An Improved Finite Element Contact Model for Anatomical Simulations

by

Gentaro Hirota

A Dissertation submitted to the faculty of The University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill

2002

Approved by:

_____

Henry Fuchs, Advisor

_____

Sarang C. Joshi, Reader

_____

Anselmo A. Lastra, Reader

_____

Stephen M. Pizer, Reader

_____

Russell M. Taylor II, Reader

GENTARO HIROTA

An Improved Finite Element Contact Model for Anatomical Simulations.

(Under the direction of Henry Fuchs.)

ABSTRACT

This work focuses on the simulation of mechanical contact between nonlinearly elastic objects such as the components of the human body. The computation of the reaction forces that act on the contact surfaces (contact forces) is the key for designing a reliable contact handling algorithm. In traditional methods, contact forces are often defined as discontinuous functions of deformation, which leads to poor convergence characteristics. This problem becomes especially serious in areas with complicated self-contact such as skin folds.

I introduce a novel penalty method for finite element simulation based on the concept of material depth, *which is the distance between a particle inside an object and the object's boundary. By linearly interpolating pre-computed material depths at node points, contact forces can be* analytically *integrated over contact surfaces without raising the computational cost. The* continuity *achieved by this formulation reduces oscillation and artificial acceleration resulting in a more reliable simulation algorithm.*

This algorithm is implemented as part of an implicit finite element program for static, quasistatic *and* dynamic analysis *of nonlinear viscoelastic solids. I demonstrate its effectiveness in an animation showing realistic effects such as folding skin and sliding contacts of tissues involved in knee flexion. The finite element model of the leg and its internal structures was derived from the Visible Human dataset.*

With this method, it is now easier for engineers and scientists to simulate a wide variety of anatomical and tissue structures such as multiple adjacent organs and biomaterials made of intertwined protein fibers. This method also helps animators to use more anatomically accurate human and animal models to enhance the realism of the generated images.

ACKNOWLEDGMENTS

Thanks to

Henry Fuchs for inspiring me about applications of anatomical simulations with great enthusiasm and for supporting this dissertation work for many years.

Sarang Joshi for helping me to build mathematical insights on my algorithm.

Henry Fuchs, Sarang Joshi, Anselmo Lastra, Stephen Pizer, and Russell Taylor for serving on my committee.

Chris Lee and Karl Reinig of the University of Colorado Health Sciences Center for allowing me to work at their laboratory and providing me anatomical data for the human leg model and for insightful discussion about the use of anatomical simulation in surgical simulations.

Carl Erikson for being the first officemate and for working with me to attack many homework problems.

Andrei State for helping me in my research assistantship in the Ultrasound Augmented Reality Group for 7 years and for sharing a great moment in Siggraph ’96.

Kenneth Hoff for giving me his code “Kenny’s FFD Viewer” as the basis of the GUI of my FEM program.

Jeremy Ackerman for showing me a real cadaver and explaining me anatomy basics.

Susan Fisher for working with me for submitting several papers.

David O’brien for presenting this work in the Computer Animation 2001 in Seoul Korea and for being a great officemate.

Paul Zimmons for numerous helps for PC and for proofreading my entire dissertation.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

chapter 3

$\boldsymbol{X}$	position of a material point in the reference configuration
$\boldsymbol{x}$	position of a material point in the current configuration
ϕ	deformation function
∇_0	partial derivative with respect to the material position
$\boldsymbol{F}$	deformation gradient
ϕ_*^{-1}	pull back operation
ϕ_*	push forward operation
$\boldsymbol{C}$	right Cauchy-Green deformation tensor
$\boldsymbol{b}$	left Cauchy-Green deformation tensor
$\boldsymbol{E}$	Green strain tensor
$\boldsymbol{e}$	Almansi strain tensor
$\lambda_1, \lambda_2, \lambda_3$	positive square roots of eigenvalues of $\boldsymbol{C}$
$\boldsymbol{N}_1, \boldsymbol{N}_2, \boldsymbol{N}_3$	eigenvectors corresponds to the eigenvalues λ_1^2, λ_2^2, and λ_3^2
$\boldsymbol{U}$	stretch tensor: $\boldsymbol{U} = \sqrt{\boldsymbol{C}}$
$D[\boldsymbol{u}]$	directional derivative: $Df(\boldsymbol{x})[\boldsymbol{u}] = \left.\dfrac{d}{d\varepsilon}\right\|_{\varepsilon=0} f(\boldsymbol{x}+\varepsilon\boldsymbol{u})$
$\boldsymbol{\varepsilon}$	small strain tensor
$\boldsymbol{v}$	velocity of a particle
$\boldsymbol{l}$	velocity gradient
$\boldsymbol{d}$	rate of deformation tensor
$\boldsymbol{t}$	traction force
$\boldsymbol{n}$	spatial normal vector
$\boldsymbol{\sigma}$	Cauchy stress tensor
$\boldsymbol{f}$	external volumetric force
$\boldsymbol{r}$	residual force
$\delta\boldsymbol{v}$	virtual velocity

δw	virtual work
v	interior of a body in the current configuration
∂v	boundary of v
da	elementary area in the current configuration
δW	total virtual work
δW_{int}	internal virtual work
V	interior of a body in the reference volume
dV	elementary volume in the reference configuration
dA	elementary area in the reference configuration
J	determinant of $\boldsymbol{F}$
tr	trace operator of matrix
$\boldsymbol{P}$	first Piola-Kirchhoff stress
$\boldsymbol{f}_0$	external volume force per unit reference volume
$\boldsymbol{t}_0$	external traction force per unit reference area.
$\boldsymbol{S}$	second Piola-Kirchhoff stress
Ψ	stored strain energy function (elastic potential), pronounced "SIGH"
$\boldsymbol{\mathcal{C}}$	material elasticity tensor
I_C, II_C, III_C	invariants of $\boldsymbol{C}$
$\delta W_{ext}^{f}(\phi, \delta \boldsymbol{v})$	virtual work done by the external volume force
$\delta W_{ext}^{p}(\phi, \delta \boldsymbol{v})$	virtual work done by the external traction force
$\boldsymbol{\mathcal{C}}$	spatial elasticity tensor, constitutive tensor

chapter 4

$\bar{\boldsymbol{x}}_a$	current position of the a^{th} nodal point
N_a	shape function of the a^{th} nodal point
$\boldsymbol{\xi}$	local 3D coordinates of isoparametric elements, pronounced "KS-EYE"
$\bar{\boldsymbol{X}}_a$	reference position of the a^{th} nodal point
$\bar{\boldsymbol{v}}_a$	velocity of the a^{th} nodal point

