elements respectively. This theorem may be used to convert volume integrals with complex bounds of integration to surface integrals for which the bounds of integration are known. Specifically the volume of a closed subdivision surface is defined to be

$$V = \frac{1}{3} \sum_{\partial V} \int_{\partial V_j} \mathbf{x} \cdot \mathbf{n}\, dA, \tag{7.6}$$

where ∂V_j is the j'th parameterized face of the surface and $\mathbf{x}$ is the Cartesian position vector of the surface $[x, y, z]$ which satisfies the relation $\nabla \cdot \mathbf{x} = 3$. Each point on the surface of the initial configuration of the cell membrane is defined piecewise over all faces j as

$$\mathbf{x}(\theta^1, \theta^2) = \sum_i N_i(\theta^1, \theta^2)\mathbf{x}_i, \tag{7.7}$$

where the N_i are the subdivision basis functions of the j'th face and $\mathbf{x}_i$ are the control vertices of the neighborhood of face j. The deformed configuration of the surface takes on a similar form

$$\tilde{\mathbf{x}}(\theta^1, \theta^2) = \sum_i N_i(\theta^1, \theta^2)(\mathbf{x}_i + \mathbf{v}_i), \tag{7.8}$$

where the $\mathbf{v}_i$ are the generalized displacements of the surface. The difference in volume δV between the undeformed and deformed configurations of the body is therefore

$$\delta V = \sum_{\partial V} \int_{\partial V_j} [\tilde{\mathbf{x}} \cdot \tilde{\mathbf{n}} - \mathbf{x} \cdot \mathbf{n}]\, dA. \tag{7.9}$$

This expression is nonlinear in displacement as both the deformed surface and the deformed unit normal $\tilde{\mathbf{n}}$ depend on displacement. If we introduce the approximation $\mathbf{n} \approx \tilde{\mathbf{n}}$ then the expression simplifies considerably to

$$\delta V = \sum_{\partial V} \int_{\partial V_j} \sum_i N_i(\theta^1, \theta^2)\mathbf{v}_i \cdot \mathbf{n}\, dA. \tag{7.10}$$

This new expression is linear in displacement and leads to a simple linear form of the volume preservation constraint. There is some error in making this approximation. However it is first order accurate in practice which is commensurate with the approximation error of the rest of our simulation. Over successive solutions a penalty term inversely proportional to the net volume gain can be added to ensure the body does not continue to expand.

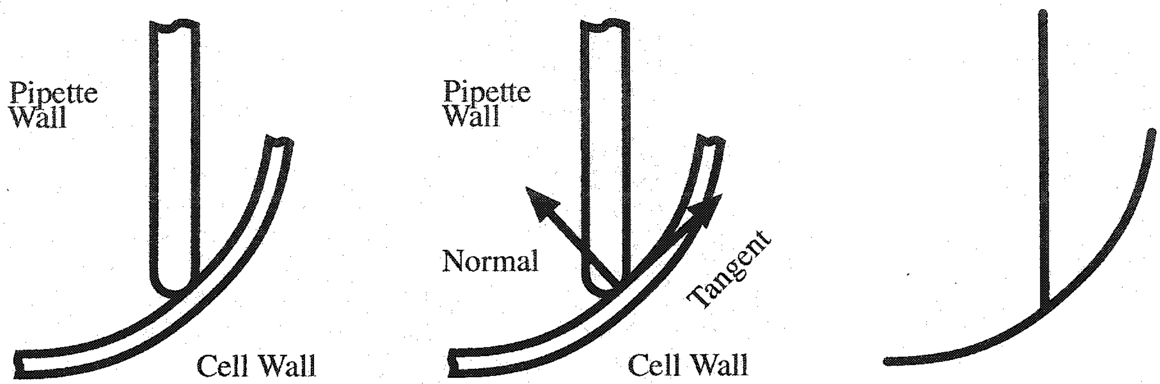
7.4.4 Load Model

The applied pressure drop within the micropipette is modeled as a uniform applied vertical loading within the interior of the pipette volume. Loads are applied by approximating their effect through numerical quadrature; at each of the quadrature evaluation points the value of the load is calculated and added to the model. The applied pipette pressure load value will be zero for all evaluation points which do not lie within the interior volume of the pipette.

7.4.5 Pipette Model

A glass pipette is orders of magnitude stiffer than a human red blood cell and is well modeled for our purposes as a rigid cylinder. During the aspiration process, the membrane of the cell maintains contact with the smooth mouth of the pipette while simultaneously sliding along it. We simulate this behavior via the application of constraints applied to the cell surface at the points of contact with the mouth of the pipette.

Physically, both the cell membrane and the pipette wall have a finite thickness as illustrated in two dimensions in Figure 7.15(a). The mouth of the pipette has a finite radius of curvature, and at the point of contact the tangent of the cell surface and the pipette mouth will be equal as shown in Figure 7.15(b). To maintain sliding contact the surface of the



(a) Thick Model (b) Tangent Directions (c) Idealized Thin Model

Figure 7.15: Cell-Pipette Contact Model

cell may not move in the direction of its own surface normal, though during motion the direction of the surface normal may change. This is also true in three dimensions. Because the wall of the pipette and the cell membrane are thin we idealize them both as thin surfaces as shown in Figure 7.15(c).

For simulation we discretize the pipette into several points around the circular mouth as shown in Figure 7.16; a relatively coarse model is shown for purposes of illustration. The shaded area represents the wall and interior of the pipette. The dark dots indicate the discretized points of contact between the pipette mouth and the cell membrane. Only the mouth of the pipette is considered in our simulation. The amount of discretization of the pipette mouth is chosen so that there is roughly one point for each discretized face of the cell model in contact with the pipette mouth; using fewer contact points compromises the accuracy of the simulation, while applying more constraints than contacting faces risks over-constraining the solution. Lagrangian constraints are applied at each of these points to simulate contact between the cell membrane and the pipette mouth.

We model contact with the pipette mouth in a two-step process. Physically the cell slides in such a way that contact is always maintained between the cell membrane and the mouth of the pipette for all infinitesimal load steps during the simulation. For finite load

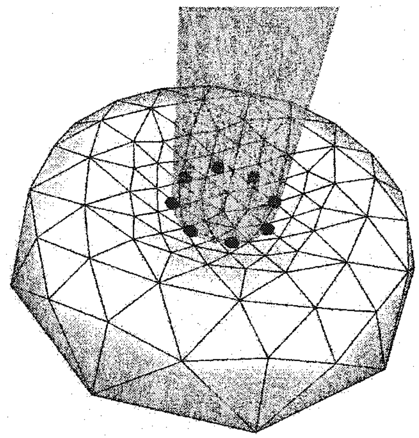
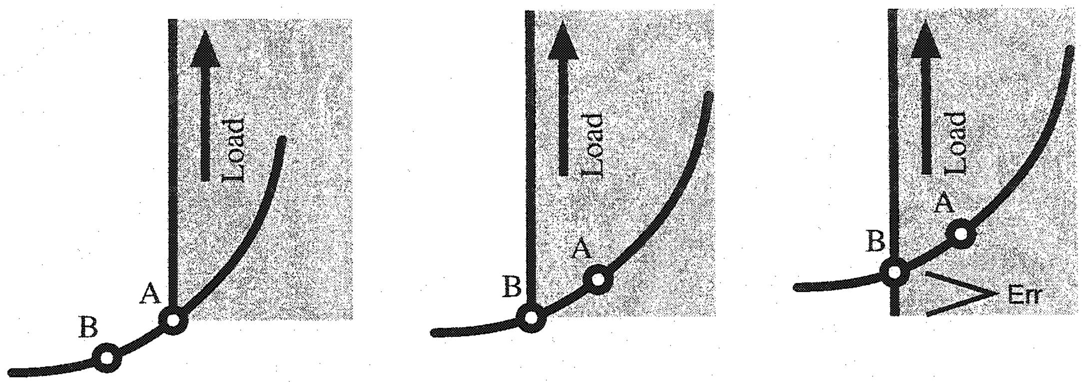


Figure 7.16: Discretization of Pipette Mouth

steps, constraints must be constructed to account for discretization errors.

Sliding contact is illustrated in Figure 7.17; only two dimensions are shown for simplicity. In each of these diagrams the cell membrane (illustrated by the curved line) is in contact with the pipette wall (the vertical line). The interior of the pipette is indicated by the dark shading. The suction force present inside of the pipette is illustrated by the vertical arrow inside of the pipette. Ideal sliding during a finite load step is illustrated (in two dimensions for simplicity) in Figures 7.17(a) and 7.17(b). Two points are shown, point "A" which is initially in contact with the pipette mouth, and point "B" which is the point which will come into contact with the pipette mouth at the end of the load step. Without the application of any kind of boundary condition applied to the cell membrane, both point A and B would move vertically upwards penetrating the pipette mouth. Ideally one would like to apply a constraint to point "B" to lie exactly on the mouth of the pipette to ensure that penetration does not occur. The problem with this approach is that the initial location of point "B" is not known a priori and there is no closed-form expression to compute it. Computing the initial location of point "B" may take many nonlinear iterations to solve and adds great expense to the computation.

Instead, we propose a two-step prediction and correction simulation process. An initial



(a) Initial Confi guration (b) Idealized Sliding (c) Possible Error

Figure 7.17: Sliding Contact Between Cell and Pipette Wall

configuration is assumed in which the surface of the cell lies exactly on the discretized points of the pipette mouth; point "A" in Figure 7.17(a) is an example of one of these points. A constraint is applied to each of these points that they not move in the direction of their own surface normal. Differentially this implies that points touching the mouth of the pipette move only in a tangential direction, inhibiting penetration. Using these constraints and the applied load, a new equilibrium position is calculated. The point initially in contact with the pipette mouth will move only in a tangential direction to the pipette mouth.

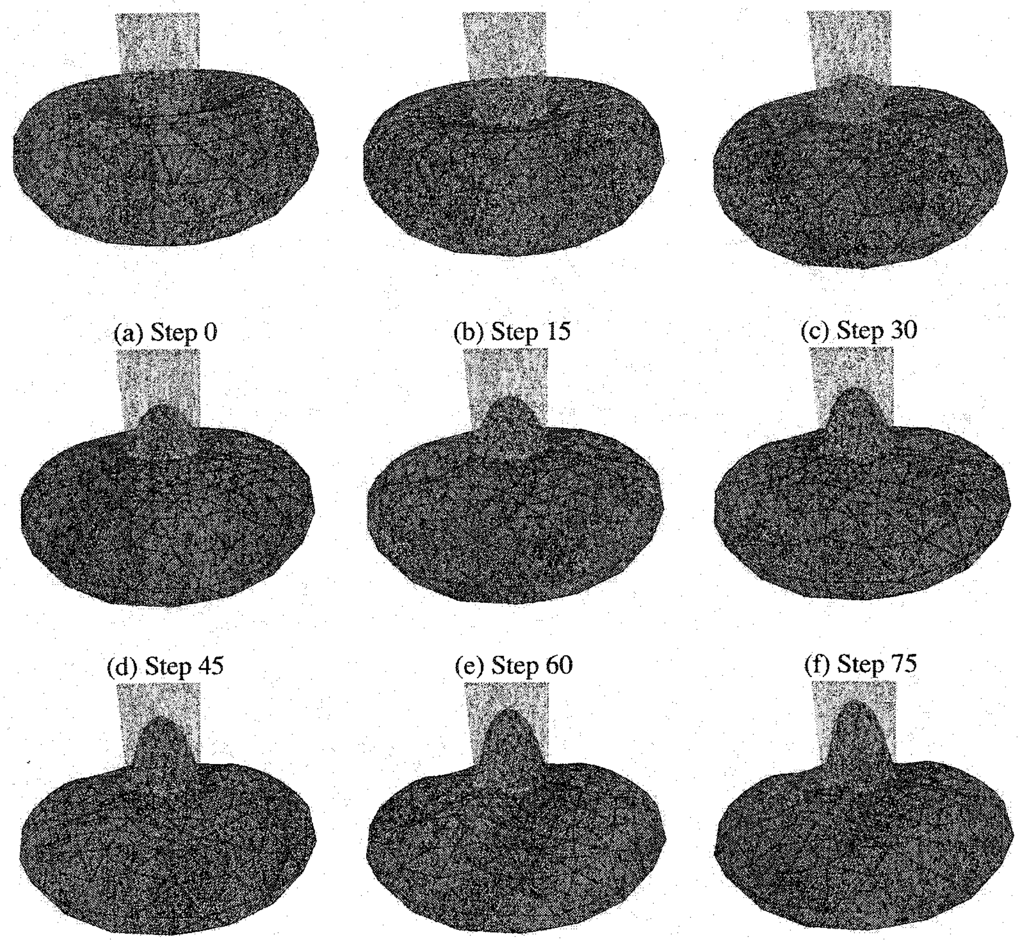
In a differential sense any point "B" located an infinitesimal distance away from point "A" would remain in perfect contact with the pipette mouth. The point that should now lie directly on the mouth of the pipette is assumed to be point which lies directly above or below the discretized point of the pipette mouth, along the axis of the pipette. This point may be found using a few newton iterations to solve for the intersection of a line extending from point of application of the constraint parallel to the axis of the pipette. Because finite load steps are taken, this point may not lie exactly on the mouth of the pipette as shown exaggerated in Figure 7.17(c). The tangential movement constraint applied to our initial point ensures that the penetration of the new point into or away from the pipette mouth should be small. Once the point has been located, a new constraint is applied which positions it to be in contact with the mouth of the pipette with no gap or penetration. The process of applying first a tangential constraint, then a positional constraint is repeated then for each load step.

7.4.6 Results

A sequence of simulation results using a linear material law are shown in Figures 7.18 and 7.19. A plot of tongue length versus applied load is shown in Figure 7.20. The robustness of this physical simulation is excellent. The material used in this simulation had a Poisson ratio $\nu = 0.3$, a thickness to major radius ratio of 800 (equivalent to $8\mu m/10nm$). The applied load was $2 \cdot 10^{-5}$ times the elastic modulus of the membrane for each load step (roughly 100 load steps were simulated).

Figure 7.18 shows the control mesh used for simulation along with a visualization of the rigid pipette. The models consists of 880 elements and 442 vertices. Figure 7.19 shows the smooth limit surface for the control meshes of Figure 7.18. Wrinkling behavior can be seen throughout the simulation; this is due to the material attempting to resist changes in area as the cell model is squeezed through the neck of the pipette. These wrinkles are expected; even though the applied loading and constraints are axi-symmetric, the resulting deformation need not be. Axisymmetric solutions for this simulation are not stable and wrinkled shapes minimize the energy of the system. If wrinkling phenomena limit the accuracy of the solution then some regularization energy may be required to smooth the solution.

The shape and character of the plot of Figure 7.20 qualitatively agree well with experimental results [52], but the numbers predicted by the linear elastic material model disagree significantly. A more realistic cell material model is needed to predict quantitative results. Elongations of up to 300% are found in elements near the neck of the pipette; because of the linear nature of the material model, these strains are not penalized any more than any other deformation of the system; a strain-hardening material law could limit these strains to values more in line with experimental data.



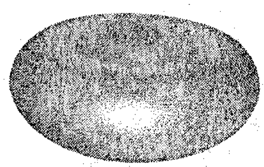
(a) Step 0 (b) Step 15 (c) Step 30

(d) Step 45 (e) Step 60 (f) Step 75

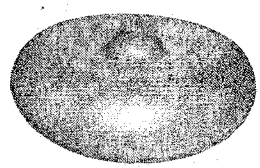
(g) Step 90 (h) Step 105 (i) Step 120

Figure 7.18: Micropipette Aspiration Simulation

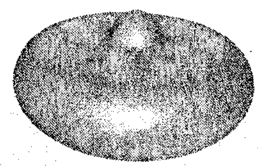
Unfortunately the robustness of the simulation using a linear elastic material law did not carry over to the use of the Evans-Skalak model. Numerical problems including ill-conditioning of the stiffness matrix and element inversions prevented the simulation from yielding valuable results. More discussion of the numerical problems encountered using the Evans-Skalak law are discussed in Section 7.5.