$\bar{\boldsymbol{u}}_a$	displacement of the a^{th} nodal point
∇_ξ	$\dfrac{\partial}{\partial \boldsymbol{\xi}}$
$\delta\bar{\boldsymbol{v}}_a$	virtual velocity of
$\boldsymbol{T}_a^{(e)}$	internal equivalent force on the a^{th} nodal point per element e
$\boldsymbol{F}_a^{(e)}$	external equivalent force on the a^{th} nodal point per element e
$\boldsymbol{T}_a, \boldsymbol{F}_a$	total equivalent forces on the a^{th} nodal point
$\boldsymbol{R}_a$	residual force on the a^{th} nodal point
$\mathbf{T}$	total internal equivalent force (array form)
$\mathbf{F}$	total external equivalent force (array form)
$\mathbf{R}$	residual force (array form)
$\delta\mathbf{v}$	virtual velocity (array form)
$\mathbf{x}$	current position of the nodal points (array form)
$\mathbf{K}$	stiffness matrix
$\boldsymbol{K}_{ab}$	components of the stiffness matrix that relate nodal point a to nodal point b
$\delta W_{\boldsymbol{c}}$	constitutive components of the virtual work
$\delta W_{\boldsymbol{\sigma}}$	initial components of the virtual work
$\boldsymbol{K}_{ab}^{(e)}$	per-element contribution to $\boldsymbol{K}_{ab}$
$\boldsymbol{K}_{\boldsymbol{c},ab}^{(e)}$	constitutive component of $\boldsymbol{K}_{ab}^{(e)}$
$\mathcal{C}_{ikjl}^{sym}$	symmetrized constitutive tensor
$\boldsymbol{K}_{\sigma,ab}^{(e)}$	initial stress component of $\boldsymbol{K}_{ab}^{(e)}$

chapter 5

$\mathcal{F}$	interior of a fixed obstacle (foundation), whose interior is, and its exterior
$\partial\mathcal{F}.$	exterior of $\mathcal{F}$
g	gap function
p	pressure

$\partial v^{(c)}$	contact surface
ε_p	contact penalty factor
ψ	contact penalty energy, pronounced “SIGH”
$\partial V^{(c)}$	contact surface in the reference configuration
$\boldsymbol{F}_a^{cont}$	equivalent contact penalty force on the a^{th} nodal point
$\boldsymbol{Q}_i$	position of i^{th} nodal point of a triangle element
$\boldsymbol{Q}$	positions of nodal points of the triangle element in matrix form
$\zeta_1, \zeta_2, \zeta_3$	barycentric coordinates in the triangle element, pronounced “ZAY-tuh”
$\boldsymbol{P}_i$	position of i^{th} nodal point of a tetrahedral element
$\boldsymbol{P}$	positions of nodal points of the tetrahedral element in matrix form
$\xi_1, \xi_2, \xi_3, \xi_4$	barycentric coordinates in the tetrahedral element
$g_1, g_2, \cdots, g_4$	material depths assigned to the node points of the tetrahedral element
G_b	b^{th} coefficients of interpolated depth
$G'_{b,ij}$	b^{th} coefficients of interpolated depth partially differentiated w.r.t. P_{ij}
G''_j	b^{th} coefficients of interpolated depth partially differentiated w.r.t. Q_{*j}
Γ	intersection region in $\zeta_1 - \zeta_2$ space
A_{tri}	area of the triangle element
$\left[\zeta_{1,1}, \zeta_{1,2}\right]$	interval of an edge of the intersection polygon in the ζ_1 direction
α , β	coefficients of the line equation ($\zeta_2 = \alpha\zeta_1 + \beta$) of the edge
Θ	set of all edges of the intersection polygon

chapter 6

W	energy as a function of the mesh configuration
$\mathbf{x}^{(0)}$	initial configuration
$\bar{\boldsymbol{x}}_i^{(0)}$	initial position of i^{th} nodal point.

$\mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{P},\boldsymbol{Q})$	penalty force at (ζ_1,ζ_2) subject to tetrahedral and triangular elements positions $\boldsymbol{P},\boldsymbol{Q}$.
$\boldsymbol{V}$	array composed of $\boldsymbol{P}$ and $\boldsymbol{Q}$
$\mathrm{F}'(\boldsymbol{V})$	penalty force integrated over intersection region Γ
$\boldsymbol{V}'$	neighborhood of $\boldsymbol{V}$
Γ'	intersecting region that corresponds to $\boldsymbol{V}'$
T	affine transformation that rectifies the triangle element
$\boldsymbol{M}$	non-translation part of T
Γ_{tri}	interior of the triangular element
$\boldsymbol{p}_1,\boldsymbol{p}_2,\cdots,\boldsymbol{p}_4$	tetrahedron vertices (transformed by T)
Ω_{tet}	interior of the tetrahedral element
Ω_i	half space defined by the i^{th} side of the tetrahedron
$b_{i,j}(\boldsymbol{V})$	j^{th} coefficient of inequality correspond to Ω_i as a function of $\boldsymbol{V}$
Γ_{tet}	intersection of the plane $\zeta_3=0$ and Ω_{tet}
Γ_i	intersection of the plane $\zeta_3=0$ and Ω_i
I	set of indices of tetrahedron sides that are not parallel to the triangle
Γ	intersecting region of the triangle and the tetrahedron
$\Gamma',\Gamma'_{tet},\Gamma'_i$	Γ,Γ_{tet}, and Γ_i corresponding to $\boldsymbol{V}'$
$area(\Pi)$	area of integration region Π

1 INTRODUCTION

1.1 Motivation

Animal bodies are mainly made of soft tissues. The protein structures filled with fluid cover skeletal systems to perform virtually all the functions necessary for the animal's survival. The flexibility of living tissues is an important aspect of medical application. In the augmented reality ultrasound breast needle biopsy project [State 1996], project members often encountered a situation where the prediction of breast deformations was necessary. **Figure 1** shows a typical physician's view of our system. On the real video images of patients, the system displays a "pit" which gives physicians the illusion that there is a rectangular hole in the skin. Inside the pit, ultrasound images are displayed at the correct 3D positions with respect to the patient's body. However, the shape of the pit is derived from the skin surface geometry that is scanned into the computer before the body is examined with the ultrasound probe. As the probe pushes the skin, the breast deforms, making the pit geometry invalid. As a result, the pit often looks as if it is floating on the depressed skin. Such mis-registration may destroy the illusion of the virtual pit (**Figure 2**). If the deformation can be predicted, one could update the pit geometry to match the real skin surface.