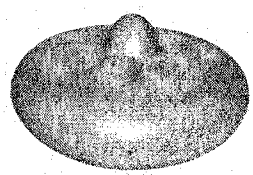
(a) Step 0



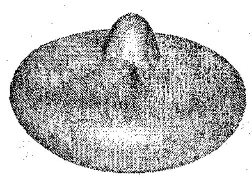
(b) Step 15



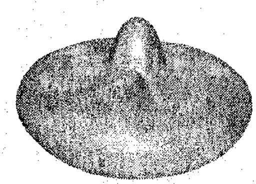
(c) Step 30



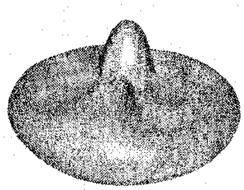
(d) Step 45



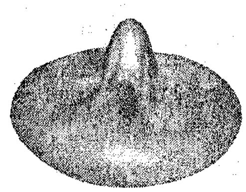
(e) Step 60



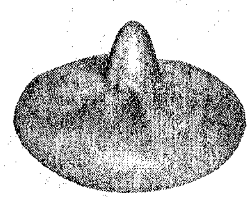
(f) Step 75



(g) Step 90



(h) Step 105



(i) Step 120

Figure 7.19: Micropipette Aspiration Simulation

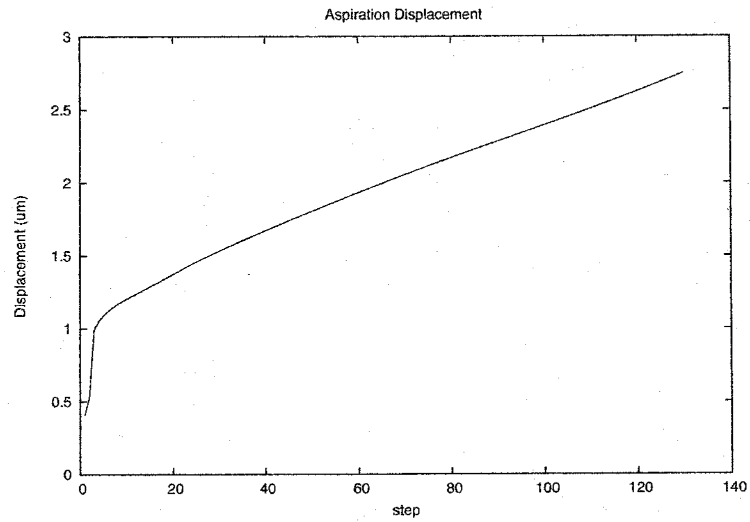


Figure 7.20: Deformation Versus Load Step for Aspiration

7.5 *Point Load Deformation of a Red Blood Cell*

As of this writing, Jim Antaki of the Sangria Project [10] is planning to conduct mechanical experiments on red blood cells using a scanning tunneling microscope (STM) which is capable of applying extremely localized displacements to cells and measuring the reaction force. The experimental setup, illustrated in Figure 7.21, consists of a red blood cell lying on a flat surface. The tip of the STM will be brought into contact with the cell membrane, and relationships between deformation and applied load will be measured. The goal of this section is to provide a description of the subdivision finite element framework used to simulate this experiment.

7.5.1 Modeling and Loading

The geometrical model we chose for our experiments is the biconcave (300 mosmol) model, as described in Section 7.3.3. Two material models were used, a standard linear elastic material, and the Evans Skalak model. The rigid, flat surface which the cell rests on was modeled as frictionless; its effect was implemented using direction constraints which only

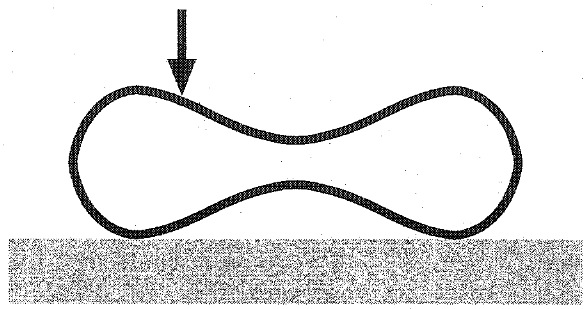
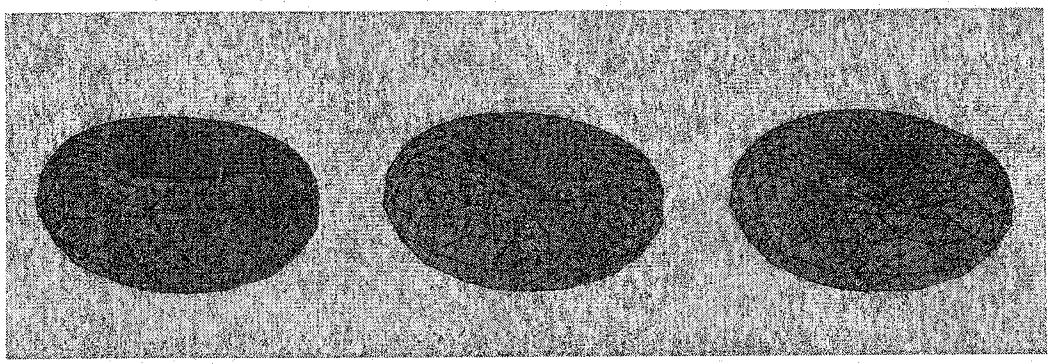


Figure 7.21: Deformation of a Red Blood Cell by STM Tip

allow movement in a direction transverse to the rigid surface. This constraint was applied to all points which came into contact with the rigid surface. It is assumed that once a point contacts the bounding lower surface, it does not release. One constraint is applied to the point furthest from the point of applied load to prevent spin around the vertical axis. The tip of the STM device was modeled as a point load; a pressure is applied in the vertical direction, while movement in the transverse direction at this point is constrained to be zero. The volume preservation constraint described in Section 7.4.3 is used to simulate the incompressible nature of the interior of the cell.

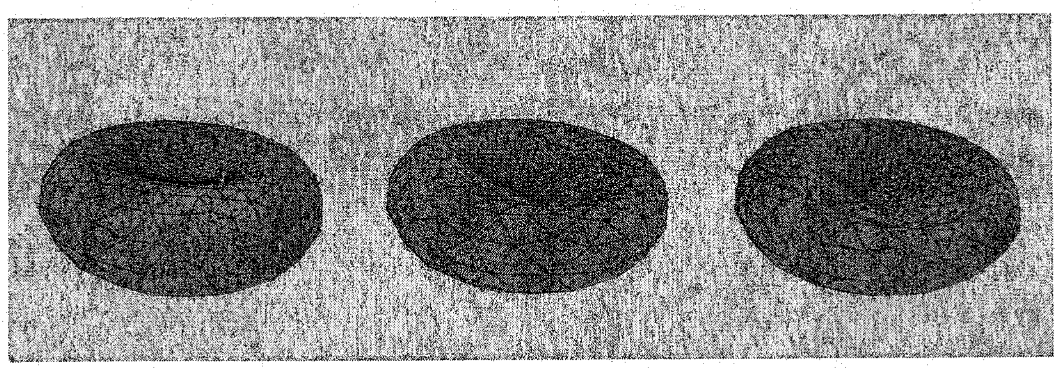
Figures 7.22 and 7.23 show stages of the simulation for two cell models with equivalent linear material properties (E=200e9, ν=0.3, thickness=0.01, width=8.0). The simulation shown in Figure 7.22 incorporates volume conservation, while that shown in Figure 7.23 does not. Qualitatively the difference between these two simulations is readily apparent; when volume conservation is enforced both stretching and bending stresses are significant load carrying mechanisms; without volume conservation, bending resistance carries most of the load. In Figure 7.22 the deformation caused by the point load is much more localized and acute, while if Figure 7.23 the entire cell model deflates more uniformly.

Quantitative comparisons between these two simulations are shown in Figure 7.24. These plots illustrate the relation between the force of the applied point load and the vertical displacement at that point. Including volume conservations stiffens the system by a



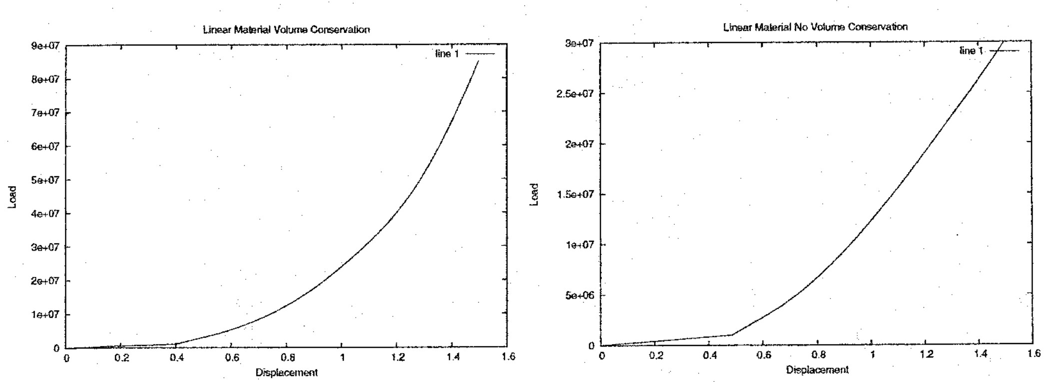
(a) Small Load (b) Medium Load (c) Large Load

Figure 7.22: Simulation with Volume Conservation



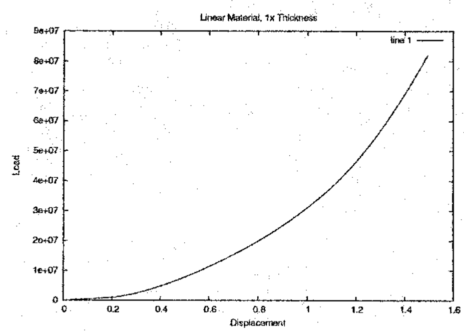
(a) Small Load (b) Medium Load (c) Large Load

Figure 7.23: Simulation without Volume Conservation

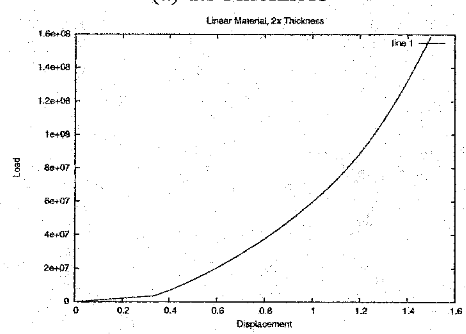


(a) Volume Conservation (b) Without Volume Conservation

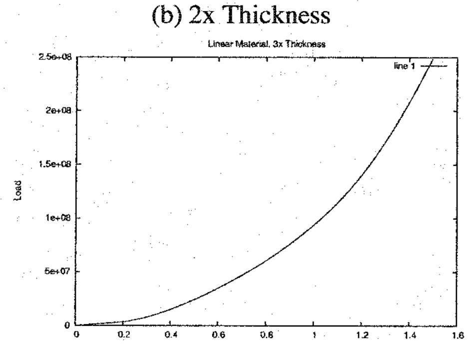
Figure 7.24: Force vs. Deformation at Location of Point Load Application



(a) 1x Thickness



(b) 2x Thickness



(c) 3x Thickness

Figure 7.25: Force vs. Deformation for Various Increases in Thickness

factor of approximately 3 for these simulations. The volume constraint also smooths the behavior of the system; without volume conservation, the model experiences a large initial deformation for a small load, then slowly stiffens. Figure 7.25 shows the effect of increasing thickness on the behavior of the system. The vertical scale for each of the plots is adjusted for the relative maximum displacement. Bending stiffness increases with the cube of thickness. The shape of the load-displacement curve is almost identical for each of the experiments shown in Figure 7.25; the effect of bending energy on the solution is illustrated by the differences in magnitude of applied load in each of these plots.

7.5.2 Evans and Skalak Material Model

Unlike the standard linear material model, the Evans-Skalak material model is dependent on the current local strain everywhere along the body. To implement this material model in our simulation, small load steps were taken. At each step the total strain at each integration point was computed in the local coordinate system and input into the Evans-Skalak energy formula. This formula was numerically differentiated to find relationships for the stress tensor and constitutive tensor of Section 2.3.3. Results for a small amount of deformation are shown in Figure 7.26. This formulation proved problematic for large strains in two ways, both due to the vast differences in magnitudes of the energies associated with stretching, shearing and bending. The lack of appreciable bending energies (due to the very very small bending constant k_{crbc}) caused numerical problems for the linear system solver using double precision floating point numbers. Beyond small deformations, elements tended to "pop" or self intersect due to the relatively low penalty for shearing. Increasing the bending stiffness removed some numerical hurdles and lessened popping; for simulations which are not dominated by bending energies, this may be a useful technique for stabilizing the system. A material law, such as the hyper-elastic formulation of Pozrikidis, might offer increased numerical stability and better correlation to experiment.

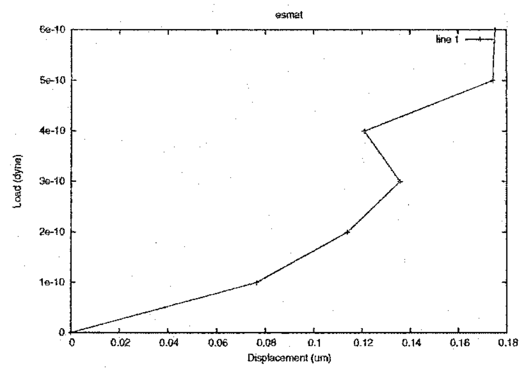


Figure 7.26: Simulation with Evans-Skalak Material Law

7.6 Summary

In this chapter we presented a framework for simulating the mechanics of red blood cells using subdivision-based thin shell finite elements. We have shown these simulations to be robust with respect shape representation and moving constraints for linear elastic materials for two commonly conducted cell experiments. Qualitatively these simulations show great promise for the future. Our simulations show significant emergent behavior in the evolution of the shape during simulation, and are not bound to fully constrained geometries as previous simulations have been. The use of the Evans-Skalak material law has not yet proven fruitful for quantitative simulation because of poor numerical conditioning. The implementation of a more physically based material law, such as the hyper-elastic formulation of Pozrikidis [49], is expected to yield quantitative results.

Chapter 8

DEMONSTRATIONS

8.1 Introduction

This chapter is intended to demonstrate the utility of the multilevel subdivision thin-shell element in other engineering contexts. The use of subdivision elements for buckling analysis is demonstrated in Section 8.2. The compact nature of the element provides a robust framework for buckling analysis; enabling smooth, curved solutions to be represented with a minimum number of control points. Section 8.3 demonstrates dynamic and vibrational analysis using subdivision elements. Lastly a description of the software package created especially for this work is provided in Section 8.4.

8.2 Buckling

Buckling of structural members is a primary failure mode in many systems and is therefore crucial to engineering design analysis. Buckling of a structure is qualitatively defined to be failure in which large displacements occur transverse to the applied load. Quantitatively, buckling occurs in thin members due to interactions between membrane and bending stresses which cause the system to become unstable. Buckling is inherently a non-linear phenomenon which cannot be predicted by standard linear analysis of plates and shells. "Linearized Buckling" is a first order estimate of the nonlinear stresses in a structure which enables estimates of buckling loads and deformed shapes to be predicted.

8.2.1 Initial Stability Analysis

A continuous system is stable equilibrium if the virtual work functional, Equation 2.16 is zero and its first variation is positive for all possible admissible displacements $\mathbf{u}$. For discretized systems this criteria is equivalent to the tangent stiffness matrix $\mathbf{K}_t$ being positive definite. Let λ_i be the eigenvalues of $\mathbf{K}_t$. If

$$\lambda_i > 0 \forall \quad i \quad 1..n \tag{8.1}$$

then $\mathbf{K}_t$ is positive definite. If $\mathbf{K}_t$ has any negative eigenvalues then there displacement exist for which resistance to deformation decreases with applied load and the body will undergo large displacements. Assuming that $\mathbf{K}_t$ is positive definite in an initial configuration of the body then the smallest displacement for which $det(\mathbf{K}_t) = 0$ will indicate a bifurcation point at the onset of buckling. One way to determine the buckling point for a given loading condition is to slowly increment to the desired loading condition in small steps checking for instabilities along the way. This approach is time consuming and error prone if the load steps are not scaled appropriately. Initial stability analysis provides an approximate solution to the buckling load.

For situations in which deformations are small, but stresses are large the tangent stiffness matrix may be approximated as

$$\mathbf{K}_t \approx \mathbf{K}_0 + \boldsymbol{\sigma} \cdot \mathbf{K}_\sigma, \tag{8.2}$$

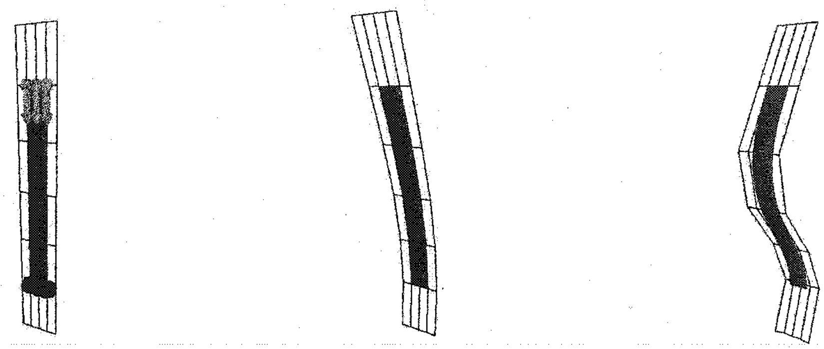
where $\mathbf{K}_0$ is the initial tangent stiffness matrix, and $\mathbf{K}_\sigma$ is the portion of the stiffness matrix which is proportional to applied load. $\mathbf{K}_\sigma$ is found by isolating the components of the nonlinear tangent stiffness matrix $\mathbf{K}_t$ proportional to stress $\boldsymbol{\sigma}$. Initially these components will be all scaled by zero. After solving $\mathbf{K}\mathbf{u} = \mathbf{f}$, and finding $\boldsymbol{\sigma}(\mathbf{u})$, this value can be substituted back in and $\mathbf{K}_\sigma$ can be found for a specific loading.

Once $\mathbf{K}_\sigma$ is known, finding critical loads is reduced to the generalized eigenvalue problem of $det(\mathbf{K}_0 + \boldsymbol{\sigma} \cdot \mathbf{K}_\sigma) = 0$. The eigenvalues $\boldsymbol{\sigma}$ will be the critical loads; a multiple of the applied loading at which buckling will occur. The smallest among these eigenvalues will be the value of the first load at which buckling occurs (to within the linearized estimate). The eigenvectors of this problem will be the shape which the body assumes just prior to the onset of buckling.

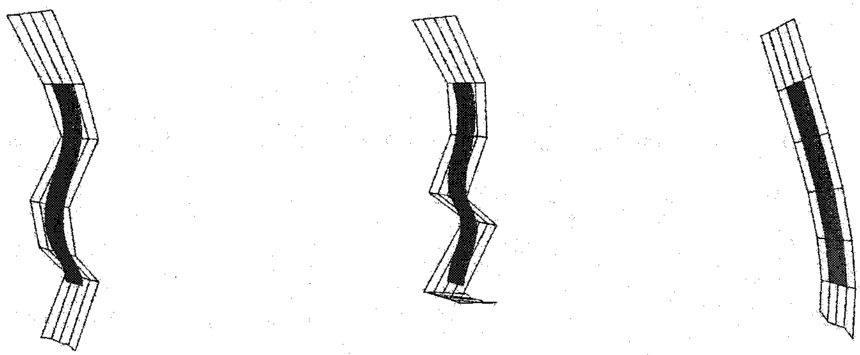
8.2.2 Prismatic Shell Buckling

Buckling analysis of a prismatic shell model is a useful test of how well thin shell simulation compares in accuracy to plate and beam simulation for rectangular bodies. The dimensions of the body are such that it is one tenth as wide as it is long and one tenth again as thick as it is wide. The thickness direction of the shell corresponds to the smallest dimension of the body. Due to the geometry of the body, the major moment of inertia is 100 times larger than the minor moment of inertia. Poisson's ratio, which represents coupling between orthogonal strains, was set to zero to simulate the constitutive behavior of a of a beam.

Figure 8.1 shows the first few buckling mode shape of the prismatic shell. The loading of the model is shown in Figure 8.1(a). A line load (illustrated as arrows) is applied to the top row of vertices acting downward along the length of the body. Clamped boundary constraints, illustrated by the line of dark spheres, are applied to the bottom of the body to prevent both translation and rotation. These boundary conditions are discussed in Section 5.2.5. The fundamental buckling mode of the body is shown in Figure 8.1(b), and represents (to the accuracy of the simulation) one quarter of a cosine wave. Other higher order buckling modes are also shown. Of great interest is the sixth buckling mode of the system, which corresponds to in plane buckling about the minor axis of the body. The fundamental in and out of plane buckling modes modes are illustrated for clarity in Figure 8.2 from an orthogonal perspective to show their behavior more clearly. The first buckling mode is an out of plane mode with bending occurring through the thickness direction. The sixth mode shows similar in plane behavior, with "bending" through the width direction. Because in plane bending modes are not explicitly represented in the mechanics of thin shells, this behavior is aggregate and it's accuracy is limited by the ability of the subdivision basis functions to represent in plane curvature deformations.



(a) Loading (b) 1st Buckling Mode (c) 2nd Buckling Mode



(d) 3rd Buckling Mode (e) 4th Buckling Mode (f) 6th Buckling Mode

Figure 8.1: Buckling Modes of a Prismatic Plate

(a) 1st Buckling Mode (b) 6th Buckling Mode

Figure 8.2: Fundamental Buckling Modes of a Prismatic Plate

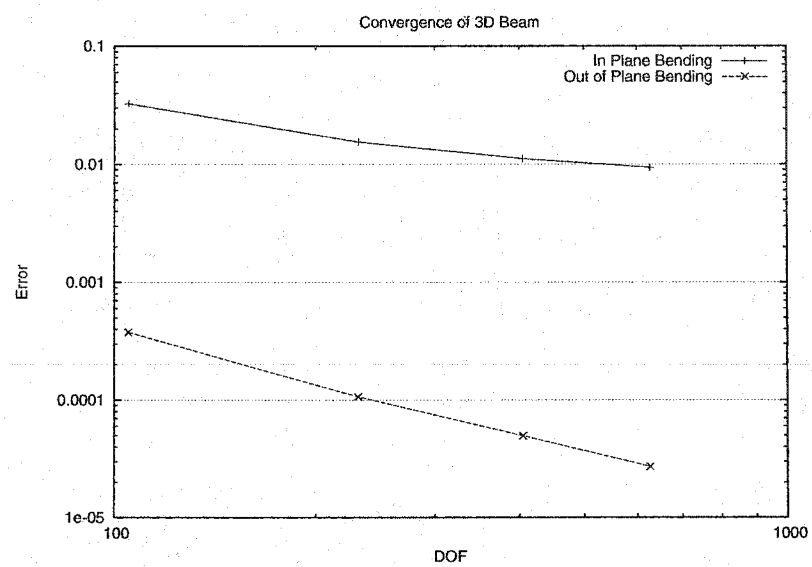


Figure 8.3: Convergence of Buckling Modes of a Prismatic Plate

The order of accuracy of these results can be compared to those computed for beams using the Euler linear buckling relation [19]. Figure 8.3 shows the convergence of the value of the computed critical loads to Euler's result versus the number of degrees of freedom. Results are shown for both the fundamental out of plane buckling mode and the in plane bending mode. The value of the critical load for the out of plane buckling mode converges with quadratic accuracy to Euler's value as the number of degrees of freedom of the system is increased. The critical load for the in plane bending mode converges at a substantially lower rate. This is to be expected as in plane curvatures are not explicitly represented in this formulation.

8.2.3 *Cylindrical Shell Buckling*

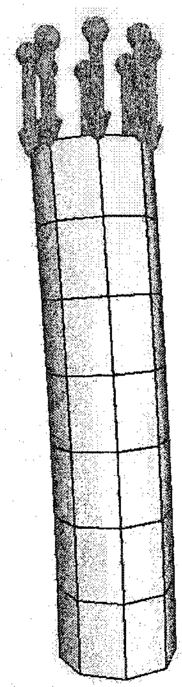
This section demonstrates the buckling deformation of a non-trivial 3D object: a cylindrical tube. Although this model possesses radial symmetry, the solution to the equations of motion governing the buckled shape are not radially symmetric, exhibiting the need for a full three-dimensional analysis. The model for this problem is the buckling of an upright cylinder under downward vertical load, with a simply supported bottom. The two fundamental buckled shapes for this problem are illustrated in Figure 8.4. The Euler buckling mode, in which the cylinder behaves as a long beam, is shown in Figure 8.4(a). The more classic and complicated buckling mode for a thin cylinder is demonstrated in Figure 8.4(b). This buckling mode is usually referred to as "chessboard" buckling (or "checkerboard" depending on your choice of game) and characterized by sinusoidal ripples alternately out of phase with one another. Which of these two buckled shapes occur depends on the geometry of the cylinder. If it is long and thin it will buckle like a beam, otherwise it will assume the chessboard shape; unless it is too short and squat it will buckle like a plate. The buckling of thin shells is well summarized in [37] and a variety of beautiful buckling experiments are shown in [56, 57].

The accuracy of the subdivision shell element in buckling is shown in Figure 8.5. Here the error in the critical load is plotted versus increasing refinement on a log-log scale. Second order convergence in the number of elements is clearly seen, testifying to the accuracy of both the subdivision shell element of Chapter 4 and the boundary formuation of Chapter 5.

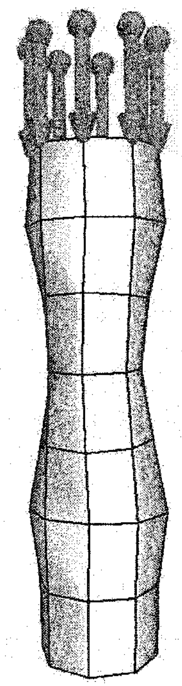
8.3 Dynamics and Vibrations

8.3.1 Linear Free Vibrations

Subdivision elements are natural candidates for vibration analysis of thin shells. Free, undamped vibrations are those modes of vibration which persist in the absence of applied loads. Each mode is described by a mode shape and an associated frequency. The mode shape for a discretized model will be a vector of generalized displacements. The derivation for free undamped vibrations is the same as for other finite element approaches. A good introduction to vibration analysis in finite element applications is given by Zienkiewicz [65]. The analysis begins with a law of motion in terms of the mass matrix M, the stiffness



(a) Euler Buckling Mode



(b) Chessboard Buckling Mode

Figure 8.4: Buckling of a Cylindrical Tube

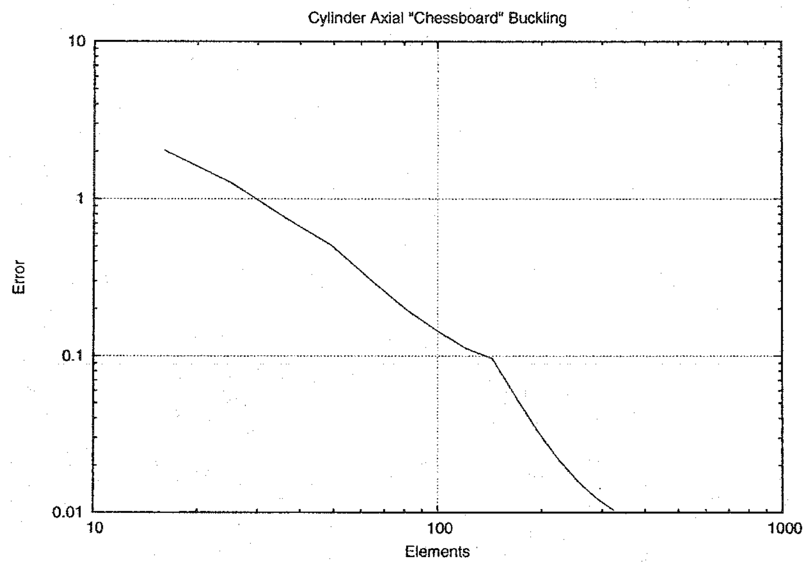


Figure 8.5: Convergence of Chessboard Buckling Mode of a Cylindrical Beam

matrix $\mathbf{K}$ and a vector of generalized displacements $\mathbf{v}$

$$\mathbf{M}\ddot{\mathbf{v}} + \mathbf{K}\mathbf{v} = \mathbf{0}. \tag{8.3}$$

The forcing term in the above relation has been identically set to zero, as we are interested in finding modes that require no external excitation. The generalized displacements of the system are assumed to vary sinusoidally in time, and are written as

$$\mathbf{v} = \hat{\mathbf{v}}e^{i\omega t} \tag{8.4}$$

for some vector of generalized displacements $\hat{\mathbf{v}}$. Equations 8.3 and 8.4 may be combined, with the result:

$$\mathbf{K}\hat{\mathbf{v}}e^{i\omega t} = \omega^2\mathbf{M}\hat{\mathbf{v}}e^{i\omega t}. \tag{8.5}$$

Lastly, dividing both sides of Equation 8.5 by $e^{i\omega t}$ yields the undamped linear free vibration equation of motion. Equation 8.6.

$$\mathbf{K}\hat{\mathbf{v}} = \omega^2\mathbf{M}\hat{\mathbf{v}}. \tag{8.6}$$

The form of Equation 8.6 of motion shows that the free vibrational modes of the system $\hat{\mathbf{v}}$ and their corresponding natural frequencies ω^2 may be interpreted as the generalized eigenvectors and eigenvalues (respectively) of the mass and stiffness matrices $\mathbf{M}$ and $\mathbf{K}$.

Constrained Systems

Most physical systems are constrained in some way by external reactions; constraints for subdivision shell elements are discussed in Chapter 5. For linear vibration systems, this is equivalent to solving Equation 8.6 subject to the restriction

$$\mathbf{C}\hat{\mathbf{v}} = \mathbf{0}, \tag{8.7}$$

where $\mathbf{C}$ is the matrix of constraint derivatives described in Section 6.6. One method of solving Equation 8.6 subject to the restriction of Equation 8.7 is to project the generalized displacements $\hat{\mathbf{v}}$ into the null space of $\mathbf{C}$ by letting

$$\hat{\mathbf{v}} = \mathbf{Z}\mathbf{p}, \tag{8.8}$$

where $\mathbf{Z}$ is the null space of $\mathbf{C}$ and $\mathbf{p}$ is a vector which represents the projection of $\hat{\mathbf{v}}$ onto the constraint. The constrained form of Equation 8.6 is given below as Equation 8.9.

$$\left(\mathbf{Z}^T\mathbf{K}\mathbf{Z}\right)\mathbf{p} = \omega^2\left(\mathbf{Z}^T\mathbf{M}\mathbf{Z}\right)\mathbf{p}. \tag{8.9}$$

The natural frequencies ω of the constrained system may be read directly from the above equation. The constrained free modes of vibration may be obtained by solving the eigensystem of Equation 8.9 and applying Equation 8.8 to find the generalized displacements.