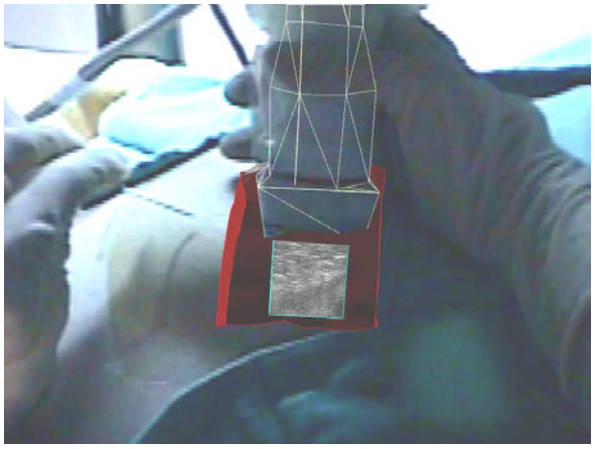

Figure 1: A typical physician's view of an augmented reality breast needle biopsy guidance system. The physician holds the ultrasound probe with her right hand. The captured ultrasound image is displayed at the 3 dimensionally correct position inside the patient's breast. The "pit" (the red square hole) serves as a virtual window opened on the patient's skin surface. The ultrasound slice, the pit, and the wireframe outline of the ultrasound slice are rendered with the 3D computer graphics technique. The rest of image is captured by a camera mounted on the physician's head.

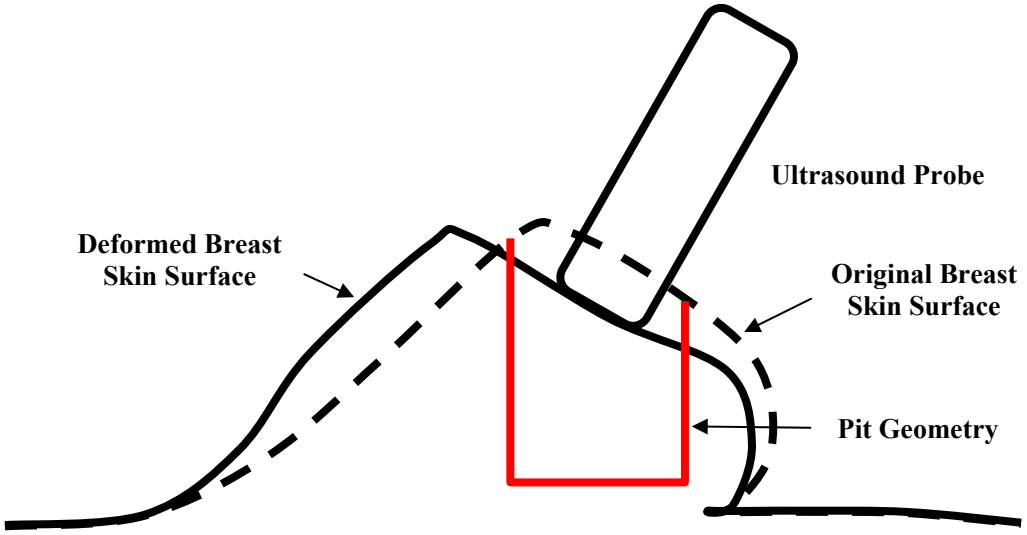

Figure 2: Mis-registration due to deformation. The pit geometry is defined based on the original breast skin surface. After deformation, the skin surface does not agree with the pit geometry.

Whenever pre-acquired images (e.g. CT and MRI 3D images) or geometry (e.g. surface points scanned by laser range finders) of a patient's body are used in a medical operation, the discrepancy due to the body deformation should be taken into account.

The simulation of soft tissue is also important in surgical simulation. An interactive soft tissue simulation directly helps increase the fidelity of the simulation. Less obvious is that a sub-interactive-rate (hence computationally intensive) soft tissue simulation is still useful for building a surgical simulation system. The simulation is useful because most surgical scenarios require procedure-specific postures, which can be derived by deforming standard models such as the "Visible Human" dataset [Spitzer 1996].

Another application of tissue deformation is in movie production. Three-dimensional computer graphics has been used in the movie industry for decades. For movies, the final products are images. Although this is the case for many medical applications mentioned above, the images for movies are often judged by believability as opposed to realism. Therefore, the audience is often simply deceived by *fake* reality rather than something *realistic* (i.e., close to the nature) [Roble 2002]. Thus physically accurate simulation is not a necessity in movie production. On the other hand, people are used to seeing the real world, which is governed by physical laws. Images generated by a 100% accurate simulation must be as believable as the real world. Even if the simulation is not so accurate, it probably increases the believability of the generated images. Another important advantage of physical simulation is the high degree of automation it provides. Ideally, once the scene is set up correctly (which can be quite time-consuming), things happen as they are "supposed" to without human intervention.

The study of tissue deformation covers many different aspects of science. Biological, chemical, electromagnetic and mechanical effects all contribute to deformation. One of the most obvious aspects is the passive mechanical reaction to external forces. Especially if one wishes to simulate deformations caused by musculoskeletal movements and bodies' mechanical interaction with the outside world, a proper treatment of the body's passive behavior is absolutely necessary.

1.2 Overview of Simulation

The macroscopic passive behavior of tissues is studied in the field of continuum mechanics. The relationship between the applied forces and the resulting motions are encoded in partial differential equations (PDE). The simulation of an anatomical structure involves two tasks:

- Defining PDEs for a particular simulation problem
- Solving the PDEs

The first task is called *modeling*. Modeling consists of two parts. One part consists of determining material properties for various tissue types (hence PDEs) and the other part focuses on defining the geometry of organs (i.e., the boundary the PDEs' domain). The boundary conditions may also include PDEs that define forces which act on boundary surfaces (such as air pressures). There are many problems that remain to be solved to build an accurate anatomical model. For example, neither the material models nor the material property measurement methods are fully established. To extract the boundaries of organs from 3D images, segmentation (classification of voxels) has to be done. Unfortunately, segmentation requires significant manual intervention by anatomists.