All results shown in this work are calculated using LAPACK's [1] the DSYGV and DGESVD commands to compute solutions to the generalized eigensystem and null space problems. A sparse constrained eigensolver was not used, though the implementation of a sparse, multi-level preconditioned solver using framework of Chapter 6 remains an open area for further study.

Validation

A simply supported square plate model is ideal for validation because analytical solutions for the natural frequencies are readily available published literature [62]. The subdivision solution framework was found to provide excellent accuracy for models with very few elements. Figure 8.6 shows an example diagram of a flat plate with uniform loading condi-

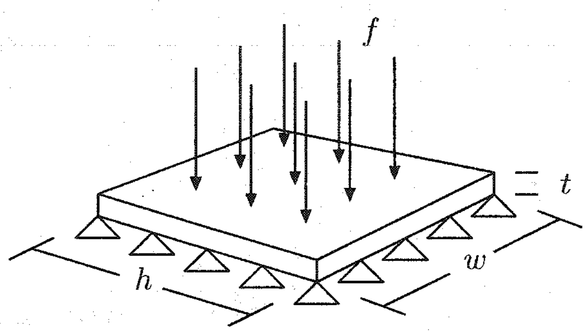


Figure 8.6: Simply Supported Plate with Uniform Loading

tions. All four edges of the plate are simply supported (positions are fixed and rotations are free). The dimensions of the example tested were as follows: width/height $w = h = 1m$, thickness $t = 0.01m$, elastic modulus $E = 100GPa$, Poisson's ration $\nu = 0.3$, applied load f $= 0$.

Figure 8.7 shows the error in the calculated lowest mode natural frequency versus discretization. Error shown is defined as the relative error in the frequency versus an analytical solution. The leftmost data point of Figure 8.7 corresponds to a model with only four elements and a total of 75 degrees of freedom, 1/3rd of which (25) represent out of plane motion, and 2/3rds of which (50) represent in plane displacements. The rightmost data point represents a model with 196 elements and 867 degrees of freedom.

8.3.2 *Dynamic Sphere*

This example shows a thin spherical body being tugged on by two opposing surface loads acting on opposite hemispheres. The rest state of the body is a spherical shape. Over

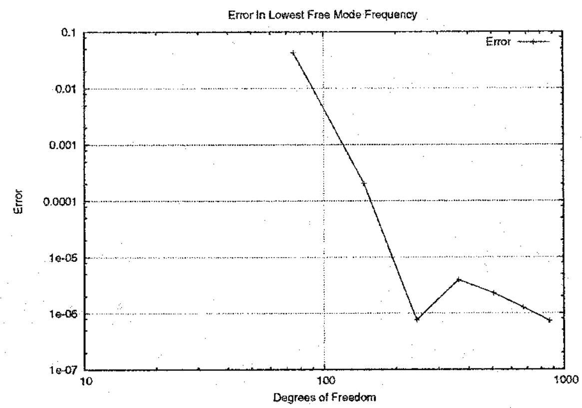


Figure 8.7: Error in Natural Frequency vs. Discretization

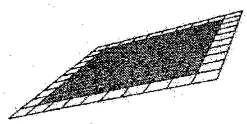
time the sphere is deformed into a football-like shape by the applied loads, until the elastic energy of the body returns it to a spherical shape and the process begins anew. Ordinary elements are colored blue, and extraordinary elements are colored green. A sequence of timesteps is shown in Figure 8.11.

8.4 Solution Environment

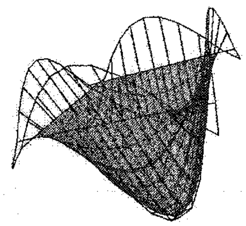
This section provides a short description of the application environment created to conduct experiments involving subdivision thin shell simulations.

8.4.1 Interactive Application

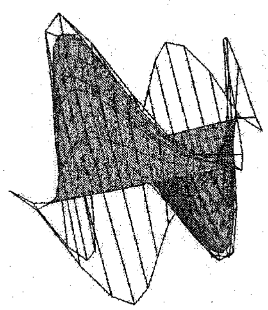
We have developed an interactive application to test and analyze the theoretical framework described in the previous chapters. An interactive application with visualization capabilities is necessary to understand (and debug) the subdivision element. Figures 8.14(a) and 8.14(b) show a parent control mesh and the same mesh subdivided once respectively. The highlighted directed edge indicates the parent edge in the control mesh and its primary child in the subdivided version. Ordinary faces are colored blue (dark) and extraordinary faces are colored green (light).



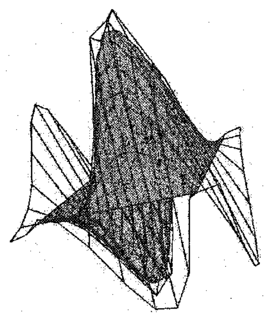
(a) Rest State



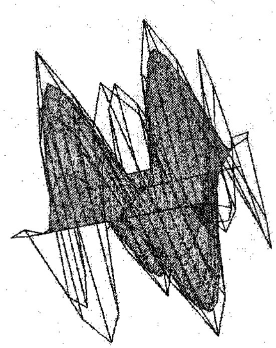
(b) Mode 1



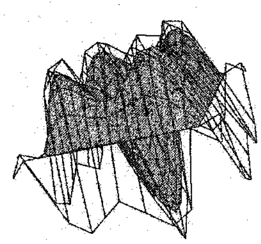
(c) Mode 2



(d) Mode 3

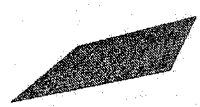


(e) Mode 4

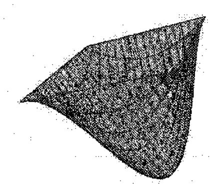


(f) Mode 5

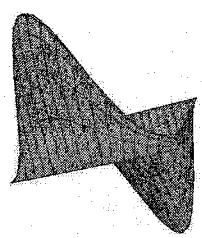
Figure 8.8: Square Plate, 64 Elements, Limit Surface and Control Mesh



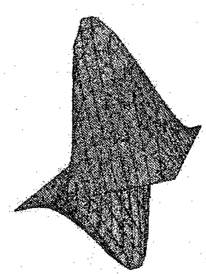
(a) Rest State



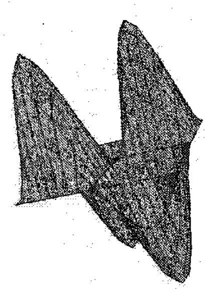
(b) Mode 1



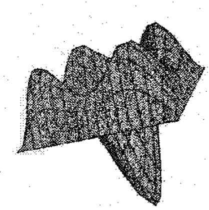
(c) Mode 2



(d) Mode 3

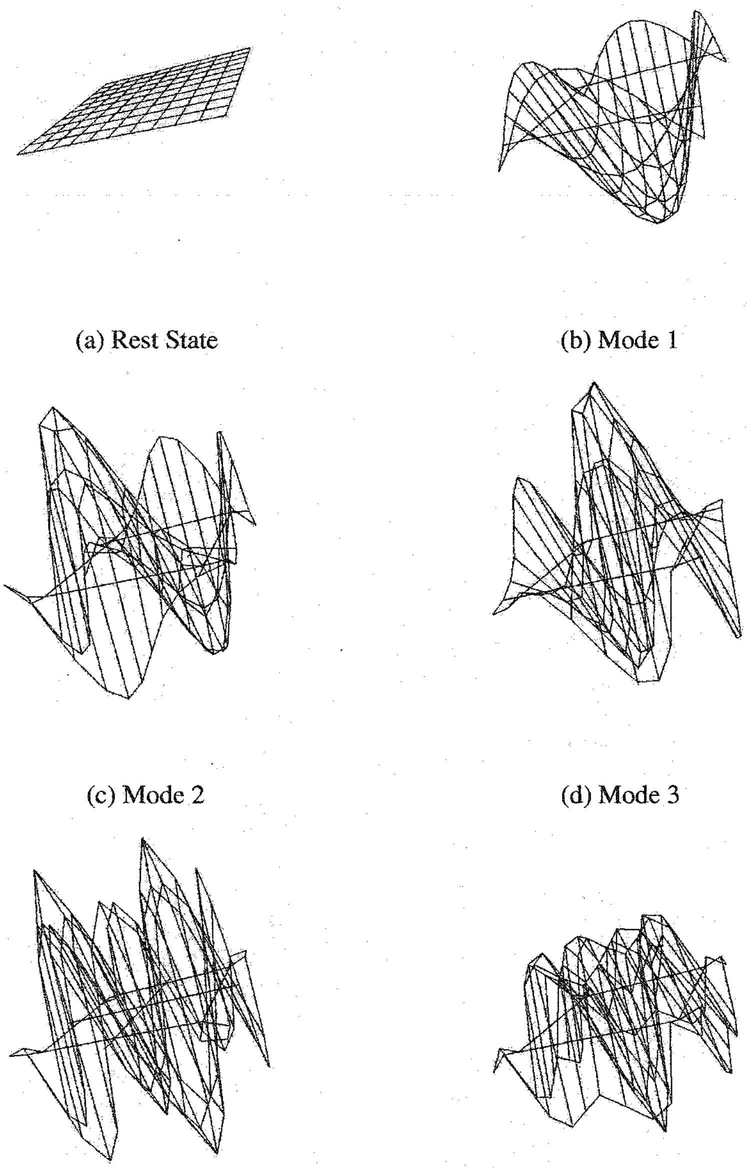


(e) Mode 4



(f) Mode 5

Figure 8.9: Square Plate, 64 Elements, Limit Surface



(a) Rest State

(b) Mode 1

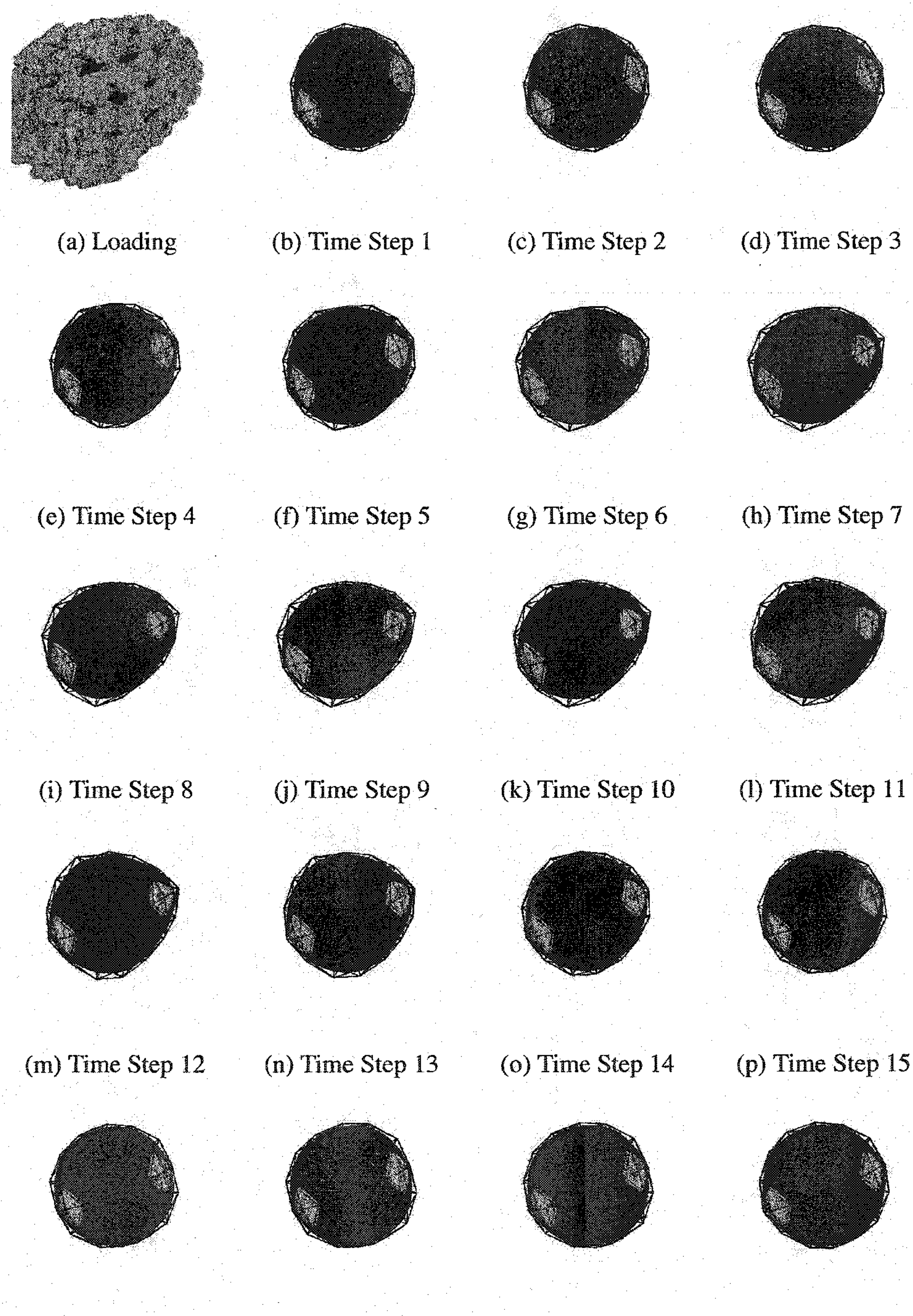
(c) Mode 2

(d) Mode 3

(e) Mode 4

(f) Mode 5

Figure 8.10: Square Plate, 64 Elements, Control Mesh



(a) Loading (b) Time Step 1 (c) Time Step 2 (d) Time Step 3

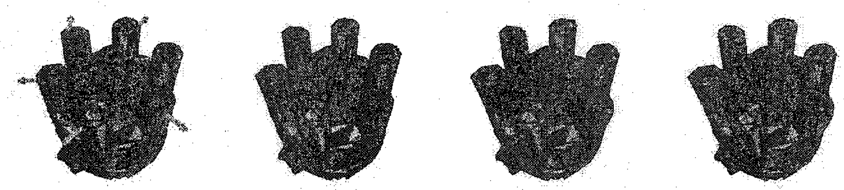
(e) Time Step 4 (f) Time Step 5 (g) Time Step 6 (h) Time Step 7

(i) Time Step 8 (j) Time Step 9 (k) Time Step 10 (l) Time Step 11

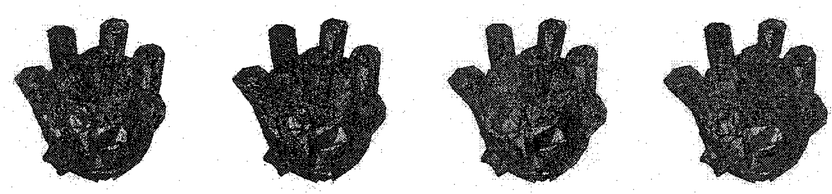
(m) Time Step 12 (n) Time Step 13 (o) Time Step 14 (p) Time Step 15

(q) Time Step 16 (r) Time Step 17 (s) Time Step 18 (t) Time Step 19

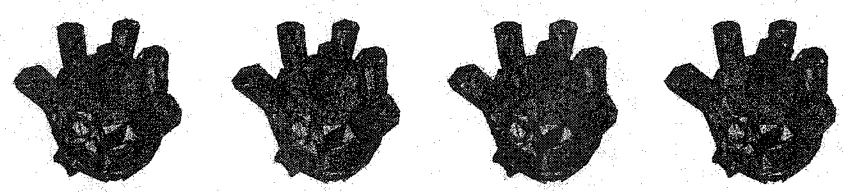
Figure 8.11: Dynamic Simulation of a Thin Spherical Body



(a) Loading (b) Time Step 1 (c) Time Step 2 (d) Time Step 3

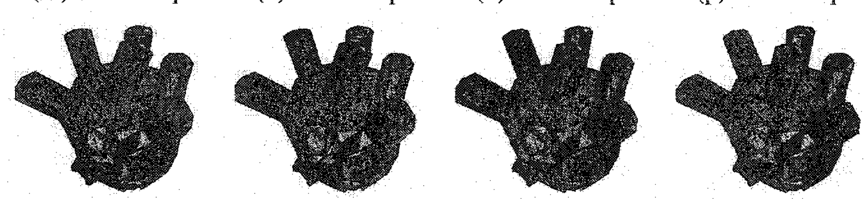


(e) Time Step 4 (f) Time Step 5 (g) Time Step 6 (h) Time Step 7



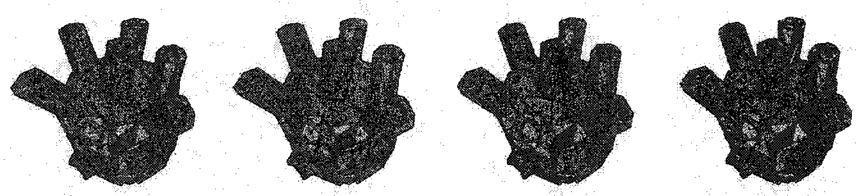
(i) Time Step 8 (j) Time Step 9 (k) Time Step 10 (l) Time Step 11

(m) Time Step 12 (n) Time Step 13 (o) Time Step 14 (p) Time Step 15

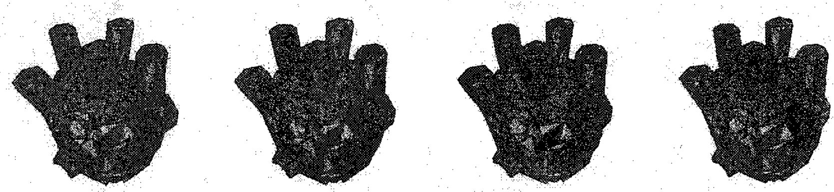


(q) Time Step 16 (r) Time Step 17 (s) Time Step 18 (t) Time Step 19

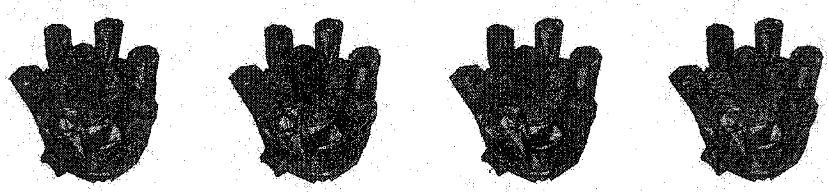
Figure 8.12: Dynamic Simulation of a Distributor Cap



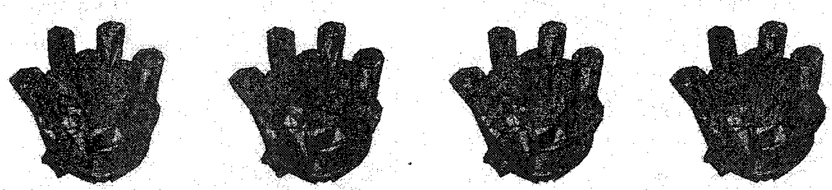
(a) Loading (b) Time Step 20 (c) Time Step 21 (d) Time Step 22



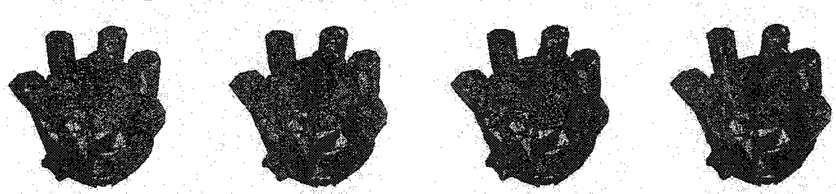
(e) Time Step 23 (f) Time Step 24 (g) Time Step 25 (h) Time Step 26



(i) Time Step 27 (j) Time Step 28 (k) Time Step 29 (l) Time Step 30

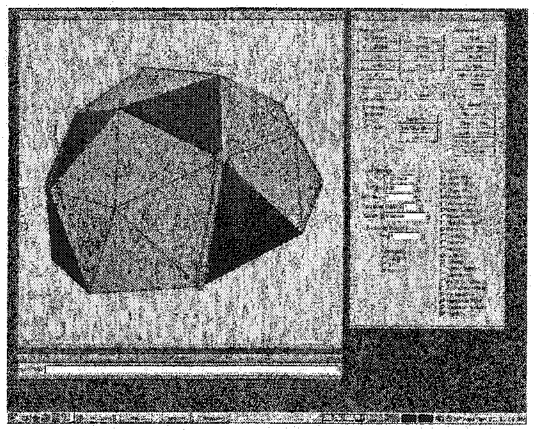


(m) Time Step 31 (n) Time Step 32 (o) Time Step 33 (p) Time Step 34

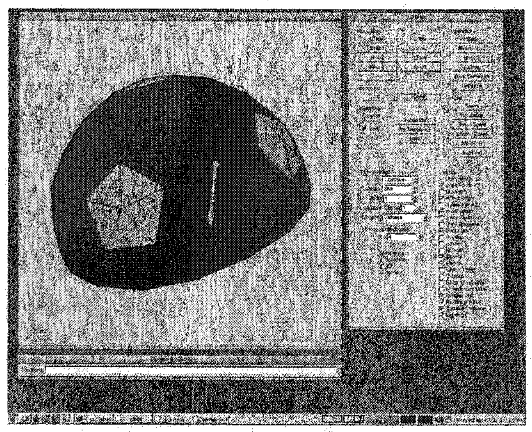


(q) Time Step 35 (r) Time Step 36 (s) Time Step 37 (t) Time Step 38

Figure 8.13: Dynamic Simulation of a Distributor Cap



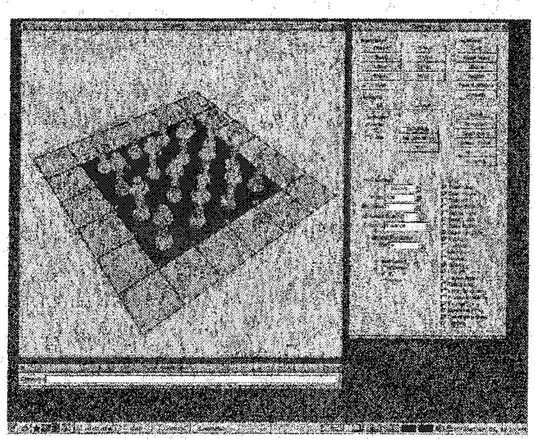
(a) Coarse Control Mesh and Parent Edge



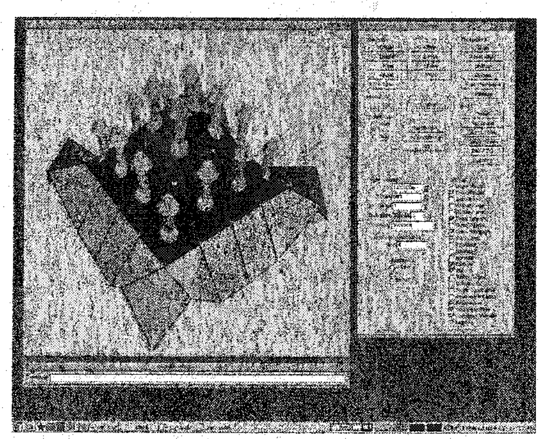
(b) Once Subdivided Mesh and Child Edge

Figure 8.14: Discretized Thin Shell Models

Figures 8.15(a) and 8.15(b) show a simulation of a simple flat plate model under uniform vertical load with simply supported boundaries in the reference and deformed configurations respectively. Each of the elements and vertices are numbered for reference. Loads are indicated by the green arrows. In both configurations the boundary ghost faces are illustrated. Although they deform and their vertices contribute to the geometry of the model, no geometry is associated with the ghost faces themselves.



(a) Undeformed Plate Model



(b) Deformed Plate Model

Figure 8.15: Plate With Simply Supported Boundaries

This application allows real time navigation and inspection of each face of each level of a subidivision hierarchy using the extended Quad-Edge data structure described in Section 4.11. Every vertex, edge, and face may be queried for applied loading and boundary conditions.

8.4.2 Source Code Interface

The test implementation of this project has been completely implemented in the *C++* language. At the heart of the code base is the `SubdivisionEvaluator` class shown in Figure 8.16. This class provides a base interface for evaluating the subdivision basis functions and their derivatives. The Loop and Catmull-Clark evaluators inherit the interface of this class and provide a specific implementation. Central to this class are a pair of complementary functions `Evaluate` and `ControlVertices` which allow the basis functions which describe the model, their derivatives and control weights to be evaluated at any parametric location on the surface.

- `ControlVertices( Edge )` returns a vector of control vertices for the neighborhood of the face which contains the given directed edge. The number of vertices returned is dependent on the local topology of the face. The control vertices are ordered in a manner consistent with the `Basis` function.

- `Basis( Edge, double u, double v, int du, int dv )` returns a vector of the evaluated basis functions for a face containing the given directed edge. The basis functions are numerically evaluated at the parametric location u, v, taking du derivatives in the u direction and dv derivatives in the v direction (either du, or dv or both may be zero to take no derivatives). The inner product of the vector of basis functions and vector of control vertices evaluates the surface (or its derivatives) at the chosen parametric location.

- `LimitVertices( Edge )` returns the vector of control mesh vertices for eval-

```
class DMat; // Double precision matrix class
class TMat; // Templated matrix class

// Base class for evaluating subdivision surfaces analytically
class SubdivisionEvaluator
{

public:

// Return the control vertices influencing a patch
virtual TMat< Vertex * > ControlVertices( Edge * ) const = 0;

// Return the basis functions influencing a patch evaluated
// at a parametric location u,v.
// du,dv are the number of derivatives in the u,v directions
// [v,w for trangular patches] respectively.
//
// Basis( some_edge, 0.3, 0.6, 2, 1 ) would give:
//      d^3              |
//   ---------- Surface |
//   d^2 u  d v          | ( u=0.3, v=0.6 )
//
// du,dv may be 0 if no derivatives are to be taken.
virtual DMat Basis( Edge *, double u, double v,
                        int du, int dv ) const = 0;

// Limit weights for origin of an edge
virtual TMat< Vertex * > LimitVertices( Edge * ) const = 0;
DMat LimitMask( Edge * ) const;
DMat TangentMask( Edge *, int which ) const;

enum FACETYPE { IMPROPER, ORDINARY, EXTRAORDINARY, EXTRAEXTRAORDINARY };

// Get info about a face
virtual FACETYPE FaceType( Edge * ) const = 0;

// Returns the appropriate quadrature points
virtual DMat QuadraturePoints( int order ) const = 0;

// Returns a Jacobian integration factor
virtual double AreaFactor() const = 0;
};

class LoopEvaluator : public SubdivisionEvaluator { ... };
class CCEvaluator   : public SubdivisionEvaluator { ... };
```

Figure 8.16: SubdivisionEvaluator Class Interface

uating the limit position of the origin of the given edge. The number of control mesh vertices returned is proportional to the valence of the origin of the given edge. This function is similar to `ControlVertices`, but is specialized for working with control vertices. The basis functions cease to formally exist at extraordinary control vertices, and limit masks are used to evaluate the surface at these points.

- `LimitMask( Edge )` returns the vector of limit weights as illustrated in Figure 2.4. The inner product of the vector of limit weights and the limit vertices gives the limit position of the desired point.

- `TangentMask( Edge, int )` Returns the vector of weights to calculate each of the limit tangents at the vertex location. The inner product of the vector of tangent weights and the limit vertices gives the limit tangent at the desired point.

- `FaceType( Edge )` returns the type of the face containing the given directed edge. The type of the face corresponds to the element classification scheme presented in Section 4.7. One extra element type is included here, the `INVALID` type. This type corresponds to faces that are topologically or geometrically not possible to evaluate, for instance degenerate faces or elements that do not possess a full neighborhood.

- `QuadraturePoints( int order )` returns the evaluation points and weights for a quadrature scheme of desired order. In our implementation we generally use Gauss-Legendre quadrature of third order [8].

- `AreaFactor()` returns the scaling factor for the Jacobian of integration. For quadrilateral control meshes, this factor is unity. For triangular meshes it is $\frac{1}{2}$.

The `SubdivisionEvaluator` class provides an effective abstract interface for the evaluation of subdivision thin shell finite elements as described in Chapter 4. This interface is inherited by the `CCEvaluator` and `LoopEvaluator` classes which provide an implementation for their respective subdivision schemes. This interface is generic enough that its use need not be restricted to subdivision based finite element schemes. Furthermore this interface could be used by traditional finite element algorithms directly to implement subdivsion finite element schemes without the rest of the code base being aware that subdivsion techniques were being applied.

8.5 Summary

This chapter has presented an overview of some of the capabilities of the multilevel subdivision thin-shell element. We have demonstrated the use of subdivision elements and subdivision boundary conditions in buckling analysis and demonstrated favorable convergence rates. We have also illustrated the use of the subdivision element and associated boundary conditions in linearized dynamic analysis. Lastly in this chapter we have summarized the application framework we have created for simulation.

Chapter 9

CONCLUSION

9.1 Summary

The objective of this study is to characterize and improve the performance of the subdivision thin shell element for practical applications in engineering design and analysis. Subdivision surfaces are increasingly used to represent geometric shapes for engineering design and simulation. Previous work has shown subdivision surface elements to possess the necessary and sufficient properties to represent thin bodies as described by the Kirchhoff-Love equations of motion for thin shell boundary value problems [17] [16]. The subdivision surface element framework provides an elegant and unified paradigm for representing the geometry, physics of thin shells; however, there many unanswered questions remained about these elements. Specifically:

- No consistent framework for evaluating and traversing subdivision element hierarchies existed. Subdivision modeling represents a sharp break from traditional finite element approaches; a concise and well-reasoned approach to evaluating, traversing, and implementing these models is needed.

- The solution of simple standard boundary value problems converged slowly to known solutions. For all the geometric elegance these elements possess, simulations involving them converged at rates no greater than other less powerful approximations.

- The computational effort involved in solving subdivision element boundary value problems directly scaled poorly with the number of unknowns. This problem is exacerbated in thin shell simulations due to the high order derivatives needed to compute curvatures.

- No applications of this element have yet been studied in the bio-sciences. The robust ability of these elements to handle large, smooth deformations makes them natural candidates for simulating thin biological membranes.

9.2 Contributions

The major contributions of this study have been the development of a unified formulation for the evaluation and traversal of subdivision meshes, accurate formulation of engineering boundary conditions, creation of an efficient multilevel solution strategy and application of our work to a new and important problem: the simulation of blood cell membranes in the human body. These contributions are summarized as follows:

- We have developed a unified framework for both quadrilateral and triangular subdivision models, along with powerful data structures and algorithms for evaluating and traversing models within a multilevel hierarchy in Chapter 4. The derivation of the mechanics of thin shells and the equations of equilibrium have been shown for the subdivision shell element, and for the subdivision beam element in Chapter 3. We have classified elements into three distinct types corresponding to the differing evaluation techniques needed and shown how the element may be evaluated in each of these situations. Our formulation allows either quadrilateral or triangular base meshes to be used without additional implementation effort.