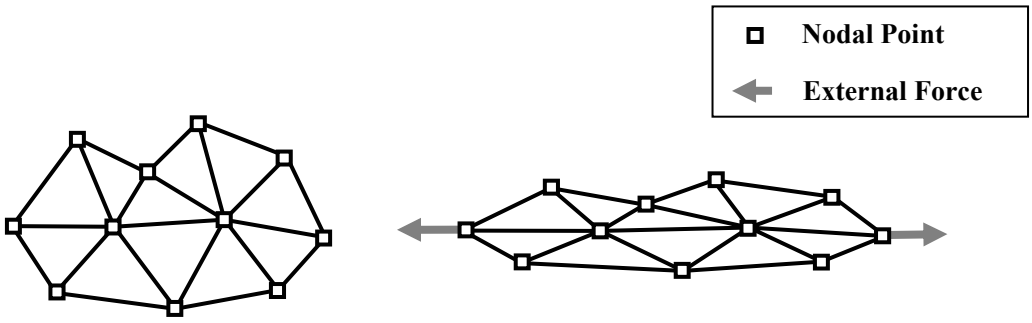

Figure 3: FEM discretization. LEFT: Object's shape is described entirely by nodal points of a finite element mesh. RIGHT: A discretized PDE is a set of equations that defines the movements of nodal points as a result of external forces.

The second task, solving the PDEs, requires a discretization method such as finite element methods (FEM). FEM are used because the nonlinear PDEs employed to express

realistic material behavior do not have analytical solutions. After FEM discretization, the object configuration (shape) is described entirely by a finite number of nodal points. Discretization also converts the PDEs to a system of nonlinear equations. The equations are the forces acting on nodal points, and they are functions of the nodal positions (**Figure 3**). In static analyses, simulation is nothing more than solving the system of nonlinear equations to find an equilibrium (zero-force) configuration. In dynamic analyses, PDEs contain derivatives with respect to time, which is usually discretized by finite difference methods, and the resulting nonlinear system represents, again, an equilibrium state of all participating forces including inertial ones. For an implicit dynamics analysis, the solver seeks a zero force configuration at each point in time.

1.3 The problem

In animal bodies, adjacent components, such as skin, muscles or bones, are not necessarily attached to each other. As body posture changes, organs push and slide against each other, changing the shape of the body. As a joint bends, the skin surface around it may stretch or fold, creating a complicated geometry. Such sliding contacts create challenging problems for the PDE solution.

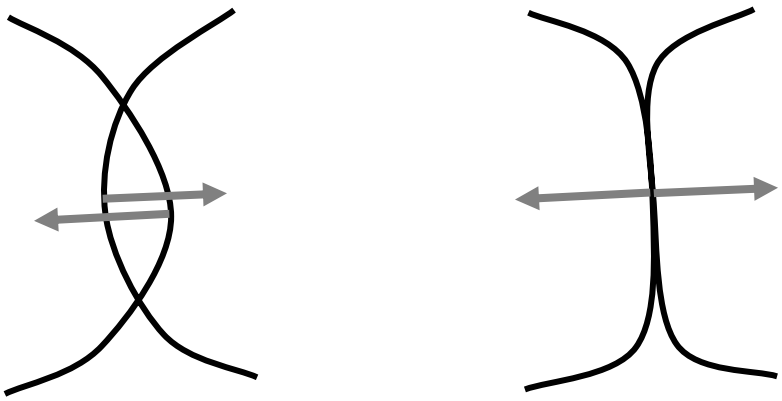

Figure 4: Typical steps for solving the contact problem. First, the penetrating regions are found. Then push-back forces are applied to reduce the amount of penetration. The steps are repeated until a stable configuration is found.

The regions of the boundary surface that are in contact are called contact surfaces. The traction forces acting upon contact surfaces are called contact forces. The contact forces deform the objects. As a result of the deformation, new parts of boundary surfaces may come into contact, or some parts of boundaries previously in contact may be separated. In the simulation, the contact surfaces (or contact forces) are unknown at the beginning. They are usually determined by an iterative algorithm. A typical iterative algorithm repeats two operations until it finds a stable solution (**Figure 4**):

- Find regions where two objects penetrate.
- Apply forces to push back the penetrating region.

The contact forces are the same as the push back forces in the final stable configuration. In conventional methods, the direction of a push back force, or the "normal", is chosen as the direction from a penetrating surface point to the closest projection. Finding a closest projection is quite an expensive operation. Furthermore, in the presence of *self-penetration* and multiple penetrations (more than three or more bodies occupy the same position) or if a more suitable projection is found farther than the closest one, it is no longer easy to mathematically define the push-back force based on projections alone.

In the FEM framework, the contact forces must be applied to each nodal point of the finite element mesh. Because of the difficulty in finding the projection, conventional methods evaluate contact forces only at a limited number of points (typically only at nodal points), which leads to a discontinuity problem. This discontinuity does not disappear even if the surface normal is smooth because the closest projection may jump from one place to another as the object deforms. The discontinuity causes oscillation problems (a phenomenon known as "contact chatter" [Heinstein 1993]) and unnatural abrupt accelerations in simulations (**Figure 5**).

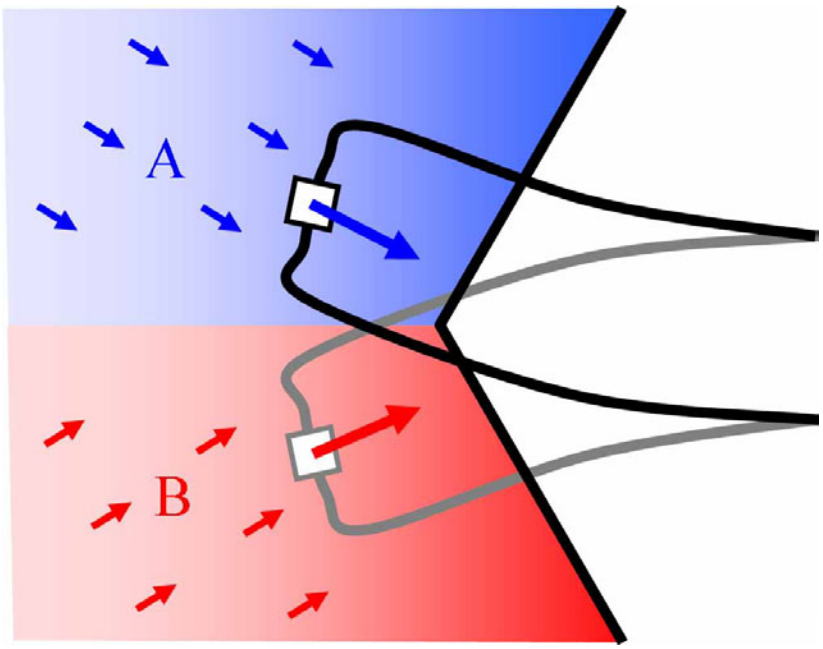

Figure 5: Contact Chatter. A stick (right) intrudes into an object (left) at a concave region. There is a discontinuity in the push-back direction between upper (A) and lower (B) regions. The contact force is sampled at a nodal point (squares). If the stick swings to the upper region, the sampled force pushes the stick downward. If the stick swings to the lower region, the sampled force pushes the stick upward. As a result, the stick oscillates between the two states.