- We have presented a new method for enforcing boundary conditions within subdivision surface finite element simulations of thin shells in Chapter 5. The proposed formulation is shown to be second order accurate for displacements with respect to increasing refinement for simply-supported and clamped boundary conditions. Second order accuracy on the boundary is consistent with the accuracy of subdivision based approaches for the interior of a body. Our proposed formulation is applicable to both triangular and quadrilateral refinement schemes, and does not impose any topological requirements upon the underlying subdivision control mesh. Several examples from the Belytschko *et. al.* [9] obstacle course of benchmark problems are used to demonstrate the convergence of the scheme.

- We have described an algorithm that exploits the hierarchical structure of subdivision surfaces to solve boundary value problems in demonstrated near $O(N)$ time in Chapter 6. It is shown that the subdivision framework can be used not only for representing the geometry of the solid and the mechanics of the simulation, but also for accelerating the numerical solution. Specifically we have illustrated the use of the subdivision matrix as an effective information transfer operator in a multilevel preconditioned conjugate gradient solution scheme. Our method allows us to construct practical simulations that are effective on a broad range of problems. Subdivision surface basis functions now form a triumvirate of representation techniques for thin shell simulation: they serve to define the model's geometry, deformation and hierarchical description.

- Demonstration of the utility of our work by applying these concepts towards modeling the membrane of blood cells in the human blood stream. We have shown these simulations to be robust with respect shape representation and moving constraints for linear elastic materials for two well known experimental frameworks.

These contributions have increased the utility of subdivision modeling and simulation of thin shells for engineering design and analysis. We have organized and clarified the implementation of these elements, improved their accuracy, increased their solution speed and demonstrated their utility in a novel problem framework.

9.3 Future Work

This investigation has brought to light several new areas for future work. Combining the efficient solution tactics presented in this document with adaptive representation techniques (such as those presented in [41] and [33]) would combine both numerical and spatial efficiency and could reduce solution times even further. A formal proof for the convergence of the boundary conditions presented in this document would establish their accuracy rigorously and guarantee convergence.

However, most important future work on this topic lies in the area of application, especially to the bio-sciences. This work has presented the building blocks for representing cell membranes as a part of a full blood flow solution scheme. Combining these efficient solution techniques and representation methods with a flow solver will create a powerful tool to be used for the design of artificial organs. Deformable thin structures appear in many other biological contexts including vascular walls and skin, and in bio-engineered devices such as stent coatings, which undergo considerable deformation during deployment. Often bending analyses of these bodies have been ignored due to the perceived difficulty of computing bending deformations and the lack of appropriate representation schemes for large deformations. A complementary area for future work is the creation of new material models which inherently incorporate large bending energies in their definition in areas where bending deformations have previously been ignored.

BIBLIOGRAPHY

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide, second edition*. SIAM Press, 1995.

[2] James F. Antaki, Guy E. Blelloch, Omar Ghattas, Ivan Malcevic, Gary L. Miller, , and Noel J. Walkington. A parallel dynamic–mesh lagrangian method for simulation of flows with dynamic interfaces. In *Proceedings of Supercomputing 2000*, Dallas, Texas, USA, November 4-10 2000.

[3] G. Arden. *Approximation Properties of Subdivision Surfaces*. PhD thesis, University of Washington, 2001.

[4] S. Ashby and R. Falgout. A Parallel Multigrid Preconditioned Conjugate Gradient Algorithm for Groundwater Flow Simulations. *Nuclear Science and Engineering*, 124:145–159, 1996.

[5] D. G. Ashwell and R. H. Gallagher. *Finite Elements for Thin Shells and Curved Members*. John Wiley and Sons, 1976.

[6] R. Banton and C. Eggleton. Simulation of red blood cell membrane deformation in an extensional flow. In *Bioengineering Conference*. ASME, 2001.

[7] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[8] K. J. Bathe. *Finite Element Procedures*. Prentice-Hall, Englewood Cliffs, N.J., 1996.

[9] T. Belytschko, H. Stolarski, W.K. Liu, N. Carpenter, and J.S.J. Ong. Stress projection for membrane and shear locking in shell finite-elements. *Computational Methods in Applied Mechanics and Engineering*, 51:221–258, 1985.

[10] Guy E. Blelloch, Omar Ghattas, Gary L. Miller, Noel J. Walkington, James F. Antaki, Bartley P. Griffith, Marina V. Kameneva Robert L. Kormos, William R. Wagner, ZhongJun Wu, and George M. Turkiyyah. Itr/acs: Simulation of flows with dynamic interfaces on multi-teraflop computers. Sangria Project Proposal http://www-2.cs.cmu.edu/ sangria.

[11] S. K. Boey, D. H. Boal, and D. E. Discher. Simulations of the erythrocyte cytoskeleton at large deformation. i. microscopic models. *Biophysical Journal*, 75:1573–1583, 1998.

[12] Braess. *Finite Elements*. Cambridge University Press, Cambridge, UK, 1997.

[13] J. Bramble and J. Pasciak. A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems. *Mathematics of Computation*, 50(181):1–17, 1988.

[14] W. L. Briggs. *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[15] E. Catmull and J. Clark. Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes. *Computer Aided Design*, 10(6):350–355, 1978.

[16] F. Cirak and M. Ortiz. Fully C^1-conforming subdivision elements for finite deformation thin-shell analysis. *International Journal for Numerical Methods in Engineering*, 51(7):813–833, July 2001.

[17] F. Cirak, M. Ortiz, and P. Schröder. Subdivision Surfaces: a New Paradigm for Thin-Shell Finite-Element Analysis. *International Journal for Numerical Methods in Engineering*, 47(12):2039–72, April 2000.

[18] F. Cirak, M. J. Scott, E. Antonsson, M. Ortiz, and P. Schröder. Integrated Modeling, Finite-Element Analysis, and Engineering Design for Thin-Shell Structures using Subdivision Surfaces. *Computer Aided Design*, 34:137,148, February 2002.

[19] R. R. Cook. *Finite Element Modeling For Stress Analysis*. John Wiley and Sons, Inc., 1995.

[20] P. Schröder D. Zorin, editor. *SIGGRAPH: Subdivision Course Notes, CDROM supplement*, 2000.

[21] W. G. Davids and G. M. Turkiyyah. Multigrid Preconditioner for Unstructured Nonlinear 3D FE Models. *Journal of Engineering Mechanics*, 125(2):186–196, February 1999.

[22] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM Journal on Matrix Analysis and Applications*, 20(3):720–755, 1999.

[23] T. DeRose, M. Kass, and Tien Truong. Subdivision Surfaces in Character Animation. In *Computer Graphics (Siggraph 1998 Proceedings)*, pages 85–94, 1998.

[24] D. E. Discher, D. H. Boal, and S. K. Boey. Simulations of the erythrocyte cytoskeleton at large deformation. ii. micropipette aspiration. *Biophysical Journal*, 75:1584–1597, 1998.

[25] D. Doo and M. Sabin. Behavior of Recursive Division Surfaces Near Extraordinary Points. *Computer Aided Design*, 10(6):356–360, 1978.

[26] C. D. Eggleton and A. S. Popel. Large deformation of red blood cell ghosts in a simple shear flow. *Physics of Fluids*, 10(6), 1998.

[27] E. Evans and Y. C. Fung. Improved measurements of erythrocyte geometry. *Microvascular Research*, 4:335–347, 1972.

[28] E. A. Evans and R. Skalak. *Mechanics and Thermodynamics of Biomembranes*. CRC Press, Inc., 1980.

[29] Foley, van Dam, Feiner, and Hughes. *Computer Graphics Principles and Practice*. Addison Wesley, 2 (in C) edition, 1996.

[30] Y. C. Fung. *Biomechanics*. Springer, 2 edition, 1993.

[31] J. M. Gere and S. P. Timoshenko. *Mechanics of Materials*. PWS Publishing Company, Boston, 3 edition, 1990.

[32] S. Green, G. Turkiyyah, and D. Storti. Subdivision-based multilevel methods for the large scale simulation of thin shells. In *Seventh ACM Proceedings on Solid Modeling and Applications*. ACM, 2002.

[33] Eitan Grinspun, Petr Krysl, and Peter Peter Schröder. Charms: a simple framework for adaptive simulation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 281–290. ACM Press, 2002.

[34] L. Guibas and J. Stolfi. Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. *ACM Transactions on Graphics*, 4(2):74–123, 1985.

[35] Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. *Computer Graphics*, 27(Annual Conference Series):35–44, 1993.

[36] K. D. Hjelmstad. *Fundamentals of Structural Mechanics*. Prentice-Hall, Upper Saddle River, NJ, 1997.

[37] N. J. Hoff. The perplexing behavior of thin circular cylindrical shells in axial compression. *Israel Journal of Technology*, 4(1):1–28, February 1966.

[38] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. *Computer Graphics*, 28(Annual Conference Series):295–302, 1994.

[39] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.

[40] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. *Computer Graphics*, 27(Annual Conference Series):19–26, 1993.

[41] Petr Krysl, Eitan Grinspun, and Peter Schröder. Natural hierarchical refinement for finite element methods. http://multires.caltech.edu.

[42] C. Loop. Smooth Subdivision Surfaces Based on Triangles. Master's thesis, University of Utah, 1987.

[43] R. H. MacNeal. Perspective on finite elements for shell analysis. *Finite Elements in Analysis and Design*, 30:175–186, 1998.

[44] L. E. Malvern. *Introduction to the Mechanics of a Continuous Medium*. Prentice-Hall, Englewood Cliffs, NJ, 1969.

[45] J. C. Meza and R. S. Tuminaro. A Multigrid Preconditioner for the Semiconductor Equations. *SIAM J. Sci. Comput.*, 17(1):118–132, January 1996.

[46] N. Mohandas and E. Evans. Mechanical properties of the red cell membrane in relation to molecular structure and genetic defects. *Annual Review of Biophysics and Biomolecular Structure*, 23:787–818, 1994.

[47] E. Oñate and F. Zárate. Rotation-free triangular plate and shell elements. *International Journal for Numerical Methods in Engineering*, 47:557–603, 2000.

[48] I. D. Parsons and J. F. Hall. The Multigrid Method in Solid Mechanics: Part I - Algorithm Description and Behavior. *International Journal for Numerical Methods in Engineering*, 29:719–737, 1990.

[49] C. Pozrikidis. Numerical simulation of flow-induced deformation of red blood cells. *Annals of Biomedical Engineering*, 31:1194–1205, 2003.

[50] G. Lim R. Mukhopadhyay and M. Wortis. Echinocyte shapes: Bending, stretching, and shear determine spicule shape and spacing. *Biophysical Journal*, 82:1756–1772, 2002.

[51] R. P. Rand. Mechanical properties of the red cell membrane ii: Viscoelastic breakdown of the membrane. *Biophysical Journal*, 4:303–316, 1964.

[52] R. P. Rand and A. C. Burton. Mechanical properties of the red cell membrane i: Membrane stiffness and intracellular pressure. *Biophysical Journal*, 4:114–135, 1964.

[53] J. E. Schweitzer. *Analysis and Application of Subdivision Surfaces*. PhD thesis, University of Washington, 1996.

[54] R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. http://www.cs.berkeley.edu/ jrs/.

[55] J. C. Simo and D. D. Fox. On a stress resultant geometrically exact shell model. part i: Formulation and optimal parameterization. *Computer Methods in Applied Mechanics and Engineering*, 72:267–304, 1989.

[56] J. Singer, J. Arbocz, and T. Weller. *Buckling Experiments, Volume 1, Basic Concepts, Columns, Beams and Plates*. Wiley, 1997.

[57] J. Singer, J. Arbocz, and T. Weller. *Buckling Experiments, Volume 2, Shells, Built-up Structures, Composites and Additional Topics*. Wiley, 2002.

[58] J. Stam. Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values. In *Computer Graphics*, pages 395–404. ACM, 1998.

[59] J. Stam. Exact Evaluation of Loop Triangular Subdivision Surfaces at Arbitrary Parameter Values. In *Computer Graphics*. ACM, 1998. CD-ROM Supplement.

[60] Hanryk Stolarski, Ted Belytschko, and Sang-Ho Lee. A review of shell finite elements and corotational theories. *Computational Mechanics Advances*, 2:125–212, 1995.

[61] G. Taubin. Is This A Quadrisected Mesh? In D. C. Anderson and K. Lee, editors, *Sixth ACM Symposium on Solid Modeling and Applications*, pages 261–266. ACM Press, 2001.

[62] S. P. Timoshenko. *Vibration Problems in Engineering*. Quinn and Boden, 2nd edition, 1937.

[63] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, London, UK, 2001.

[64] P. Schröder U. Reif. Curvature integrability of subdivision surfaces. *Advances in Computational Mathematics*, 14(2):157–174, 2001.

[65] O. C. Zienkiewicz and R.L. Taylor. *The Finite Element Method*. Butterworth-Heinemann, 2000.

[66] D. Zorin. *Stationary Subdivision and Multiresolution Surface Representation*. PhD thesis, California Institute of Technology, 1998.

Appendix A

APPENDIX

A.1 *Nonlinear Beam Derivation*

A.1.1 Virtual work

Begin with virtual work and its first variation

$$\delta\Pi = \int_V \delta\boldsymbol{\epsilon} : \boldsymbol{\sigma} - \delta\mathbf{u}\cdot\mathbf{f}\,dV \tag{A.1}$$

$$\delta^2\Pi = \int_V \delta\boldsymbol{\epsilon} : \mathsf{C}\delta\boldsymbol{\epsilon} + \delta^2\boldsymbol{\epsilon} : \boldsymbol{\sigma}\,dV, \tag{A.2}$$

$$\mathsf{C} = \frac{\partial\boldsymbol{\sigma}}{\partial\boldsymbol{\epsilon}}. \tag{A.3}$$

A.1.2 Displacement Formulation

Let

$$\tilde{\mathbf{x}}(\theta^1) = \mathbf{x}(\theta^1) + \mathbf{u}(\theta^1). \tag{A.4}$$

Substituting in to Equation 3.7, we find:

$$\epsilon = \left[\dot{\mathbf{u}}\cdot\mathbf{a}_1 + \frac{1}{2}\dot{\mathbf{u}}\cdot\dot{\mathbf{u}}\right] - \theta^3\left[(\dot{\mathbf{a}}_1 + \ddot{\mathbf{u}})\cdot\frac{(\mathbf{a}_1+\dot{\mathbf{u}})\times\tilde{\mathbf{a}}_2}{|\mathbf{a}_1+\dot{\mathbf{u}}|}\right]. \tag{A.5}$$

The leftmost bracketed term in the surface or membrane strains. The rightmost bracketed term defines bending. These terms may be treated separately; the choice of $\mathbf{x}$ as the "middle" line eliminates any cross terms.