1.4 Thesis Statement and List of Claims

This thesis addresses the continuity problem of contact forces in finite element methods. The thesis is stated as:

> *A finite element formulation based on continuous contact forces integrated over intersecting surface regions provides a reliable computational method for simulating deformations of anatomical structures with complex sliding contacts.*

In this dissertation, I will claim following contributions.

1. I defined a quantity called material depth that is the actual depth from the object surface and is carried with particles as the object is deformed. I showed how material depth provides an unambiguous and useful definition of a gap function.
2. I developed an algorithm that integrates contact forces over contact regions based on linearly interpolated material depths. The integration results in continuous contact forces.
3. I showed the residual errors of the area-integration based algorithm converge to a lower level than errors of a point collocation based algorithm.
4. I demonstrated the proposed algorithm can be applied to realistic anatomical simulation.

These claims will be revisited in section 9.1.

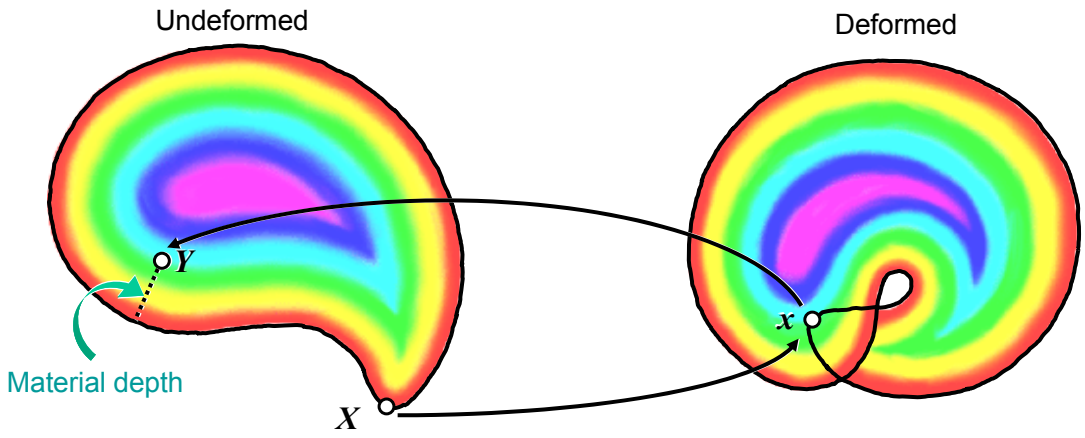

Figure 6: Material depth

1.5 Proposed Solution

I propose an algorithm that significantly reduces the discontinuity of contact forces. The novelty of my algorithm is summarized in the following two points:

- Approximate normals derived from *material depth*: To avoid the complication of projections I derive contact force directions (normals) from material depth, the distance from the object boundary in the *material (reference) configuration* (see Figure 6). In the material configuration, objects are not deformed yet, so there is no self-penetration. Therefore, the material depth can be determined without ambiguity. Furthermore the material depth is evaluated only at the nodal points of FEM meshes and then it is linearly interpolated for throughout the object.
- Analytical area integration of contact forces: The simplicity of the material depth computation enables the analytical integration of contact forces over intersecting areas.

The definition of material depth is very simple; it does not have the ambiguity problems that projection has. Area integration provides contact force continuity, which improves the convergence characteristics of the finite element analysis. **Figure 7** is an illustration of the contact smoothing used in my method.

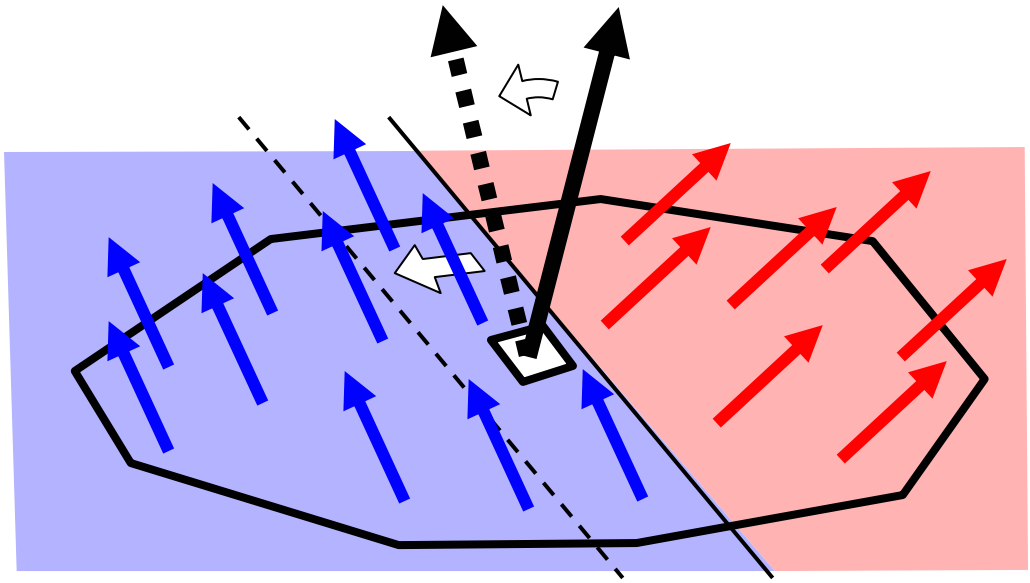

Figure 7: Contact force smoothing by area integration. The nodal contact force is integrated over the local area around the nodal point at the center. The discontinuous border of the contact force moves as objects deforms. A contact force sampled at a particular point changes abruptly. But, since the area-integration averages all changes occur in the area, the nodal force changes continuously.

I also examine my algorithm by using both analytical and numerical methodologies. On the analytical side, I provide proofs of the contact force continuity and the existence of a solution. On the numerical side, I compare my algorithm and a conventional point-sampling-based method and show the superior convergence characteristics of my algorithm.

The algorithm is also applied to anatomical models. A large movement of a musculoskeletal system complete with bones, muscles, and surrounding connective tissues has rarely been simulated. I demonstrate the robustness of my method with a simulation of a large human knee flexion.

2 RELATED WORK

2.1 Finite Element Method

The macroscopic behaviors of soft tissues are described in terms of partial differential equations (PDE). Usually, known external forces and/or geometric constraints define the boundary conditions of the PDE. The Finite Element Method (FEM) is the most commonly used numerical technique to solve these boundary value problems [Fung 1993]. The origin of the FEM is difficult to pinpoint because the mathematical concepts behind the FEM have been around for many years [Zienkiewicz 1995]. The advent of fast electronic circuits for digital computation in the mid-1950s provided the basis for the development of modern finite element methods. The initial application of the finite element method was in the aircraft industry which needed accurate analyses of delta-winged jet planes. Turner et al. introduced triangular and rectangular *panels* which were used to partition the geometry of airplane wings [Turner 1956]. Mathematically, what they did was to partition of the domain of the PDE, which is a fundamental concept in the FEM. Each panel (or *element*) contains nodal points. Each nodal point is assigned physical quantities (such as a position and velocity). These physical values are interpolated between nodal points. Clough later coined the word "Finite Element" to describe these building blocks of computational models [Clough 1960]. The simulation method of Turner et al. proved to be very effective and the finite element method eventually spread to every branch of engineering.

Before the FEM was developed in the engineering field, significant mathematical studies had been done on approximate solutions to PDEs. One approximation method is the linear combination of basis functions. Typical solution methods derive algebraic equations by computing convolutions of weighting functions and the PDE. Then the algebraic equations are solved for the coefficients of their basis functions. One method of choosing these

functions is Galerkin's method [Galerkin 1915] which uses the same basis functions as the weighting functions. The FEM is interpreted as a special case of these approximation methods where each coefficient is a physical quantity assigned to a nodal point. The basis function is a piecewise function that is used to interpolate values between the nodal points. Thus the basis functions used in the FEM have local supports determined by the sizes and shapes of elements. In FEM terminology, the basis and weighting functions are called *shape* and *trial* functions respectively. From the mathematical point of view, having piecewise local support for the basis functions is not essential. Mathematical arguments for error and convergence can be made for more general basis functions. However, the local support of shape functions increases the sparsity of stiffness matrices[1] and results in the efficient computation of the PDE solution.

2.2 Contact Problem

As mentioned in the introduction, contacts are an important aspect in analyzing biomechanical problems. Among such problems, sliding contact between articular[2] cartilages is probably the most extensively studied topic [Donzelli 1995 1998, Ün 2001]. Contacts also occur in man-made machines, and their simulation has been an important issue in many mechanical engineering applications such as the crashworthiness of automobiles. In either case, contact problems impose peculiar boundary conditions on PDEs. Contact forces applied on the boundary surfaces can be seen as external forces, but they are not given a priori. The position of the contact surface is not known either. Yet, the solution of the PDE should satisfy the impenetrability constraint (two objects cannot penetrate each other). The solution method, FEM, has to be extended to deal with this new situation.

[1] The partial derivative of nodal force used in the Newton iteration of nonlinear system solver. See section 4.6 for more detail.

[2] Of or relating to a joint or joints.

2.2.1 Two Body Contact and Self-Contact Problems

The complexity of a contact problem depends on the type of contact. There are two distinct classes of problems—the *two-body contact problem* and the *self-contact problem*. The two-body contact problem occurs when an object (body) contacts only with other objects; an object never comes into contact with itself. Such an assumption is often reasonable when the amount of deformation is relatively small (e.g. articular cartilages). The self-contact problem, however, covers all contact scenarios including two-body contacts and contacts between two parts of the same object.

Most mechanical parts are made of metal. In most engineering applications, the deformations of those nearly-rigid parts are not large enough to cause self-contact unless the purpose of the simulation is to analyze the destruction of the mechanical structures. Therefore, many algorithms are designed exclusively for two-body problems. In biomechanical applications, on the other hand, the building materials are often very flexible; hence there are more chances for self-contact. Below, various algorithms are discussed. The ability to handle self-contact is an important consideration when each algorithm is examined.

2.2.2 Master-Slave Approach and Contact Detection

The concept of *slave nodes* and *master surfaces* (or *master segments* in 2D cases) is used in almost all algorithms. Consider two colliding objects. Nodal points on the surface mesh of one object are designated as slave nodes, whereas master surfaces are selected from the surface mesh of the second object. If a slave node penetrates a master surface, a contact force is applied to push the node back toward the outside of the master surface. Thus the contact force points toward the master surface's normal direction. In more rigorous algorithms, the impenetrability constraint is constructed for each master-slave pair. With each iteration, new surface geometry is obtained by solving an energy minimization problem subject to the impenetrability constraints, and the set of constraints is updated according to the modified geometry. The process repeats until the constraint set becomes stable. Each iteration step can be a time step in a dynamic simulation. It can also be in an internal convergence loop such as the Newton-Raphson iteration [Le Tallec 1994, BONET 1997].

To solve two-body contact problems, it is not difficult to assign the master and slave roles to objects' boundary surfaces. Benson et al. proposed the *single surface algorithm*, which is a master-slave method that can also handle self-contact cases. This algorithm dynamically assigns the master-slave roles based on the result of a neighborhood search [Benson 1990]. Benson et al. also addressed the asymmetry problem of the master-slave strategy by performing two passes at every time step; the first pass is followed by the second pass with reversed master-slave assignments. This algorithm is used in well-known finite element codes (DYNA [LIN 1998] and NIKE) developed at Lawrence Livermore National Laboratory. DYNA spawned a few commercial products (e.g. LS-DYNA) which are used for various contact-impact intensive applications such as the design of highway guardrails [Reid 2001].