A.1.3 Membrane

$$\epsilon = \dot{\mathbf{u}} \cdot \mathbf{a}_1 + \frac{1}{2}\dot{\mathbf{u}} \cdot \dot{\mathbf{u}} \tag{A.6}$$

$$\delta\epsilon = \delta\dot{\mathbf{u}} \cdot (\mathbf{a}_1 + \dot{\mathbf{u}}) = \delta\dot{\mathbf{u}} \cdot \tilde{\mathbf{a}}_1 \tag{A.7}$$

$$\delta^2\epsilon = \delta\dot{\mathbf{u}} \cdot \delta\dot{\mathbf{u}}. \tag{A.8}$$

Compute energy

$$\delta\Pi = \int_V \delta\dot{\mathbf{u}} \cdot \tilde{\mathbf{a}}_1 \boldsymbol{\sigma} - \delta\mathbf{u} \cdot \mathbf{f}\, dV \tag{A.9}$$

$$\delta^2\Pi = \int_V \delta\dot{\mathbf{u}} \cdot \tilde{\mathbf{a}}_1 \mathsf{C} \tilde{\mathbf{a}}_1 \cdot \delta\dot{\mathbf{u}} + \delta\dot{\mathbf{u}} \cdot \delta\dot{\mathbf{u}} \boldsymbol{\sigma}\, dV. \tag{A.10}$$

Restrict to a basis

$$\mathbf{u}(\theta) = N^i(\theta)\mathbf{v}_i \qquad\qquad \delta\mathbf{u} = N^i(\theta)\delta\mathbf{v}_i. \tag{A.11}$$

Solving for equilibrium $\delta\Pi = 0\ \forall \delta\mathbf{u}$ yields a system of nonlinear equations

$$\delta\mathbf{v}_i \cdot \int_V \dot{N}^i \tilde{\mathbf{a}}_1 \mathsf{C}\epsilon - N^i \mathbf{f}\, dV = 0 \quad \forall \delta\mathbf{v}_i. \tag{A.12}$$

A Newton-like solver will use the second variation $\delta^2\Pi$ to solve the system

$$\delta^2\Pi = \delta\mathbf{v}_i K^{ij} \delta\mathbf{v}_j \tag{A.13}$$

$$K^{ij} = \int_V \dot{N}^i \dot{N}^j \left[\tilde{\mathbf{a}}_1 \otimes \mathsf{C}\tilde{\mathbf{a}}_1 + \mathsf{C}\epsilon \mathbf{1}\right] dV. \tag{A.14}$$

A.1.4 Bending

$$\begin{aligned} \epsilon &= -\dot{\tilde{\mathbf{a}}}_1 \cdot \tilde{\mathbf{a}}_3 + \dot{\mathbf{a}}_1 \cdot \mathbf{a}_3 \\ &= -(\dot{\mathbf{a}}_1 + \ddot{\mathbf{u}}) \cdot \frac{(\mathbf{a}_1 + \dot{\mathbf{u}}) \times \tilde{\mathbf{a}}_2}{|\mathbf{a}_1 + \dot{\mathbf{u}}|} + \dot{\mathbf{a}}_1 \cdot \mathbf{a}_3 \end{aligned} \tag{A.15}$$

Let

$$\begin{aligned} f &= \dot{\tilde{\mathbf{a}}}_1 \cdot \tilde{\mathbf{a}}_2 \times \tilde{\mathbf{a}}_1 \\ \delta f &= \delta\ddot{\mathbf{u}} \cdot \tilde{\mathbf{a}}_2 \times \tilde{\mathbf{a}}_1 - \delta\dot{\mathbf{u}} \cdot \tilde{\mathbf{a}}_2 \times \dot{\tilde{\mathbf{a}}}_1 \\ \delta^2 f &= \delta\ddot{\mathbf{u}} \cdot \tilde{\mathbf{a}}_2 \times \delta\dot{\mathbf{u}} - \delta\dot{\mathbf{u}} \cdot \tilde{\mathbf{a}}_2 \times \delta\ddot{\mathbf{u}} \\ &\neq 2\delta\ddot{\mathbf{u}} \cdot \tilde{\mathbf{a}}_2 \times \delta\dot{\mathbf{u}} \end{aligned} \tag{A.16}$$

and

$$g = \frac{1}{|\tilde{\mathbf{a}}_1|} = ((\mathbf{a}_1 + \dot{\mathbf{u}}) \cdot (\mathbf{a}_1 + \dot{\mathbf{u}}))^{-1/2}$$

$$\delta g = -\delta \dot{\mathbf{u}} \cdot \frac{\tilde{\mathbf{a}}_1}{|\tilde{\mathbf{a}}_1|^3} = -\delta \dot{\mathbf{u}} \cdot (\mathbf{a}_1 + \dot{\mathbf{u}})((\mathbf{a}_1 + \dot{\mathbf{u}}) \cdot (\mathbf{a}_1 + \dot{\mathbf{u}}))^{-3/2} \tag{A.17}$$

$$\delta^2 g = -\delta \dot{\mathbf{u}} \cdot \frac{\mathbf{1}}{|\tilde{\mathbf{a}}_1|^3} \delta \dot{\mathbf{u}} + \delta \dot{\mathbf{u}} \cdot \frac{3\tilde{\mathbf{a}}_1 \otimes \tilde{\mathbf{a}}_1}{|\tilde{\mathbf{a}}_1|^5} \delta \dot{\mathbf{u}}$$

then

$$\epsilon = fg + \dot{\mathbf{a}}_1 \cdot \mathbf{a}_3$$

$$\delta\epsilon = \delta f g + \delta g f \tag{A.18}$$

$$\delta^2\epsilon = \delta^2 f g + \delta f \delta g + \delta g \delta f + \delta^2 g f$$

Restrict to a basis

$$\mathbf{u}(\theta) = N^i(\theta)\mathbf{v}_i \qquad \delta\mathbf{u} = N^i(\theta)\delta\mathbf{v}_i \tag{A.19}$$

$$\delta\Pi = (f\delta g + g\delta f) : \boldsymbol{\sigma} - \delta\mathbf{u} \cdot \mathbf{f}\, dV \tag{A.20}$$

A.2 Nonlinear Shell Derivation

A.2.1 Virtual Work

$$\delta\Pi = \int_V \delta\epsilon_{ij}\sigma^{ij} - \delta\mathbf{u} \cdot \mathbf{f}\, dV \tag{A.21}$$

$$\delta^2\Pi = \int_V \delta\epsilon_{ij}\mathsf{C}^{ijkl}\delta\epsilon_{kl} + \delta^2\epsilon_{ij}\sigma^{ij}\, dV \tag{A.22}$$

$$\mathsf{C}^{ijkl} = \frac{\partial\sigma^{ij}}{\partial\epsilon_{kl}}. \tag{A.23}$$

The goal is to solve $\delta\Pi = 0 \,\forall\, \delta\mathbf{u}$. $\delta^2\Pi$ is handy for numerical schemes that need gradient information.

A.2.2 Parameterization

$$\mathbf{r}(\theta^1, \theta^2, \theta^3) = \mathbf{x}(\theta^1, \theta^2) + \theta^3 \mathbf{a}_3 \qquad \tilde{\mathbf{r}}(\theta^1, \theta^2, \theta^3) = \tilde{\mathbf{x}}(\theta^1, \theta^2) + \theta^3 \tilde{\mathbf{a}}_3 \tag{A.24}$$

$$\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{u} \tag{A.25}$$

$$\mathbf{a}_1 = \frac{\partial \mathbf{x}}{\partial \theta^1} \equiv \mathbf{x}_{,1} \qquad \tilde{\mathbf{a}}_1 = \frac{\partial \tilde{\mathbf{x}}}{\partial \theta^1} \equiv \tilde{\mathbf{x}}_{,1} \tag{A.26}$$

$$\mathbf{a}_2 = \frac{\partial \mathbf{x}}{\partial \theta^2} \equiv \mathbf{x}_{,2} \qquad \tilde{\mathbf{a}}_2 = \frac{\partial \tilde{\mathbf{x}}}{\partial \theta^2} \equiv \tilde{\mathbf{x}}_{,2} \tag{A.27}$$

$$\mathbf{a}_3 = \frac{\mathbf{a}_1 \times \mathbf{a}_2}{|\mathbf{a}_1 \times \mathbf{a}_2|} \qquad \tilde{\mathbf{a}}_3 = \frac{\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2}{|\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2|} \tag{A.28}$$

A.2.3 Discretization

$$\mathbf{u}(\theta^1, \theta^2) = \mathbf{v}_m N^m(\theta^1, \theta^2) \tag{A.29}$$

A.2.4 Membrane Strain

By definition

$$\epsilon_{ij} = \frac{1}{2}\left(\tilde{\mathbf{a}}_i \cdot \tilde{\mathbf{a}}_j - \mathbf{a}_i \cdot \mathbf{a}_j\right) \tag{A.30}$$

$$= \frac{1}{2}\left((\mathbf{a}_i + \mathbf{u}_{,i}) \cdot (\mathbf{a}_j + \mathbf{u}_{,j}) - \mathbf{a}_i \cdot \mathbf{a}_j\right) \tag{A.31}$$

$$= \frac{1}{2}\left(\mathbf{u}_{,i} \cdot \mathbf{a}_j + \mathbf{u}_{,j} \cdot \mathbf{a}_i + \mathbf{u}_{,i} \cdot \mathbf{u}_{,j}\right) \tag{A.32}$$

Membrane Strain First Variation

$$\delta\epsilon_{ij} = \frac{1}{2}\left(\delta\mathbf{u}_{,i} \cdot \mathbf{a}_j + \delta\mathbf{u}_{,j} \cdot \mathbf{a}_i + \delta\mathbf{u}_{,i} \cdot \mathbf{u}_{,j} + \delta\mathbf{u}_{,j} \cdot \mathbf{u}_{,i}\right) \tag{A.33}$$

$$= \delta\mathbf{v}_m \cdot \frac{1}{2}\left(N^m_{,i}\left(\mathbf{a}_j + \mathbf{u}_{,j}\right) + N^m_{,j}\left(\mathbf{a}_i + \mathbf{u}_{,i}\right)\right) \tag{A.34}$$

Membrane Second Variation

$$\delta^2\epsilon_{ij} = \frac{1}{2}\left(\delta\mathbf{u}_{,i}\cdot\delta\mathbf{u}_{,j} + \delta\mathbf{u}_{,j}\cdot\delta\mathbf{u}_{,i}\right) \tag{A.35}$$

$$= \delta\mathbf{v}_m\cdot\frac{1}{2}\left(N^m_{,i}\mathbf{1}N^n_{,j} + N^m_{,j}\mathbf{1}N^n_{,i}\right)\delta\mathbf{v}_n \tag{A.36}$$

Discretization

$$\delta\Pi = \delta\mathbf{v}_m\cdot\int_V \frac{1}{2}\left[N^m_{,i}\left(\mathbf{a}_j+\mathbf{u}_{,j}\right) + N^m_{,j}\left(\mathbf{a}_i+\mathbf{u}_{,i}\right)\right]\sigma^{ij} - N^m\mathbf{f}\,dV \tag{A.37}$$

$$\delta^2\Pi = \delta\mathbf{v}_m\cdot\mathbf{K}^{mn}\mathbf{v}_n \tag{A.38}$$

$$\begin{aligned}\mathbf{K}_{mn} = \int_V \frac{1}{4}\left[N^m_{,i}\left(\mathbf{a}_j+\mathbf{u}_{,j}\right) + N^m_{,j}\left(\mathbf{a}_i+\mathbf{u}_{,i}\right)\right]\otimes \\ \mathsf{C}^{ijkl}\left[N^n_{,k}\left(\mathbf{a}_l+\mathbf{u}_{,l}\right) + N^n_{,l}\left(\mathbf{a}_k+\mathbf{u}_{,k}\right)\right] \\ +\frac{1}{2}\left(N^m_{,i}\mathbf{1}N^n_{,j} + N^m_{,j}\mathbf{1}N^n_{,i}\right)\sigma^{ij}\,dV\end{aligned} \tag{A.39}$$

A.2.5 Bending Strain

Definition

$$\epsilon_{\alpha\beta} = -\tilde{\mathbf{a}}_{\alpha,\beta}\cdot\tilde{\mathbf{a}}_3 + \mathbf{a}_{\alpha,\beta}\cdot\mathbf{a}_3 \tag{A.40}$$

The goal is to now find the 1st and 2nd variations of strain. Because strain has several terms, it is useful to decompose it into a numerator $f_{\alpha\beta}$ and a denominator g. Let

$$\epsilon_{\alpha\beta} = f_{\alpha\beta}g + \mathbf{a}_{\alpha,\beta}\cdot\mathbf{a}_3 \tag{A.41}$$

$$\delta\epsilon_{\alpha\beta} = \delta f_{\alpha\beta}g + \delta g f_{\alpha\beta} \tag{A.42}$$

$$\delta^2\epsilon_{\alpha\beta} = \delta^2 f_{\alpha\beta}g + \delta f_{\alpha\beta}\delta g + \delta g\delta f_{\alpha\beta} + \delta^2 g f_{\alpha\beta}. \tag{A.43}$$

0th variation

$$f_{\alpha\beta} = \tilde{\mathbf{a}}_{\alpha,\beta} \cdot \tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2 \tag{A.44}$$

$$g = -\frac{1}{|\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2|} \tag{A.45}$$

Strain

$$\epsilon_{\alpha\beta} = f_{\alpha\beta} g + \mathbf{a}_{\alpha,\beta} \cdot \mathbf{a}_3 \tag{A.46}$$

$$x = \frac{1}{|\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2|^3} \tag{A.47}$$

A.2.6 1st Variation

$$\delta f_{\alpha\beta} = \delta\mathbf{u}_{,\alpha\beta} \cdot \tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2 + \delta\mathbf{u}_{,1} \cdot \tilde{\mathbf{a}}_2 \times \tilde{\mathbf{a}}_{\alpha,\beta} + \delta\mathbf{u}_{,2} \cdot \tilde{\mathbf{a}}_{\alpha,\beta} \times \tilde{\mathbf{a}}_1 \tag{A.48}$$

$$\delta g = \frac{1}{|\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2|^3} \left(\delta\mathbf{u}_{,1} \cdot \tilde{\mathbf{a}}_2 \times (\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2) + \delta\mathbf{u}_{,2} \cdot (\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2) \times \tilde{\mathbf{a}}_1\right) = x\delta y \tag{A.49}$$

Strain

$$\begin{aligned}\delta\epsilon_{\alpha\beta} =& \delta f_{\alpha\beta} g + \delta g f_{\alpha\beta} \\ =& -\delta\mathbf{u}_{,\alpha\beta} \cdot \tilde{\mathbf{a}}_3 - \frac{\delta\mathbf{u}_{,1} \cdot \tilde{\mathbf{a}}_2 \times \tilde{\mathbf{a}}_{\alpha,\beta} + \delta\mathbf{u}_{,2} \cdot \tilde{\mathbf{a}}_{\alpha,\beta} \times \tilde{\mathbf{a}}_1}{|\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2|} \\ &+ \frac{\tilde{\mathbf{a}}_{\alpha,\beta} \cdot \tilde{\mathbf{a}}_3}{|\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2|} \left(\delta\mathbf{u}_{,1} \cdot \tilde{\mathbf{a}}_2 \times \tilde{\mathbf{a}}_3 + \delta\mathbf{u}_{,2} \cdot \tilde{\mathbf{a}}_3 \times \tilde{\mathbf{a}}_1\right)\end{aligned} \tag{A.50}$$

$$\delta x = \frac{-3}{|\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2|^5} \left(\delta\mathbf{u}_{,1} \cdot \tilde{\mathbf{a}}_2 \times (\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2) + \delta\mathbf{u}_{,2} \cdot (\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2) \times \tilde{\mathbf{a}}_1\right) \tag{A.51}$$