The most difficult task in master-slave algorithms is finding the master-slave pairs. This task is often called *contact detection*. A slave node is almost never exactly on the master surface. It is impossible to identify master-slave pairs unambiguously (This problem is discussed in detail in Chapter 5). Therefore, the pairing has to rely on some estimation method. The single surface algorithm uses the nearest-node strategy; for each slave node, the closest node is found, and the surface element that contains the closest node is considered to be the master surface. This strategy often fails when complex self-contact occurs. The algorithm tries to detect adverse situation by examining surface normals, but it cannot handle all geometrically complex cases.

The *splitting pinball method* is a unique algorithm proposed by Belytschko et al. [Belytschko 1991 1992]. This method uses repulsive forces between hierarchical bounding spheres around or near individual surface elements. The contact detection is simplified to an interference check between the spheres. Therefore, this method does not suffer from the contact detection ambiguity problem found in master-slave approaches. However, the algorithm's applicability to complex contact is not clear. The fine details of a surface, such as folding skin, would require that the bounding spheres be repeatedly subdivided, resulting in a high computational cost.

Heinstein et al. proposed a more sophisticated contact detection method that approximates trajectories of slave nodes and master surfaces by linearly interpolating two successive time steps, and then computes the exact time of collision [Heinstein 1993 May

and July]. They use heuristics based on the history of master-slave pairings in previous steps to alleviate problems in complex self-contact cases. Their heuristics also reduce the normal discontinuity and contact chatter problems discussed below. This algorithm is used at Sandia National Laboratories as a part of a dynamic code (PRONTO) and a quasistatic code (SANTOS). The parallelized version of PRONTO was used to simulate very large scale contact-impact problems [Brown 2000].

2.2.3 Contact Force Discontinuity and Surface Smoothness

In master-slave methods, except for rare exceptions, the normal of a master surface is used as the direction of the contact force applied to a slave node. The boundary surfaces are usually triangulated, and the normal is not continuous across the border of triangles. The contact force, seen as a function of nodal point positions, is not continuous. In static analysis, this discontinuity sometimes means that the equilibrium state cannot be achieved. In dynamic analysis, the discontinuity may introduce high frequency noises into the system. A typical phenomenon caused by the discontinuity is the oscillation of slave nodes on the edge between two master surfaces. The phenomenon is often called “contact chatter”.

Padmanabhan et al. used a Hermite cubic curve (their method is for 2D problems) for smoothing [Padmanabhan 1998]. Puso et al. use Gregory patches for boundary surface representation in order to smooth out the normals and, thus, the contact forces [Puso 2001]. However, the method seems to be limited to two-body contact problems. Also, surface smoothing has several problems. First of all, there are numerical concerns in maintaining surface continuity. If continuity is maintained as a set of constraints, they introduce artificial stiffness to the surfaces and increase the discretization error. If automatically continuous surfaces (such as NURBS) are used, the sparsity of the stiffness matrices is compromised, raising computational cost. Secondly, “smooth” surfaces, including Gregory patches and NURBS, do not guarantee smoothness. If buckling occurs in the simulation, normal discontinuity (in the form of cusps) may also emerge. Finally, in many applications, the smoothness of surfaces is not desirable. Surfaces of many objects such as (wrinkled) skin and shock absorbing bellows in automobiles are not supposed to be smooth.

Smooth surface normals do not always mean contact force continuity. A small deformation may change the master-slave pairing and cause the contact force to switch from one direction to the other. The algorithms mentioned above (except for the pinball method) calculate contact forces at slave nodes only. Because of their "point sampling" nature, the contact force applied to a slave node becomes discontinuous again. To resolve the problems of point sampling, researchers have attempted to smooth out contact force discontinuities using intermediate continuous contact surface elements. Papadopoulos et al. proposed a method that projects contacting surface elements to an intermediate plane and defines contact elements as intersections of projected elements [Papadopoulos 1993]. However, there is no mention of dynamically generating intermediate planes for high curvature or self-contacting surfaces. So the idea of intermediate contact elements seems to be applicable to two-body contact between relatively smooth surfaces only.

2.2.4 Issues in the Strict Prevention of Penetration

In the above discussion, it was assumed that the slave node is often found penetrating other objects. One might wonder why those algorithms do not simply prevent such incidents. It is worth commenting on the possibility of preventing penetration completely.

In Heinstein's method [Heistein 1993 May and July], even if the collision times are computed, some penetrations are tolerated for several time steps before they are eliminated. To completely prevent penetrations, the exact birth and death times of impenetrability constraints for slave-master pairs must be computed at every simulation step. Such computation is so expensive that there does not seem to be any algorithm which implements this strategy. Enforcing zero penetration seems practically impossible.

Ideally, if two objects are in contact, they share a single contact surface. However, because it is virtually impossible for two independently discretized surfaces to have common surface geometry, even if the penetration is completely eliminated, gaps (which are equally inaccurate) emerge. A strict impenetrability constraint does not make sense after the exact shape of boundary is compromised by discretization.

2.3 Techniques Used in Entertainment Applications

Most deformation techniques employed in movie special effects and video games use kinematic approaches. Their major advantage is achieving interactive performance due to the algorithm's relatively small computational cost. Two examples of these kinematic methods are free-form deformation (FFD) [Sederberg 1986] and "skinning" or skeleton subspace deformation (SSD) [Lewis 2000]. SSD is the smooth blending of multiple rigid transformations. FFD and SSD belong to a group of algorithms that employ "space deformation" [Bechmann 1994], which can be viewed as a 3D transformation. One can also deform objects by directly moving the control points of surfaces [DeRose 1998]. In these methods, the impenetrability constraint is satisfied by heuristic techniques often requiring extensive user interaction to produce the desired effects.

Space deformation can be applied to the human body without causing penetration between the organs [Bregler 2000], but sliding effects between organs cannot be obtained with this method. Many commercial software packages allow animators to embed formulae to express specific needs for deformations [Anderson 1999]. These "deformers" can be written in such a way that penetration between objects is minimized, but only for specific and limited scenarios. By using pose space deformation [Lewis 2000], one can partially automate the process. For example, a user can "teach" the system to avoid penetration of the skin around an elbow by directly adjusting the skin geometry. Henceforth, the system automatically reduces penetration. The drawback of the method is that it is inflexible since the user must instruct the system about how to handle every new contact scenario.