2nd Variation

$$
\begin{aligned}
\delta^2 f_{\alpha\beta} =& \delta\mathbf{u}_{,\alpha\beta} \cdot [\tilde{\mathbf{a}}_1 \times]\, \delta\mathbf{u}_{,2} - \delta\mathbf{u}_{,\alpha\beta} \cdot [\tilde{\mathbf{a}}_2 \times]\, \delta\mathbf{u}_{,1} + \\
& \delta\mathbf{u}_{,1} \cdot [\tilde{\mathbf{a}}_2 \times]\, \delta\mathbf{u}_{,\alpha\beta} - \delta\mathbf{u}_{,1} \cdot [\tilde{\mathbf{a}}_{\alpha,\beta} \times]\, \delta\mathbf{u}_{,2} + \\
& \delta\mathbf{u}_{,2} \cdot [\tilde{\mathbf{a}}_{\alpha,\beta} \times]\, \delta\mathbf{u}_{,1} - \delta\mathbf{u}_{,2} \cdot [\tilde{\mathbf{a}}_1 \times]\, \delta\mathbf{u}_{,\alpha\beta}
\end{aligned}
\tag{A.52}
$$

$$
\delta^2 g = \delta x \delta y + x \delta^2 y \tag{A.53}
$$

Strain

This quantity is too complicated to expand. The code implementation constructs the second variation of strain from the parts below.

$$
\delta^2 \epsilon_{\alpha\beta} = \delta^2 f_{\alpha\beta} g + \delta f_{\alpha\beta} \delta g + \delta g \delta f_{\alpha\beta} + \delta^2 g f_{\alpha\beta} \tag{A.54}
$$

Useful values

$$
\delta y = \delta\mathbf{u}_{,1} \cdot \tilde{\mathbf{a}}_2 \times (\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2) + \delta\mathbf{u}_{,2} \cdot (\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2) \times \tilde{\mathbf{a}}_1 \tag{A.55}
$$

Lemma:

$$
A \times (B \times C) = (A \cdot C)B - (A \cdot B)C \tag{A.56}
$$

$$
(A \times B) \times C = (C \cdot A)B - (C \cdot B)A \tag{A.57}
$$

$$
\begin{aligned}
\delta^2 y =& - \delta\mathbf{u}_{,1} \cdot [(\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2) \times]\, \delta\mathbf{u}_{,2} + \delta\mathbf{u}_{,2} \cdot [(\tilde{\mathbf{a}}_1 \times \tilde{\mathbf{a}}_2) \times]\, \delta\mathbf{u}_{,1} \\
& + \delta\mathbf{u}_{,1} \cdot (\tilde{\mathbf{a}}_2 \cdot \tilde{\mathbf{a}}_2) \mathbf{1} \delta\mathbf{u}_{,1} - \delta\mathbf{u}_{,1} \cdot (\tilde{\mathbf{a}}_1 \cdot \tilde{\mathbf{a}}_2) \mathbf{1} \delta\mathbf{u}_{,2} \\
& - \delta\mathbf{u}_{,2} \cdot (\tilde{\mathbf{a}}_2 \cdot \tilde{\mathbf{a}}_1) \mathbf{1} \delta\mathbf{u}_{,1} + \delta\mathbf{u}_{,2} \cdot (\tilde{\mathbf{a}}_1 \cdot \tilde{\mathbf{a}}_1) \mathbf{1} \delta\mathbf{u}_{,2} \\
& - \delta\mathbf{u}_{,1} (\tilde{\mathbf{a}}_2 \otimes \tilde{\mathbf{a}}_2) \delta\mathbf{u}_{,1} + \delta\mathbf{u}_{,1} (\tilde{\mathbf{a}}_1 \otimes \tilde{\mathbf{a}}_2) \delta\mathbf{u}_{,2} \\
& + \delta\mathbf{u}_{,2} (\tilde{\mathbf{a}}_2 \otimes \tilde{\mathbf{a}}_1) \delta\mathbf{u}_{,1} - \delta\mathbf{u}_{,2} (\tilde{\mathbf{a}}_1 \otimes \tilde{\mathbf{a}}_1) \delta\mathbf{u}_{,2}
\end{aligned}
\tag{A.58}
$$

A.3 Limit Masks

A.3.1 Loop Scheme Limit Masks

The limit position/displacement masks for Loop's scheme are presented in [38]. With respect to Figure 5.6(a) for a vertex of valence n, the values are as follows:

$$\alpha = 1 - n\beta, \qquad \beta_i = \frac{1}{n}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\left(\frac{2\pi}{n}\right)\right)^2\right). \tag{A.59}$$

The tangent masks for each of the two tangent vectors are defined as:

$$\alpha = 0, \qquad \beta_i = \cos\left(\frac{2\pi i}{n}\right) \tag{A.60}$$

$$\alpha = 0, \qquad \beta_i = \sin\left(\frac{2\pi i}{n}\right). \tag{A.61}$$

A.3.2 Catmull-Clark Scheme Limit Masks

The limit masks for the Catmull-Clark subdivision scheme are reprinted here from [35]. The limit position/displacement mask with respect to Figure 5.6(b) is defined to be

$$\alpha = \frac{n^2}{n(n+5)}, \qquad \beta_i = \frac{4}{n(n+5)}, \qquad \gamma_i = \frac{1}{n(n+5)}. \tag{A.62}$$

The first limit tangent is

$$\begin{aligned} \alpha = 0, \quad \beta_i &= A_n \cos\left(\frac{2\pi i}{n}\right), \\ \gamma_i &= \cos\left(\frac{2\pi i}{n}\right) + \cos\left(\frac{2\pi\,(i+1)}{n}\right), \\ A_n &= 1 + \cos\left(\frac{2\pi}{n}\right) + \cos\left(\frac{\pi}{n}\right)\sqrt{2\left(9 + \cos\left(\frac{2\pi}{n}\right)\right)}. \end{aligned} \tag{A.63}$$

The second limit tangent may be found by substituting $i \leftarrow i+1$ for each vertex. A complete eigen analysis of the Loop and Catmull-Clark subdivision schemes may be found in [59] and [58] respectively.

A.4 Model Problem Definitions

The geometry and material properties for the Belytschko *et. al.*benchmark model problem set [9] are listed here.

A.4.1 Scordelis-Lo Roof Properties

Length L	50.0
Radius R	25.0
Thickness t	0.25
Elastic Modulus	$4.32 \cdot 10^8$
Poisson's Ratio	0.0
Boundary Conditions	Rigid diaphragm support at both ends
Loading	Uniform vertical gravity load, 90.0 per unit area
Vertical Displacement at Mid-side of free edge	0.3024

A.4.2 Pinched Cylinder Properties

Length L	600.0
Radius R	300.0
Thickness t	3.0
Elastic Modulus	$3.0 \cdot 10^6$
Poisson's Ratio	0.3
Boundary Conditions	Rigid diaphragm support at both ends
Loading	Opposing radial loads $F = 1.0$
Radial displacement at point load	$0.18248 \cdot 10^{-4}$

A.4.3 Hemispherical Shell Properties

Radius R	10.0
Thickness t	0.04
Elastic Modulus	$6.825 \cdot 10^7$
Poisson's Ratio	0.3
Boundary Conditions	None (Excluding rigid body motion)
Loading	Opposing radial point loads every 90 °, $F = 2.0$
Radial displacement at mid-side of free edge	0.0924

A.5 XML Interchange Format

A.5.1 XML Model File Format Document Type Definition (DTD)

Below is the document type definition for our XML model interchange file format. It is written in the XML DTD language for specifying XML documents. XML stands for Extensible Modeling Language and is an emerging standard for data exchange. See http://www.wc3.org for information on the structure of DTD documents.

```
<!ELEMENT model (vlist, flist, slist?, clist?, llist?)>

<!ELEMENT vlist (vertex*)>
<!ELEMENT flist (face*)>
<!ELEMENT slist (eref*)>
<!ELEMENT clist (vconstraint*, econstraint*)>
<!ELEMENT llist (fload*, vload*, eload*)>

<!ELEMENT vertex (point, pre_disp*, veloc*)>
<!ATTLIST vertex
id CDATA #REQUIRED
>

<!ELEMENT face (ref*, material)>
<!ATTLIST face id CDATA #REQUIRED>

<!ELEMENT ref EMPTY>
<!ATTLIST ref val CDATA #REQUIRED>

<!ELEMENT eref EMPTY>
<!ATTLIST eref
org  CDATA #REQUIRED
```

```
dst  CDATA #REQUIRED
>

<!ELEMENT econstraint (eref*)>
<!ATTLIST econstraint type CDATA #REQUIRED val CDATA #IMPLIED>

<!ELEMENT vconstraint (ref*)>
<!ATTLIST vconstraint type CDATA #REQUIRED val CDATA #IMPLIED>

<!ELEMENT fconstraint (eref*)>
<!ATTLIST fconstraint type CDATA #REQUIRED val CDATA #IMPLIED>

<!ELEMENT fload (ref*)>
<!ATTLIST fload
type CDATA #REQUIRED
val  CDATA #REQUIRED
>

<!ELEMENT vload (ref*)>
<!ATTLIST vload
type CDATA #REQUIRED
val  CDATA #REQUIRED
>

<!ELEMENT point EMPTY>
<!ATTLIST point
x CDATA #REQUIRED
y CDATA #REQUIRED
z CDATA #REQUIRED
>

<!ELEMENT veloc EMPTY>
<!ATTLIST veloc
x CDATA #REQUIRED
y CDATA #REQUIRED
z CDATA #REQUIRED
>

<!ELEMENT material EMPTY>
<!ATTLIST material
t  CDATA #REQUIRED
E  CDATA #REQUIRED
nu CDATA #REQUIRED
>
```

A.5.2 Example XML Model: Simply Supported Flat Plate with Uniform Vertical Load

The model file below specifies a simply supported flat plate with an applied uniform vertical load. The model consists of a four by four grid of square elements; four central elements and twelve ghost faces. The file structure consists of a list of vertices which are reference by a list of faces. A list of constraints references the affected edges, and a list of loads references the affected faces. The "pre-disp" data field of each vertex gives a displacement which reverts the model to the undeformed configuration; if the model is undeformed each pre_disp field will be the zero vector.

```
<?xml version="1.0"?>
<!DOCTYPE model SYSTEM "/usr/local/dtd/model.dtd">
<model>
  <vlist>
    <vertex id="0">
      <point x="-1" y="-1" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="1">
      <point x="-0.5" y="-1" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="2">
      <point x="0" y="-1" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="3">
      <point x="0.5" y="-1" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="4">
      <point x="1" y="-1" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="5">
      <point x="-1" y="-0.5" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="6">
      <point x="-0.5" y="-0.5" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="7">
      <point x="0" y="-0.5" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="8">
      <point x="0.5" y="-0.5" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="9">
      <point x="1" y="-0.5" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="10">
      <point x="-1" y="0" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="11">
      <point x="-0.5" y="0" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="12">
      <point x="0" y="0" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="13">
      <point x="0.5" y="0" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="14">
      <point x="1" y="0" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="15">
      <point x="-1" y="0.5" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="16">
      <point x="-0.5" y="0.5" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="17">
      <point x="0" y="0.5" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="18">
      <point x="0.5" y="0.5" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="19">
      <point x="1" y="0.5" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="20">
      <point x="-1" y="1" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="21">
      <point x="-0.5" y="1" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="22">
      <point x="0" y="1" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="23">
      <point x="0.5" y="1" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
    <vertex id="24">
      <point x="1" y="1" z="0"/>
      <pre_disp x="0" y="0" z="0"/>
    </vertex>
  </vlist>
  <flist>
    <face id="0">
      <ref val="0"/>
      <ref val="1"/>
      <ref val="6"/>
      <ref val="5"/>
      <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
    </face>
    <face id="1">
      <ref val="1"/>
      <ref val="2"/>
      <ref val="7"/>
      <ref val="6"/>
      <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
    </face>
    <face id="2">
      <ref val="2"/>
      <ref val="3"/>
      <ref val="8"/>
      <ref val="7"/>
      <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
    </face>
    <face id="3">
      <ref val="3"/>
      <ref val="4"/>
      <ref val="9"/>
      <ref val="8"/>
      <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
    </face>
    <face id="4">
      <ref val="5"/>
      <ref val="6"/>
      <ref val="11"/>
      <ref val="10"/>
      <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
    </face>
    <face id="5">
      <ref val="6"/>
      <ref val="7"/>
      <ref val="12"/>
      <ref val="11"/>
      <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
    </face>
    <face id="6">
      <ref val="7"/>
      <ref val="8"/>
      <ref val="13"/>
      <ref val="12"/>
      <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
    </face>
    <face id="7">
      <ref val="8"/>
      <ref val="9"/>
      <ref val="14"/>
      <ref val="13"/>
      <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
    </face>
```

```
<face id="8">
  <ref val="10"/>
  <ref val="11"/>
  <ref val="16"/>
  <ref val="15"/>
  <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
</face>
<face id="9">
  <ref val="11"/>
  <ref val="12"/>
  <ref val="17"/>
  <ref val="16"/>
  <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
</face>
<face id="10">
  <ref val="12"/>
  <ref val="13"/>
  <ref val="18"/>
  <ref val="17"/>
  <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
</face>
<face id="11">
  <ref val="13"/>
  <ref val="14"/>
  <ref val="19"/>
  <ref val="18"/>
  <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
</face>
<face id="12">
  <ref val="15"/>
  <ref val="16"/>
  <ref val="21"/>
  <ref val="20"/>
  <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
</face>
<face id="13">
  <ref val="16"/>
  <ref val="17"/>
  <ref val="22"/>
  <ref val="21"/>
  <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
</face>
<face id="14">
  <ref val="17"/>
  <ref val="18"/>
  <ref val="23"/>
  <ref val="22"/>
  <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
</face>
<face id="15">
  <ref val="18"/>
  <ref val="19"/>
  <ref val="24"/>
  <ref val="23"/>
  <material E="1e+11" nu="0.3" t="0.01" rho="1.0"/>
</face>
</flist>
<clist>
  <econstraint type="simple" val="">
    <eref org="6" dst="7"/>
    <eref org="16" dst="17"/>
    <eref org="7" dst="8"/>
    <eref org="17" dst="18"/>
    <eref org="6" dst="11"/>
    <eref org="8" dst="13"/>
    <eref org="11" dst="16"/>
    <eref org="13" dst="18"/>
  </econstraint>
</clist>
<llist>
  <fload type="uniform" val="0 0 -1e6">
    <ref val="5"/>
    <ref val="6"/>
    <ref val="9"/>
    <ref val="10"/>
  </fload>
</llist>
</model>
```

VITA

Seth Green

Education

University of Washington - Seattle, Washington

- Doctorate in Mechanical Engineering, December 2003

Stanford University - Palo Alto, California

- M.S. Mechanical Engineering, 1998

University of Washington - Seattle, Washington

- B.S. Mechanical Engineering with College Honors, 1996

Publications

Journal Publications

- Green, S., Turkiyyah, G., Storti, D. Second Order Accurate Constraint Formulation for Subdivision Surface Element Simulation of Thin Shells, (in submission to International Journal for Numerical Methods in Engineering)

- Capell, S., Green, S., Curless, B., Popovic, Z., Duchamp, T. Interactive Skeleton Driven Deformations, ACM SIGGRAPH, 2002

Fully Refereed Conference Publications

- Green, S., Turkiyyah, G., Storti, D. Subdivision-Based Multilevel Methods for Large Scale Engineering Simulation of Thin Shells, Proceedings of ACM Symposium on Solid Modeling and Applications, 2002
- Capell, S., Green, S., Curless, B., Popovic, Z., Duchamp, T. A Multiresolution Framework for Dynamic Deformations, ACM SIGGRAPH Symposium on Computer Animation 2001

Conference Presentation

- Green, S., Turkiyyah, G., Storti, D. Methods for the Large Scale Simulation of Blood Cell Membranes, Second International Interdisciplinary Conference on Cardiovascular Medicine and Science. Bethesda, Maryland. July 23-25, 2003