To overcome the drawbacks of kinematic methods, many of them employ the notion of force and energy. Wilhelms et al. simulate a sliding skin layer by the relaxation of a spring mesh [Wilhelms 1997]. The relaxation scheme does not account for buckling; hence realistic folding does not occur. Accurate physical simulation, which has been developed in the engineering community, can provide a powerful tool for automatically generating realistic deformations. Traditionally, they have been very expensive in terms of computation time but are becoming increasingly affordable with the advent of faster computer hardware. Many graphics researchers have demonstrated animations created by techniques based on physical

principles [Baraff 1992, Koch 1996, O`Brien 1999, Terzopoulos 1987, Zhuang 2000], but none of them addressed the issue of robustness in contact handling.

Recently, Bridson et al. proposed a robust treatment of contacts for cloth animation [Bridson 2002]. The algorithm applies just enough impulses to nodal points to prevent penetration. The drawback of such a strategy is that they have to assume "inelastic" collisions regardless of the material properties. Furthermore, their impulse method does not always resolve complicated geometrical cases. Whenever the method fails, the algorithm creates "rigid impact zones": the deformations of the problematic parts are frozen, and the relative motions between sides in contact are prohibited. Therefore, their method seems to be limited to certain types of high-friction cloth simulation.

3 SOLID MECHANICS OF TISSUES

3.1 Introduction

Most of this chapter is tutorial in nature. For more details, readers are referred to three textbooks listed in the bibliography of this thesis: [Fung 1993], [Bonet 1997], and [Ciarlet 1987].

There is no difference between ordinary artificial or natural materials and biological ones in the sense that they are all made of numerous molecules and atoms. It is often possible to tell chemical properties by examining the kinds of atoms (chemical composition) and their 3D bonding configuration (molecular structure). Mechanical properties, on the other hand, are affected also by the larger scale of these structures. For example, a block of steel behaves quite differently compared to a spring made of steel. A spring may be very flexible and can be squeezed easily, but a steel block is almost rigid. This property is also called the scale dependency of mechanical property. If the mechanical property of a spring is being discussed, it is "flexible". However, if the spring made larger until only the metal part can be seen, it is "hard". This scale dependency is especially important for biological tissues since they have an enormous hierarchy of structure from the visible scale to the molecular level. This hierarchy looks similar to a fractal structure: every time a tissue is magnified, a new structures (and a new mechanical property) emerge.

Heterogeneity of the material exists at all conceivable levels. To perform an accurate simulation, one might wish to go down to as small scale as possible. The smallest possible scale is often determined by the capacity of the computer hardware. Simulation of molecular dynamics is available only for nano-scale phenomena, in which only a handful atoms are involved [SMONDYREV 1999]. To simulate any visible phenomena, one has to use a macroscopic treatment called continuum mechanics, which "homogenizes" the multi-level

heterogeneous structures into a continuous media. In continuum mechanics, the material behavior is described by partial differential equations. Once the partial differential equations are given, the finite element method (or other approximate solution techniques) can be used to perform the simulation even with the limited computational power of current hardware technology.

In this chapter, the hierarchical structure of biological structure is described. Then details of the continuum mechanics for an elastic body are explained. Finally, typical material models for biological tissues are presented.

3.2 Tissue Structure

3.2.1 Tissue Types

A tissue is made of cells and extracellular matrices. The extracellular matrix consists of fibers and a ground substance. Fibers are mainly made of proteins such as collagen and elastin, and the ground substance is often proteoglycans (carbohydrate consisting of various polysaccharide side chains linked to a protein). Animal tissues fall into four different categories: connective, epithelial, muscle, and nerve.

Connective tissues include cartilage, tendons, ligaments, the matrices of bones, the adipose (fatty) tissues, and skin. Any tissue whose main portion is extracellular matrix is considered a connective tissue. Blood and lymph are also classified as connective tissues. The extracellular matrix of skin is composed of randomly and densely interwoven protein (collagen and elastin) fibers. In tendons and flat sheets, on the other hand, the fibers are aligned in roughly the same direction to form dense regular connective tissues. There are also thin fibrous connective tissues, which bind blood vessels as well as the basal membrane to support liver and muscle cells.

The epithelium is made up of sheets of cells that cover the inside and outside surface of organs. It is found on the surface of the lungs, stomach, intestines, and blood vessels. It is

usually built on top of a supportive basal[3] membrane, but, unlike connective tissues, epithelia are tightly packed cells with little extracellular matrix between them.

The muscle cell is the building block of various muscle tissues. There are three types of muscle tissue: striated, smooth, and cardiac. Striated muscles are also called skeletal or voluntary muscles and are usually attached to bones in order to exert forces for articulated movement. Smooth muscle is in the digestive tract, bladder, arteries, and veins and is controlled by the autonomic nervous system. Cardiac muscles function as motors to pump blood.

Nerve cells, or neurons, are designed to transfer signals from one part of body to another. A neuron has a cell body (soma), small branches (dendrites), and a long fiber (axon). The dendrites connect one neuron to another, and the axon transmits signals longer distances. At the end of an axon, there are numerous synapses, which pass the signals to other neurons or muscles.

3.2.2 Collagen

The human body is about 60 percent water. The rest consists of starch, sugar, minerals, fat, and protein. Proteins, especially fibrous ones, provide the structural integrity for many tissues. Fibrous proteins include fibrin, keratin, and collagen. Collagen is as important as steel in man-made structures since it is the most significant material in the human body: collagen accounts for about a third of body's total protein and is contained in almost all tissues. After it is synthesized in cells, collagen is secreted and forms extracellular matrices. Tendon tissue is known to have the highest collagen content, but collagen is also found in skin, cartilage, ligaments, internal organs, muscles, intervertebral discs, basal membranes, and the eye.

[3] Anatomy term meaning "forming a base"

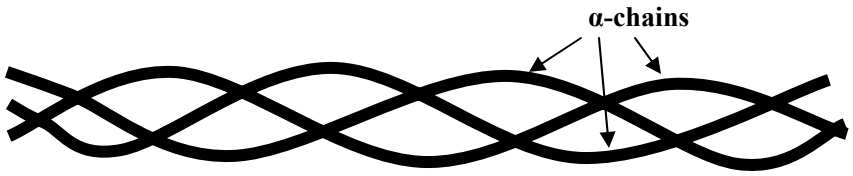

Figure 8: Collagen structure. A collagen molecule consists of three polypeptide chains (α-chains). Three chains twist together into a right handed triple helix.

A collagen molecule consists of three polypeptide chains. The individual polypeptide chain is called an alpha-chain and is a shallow left handed helix. The pitch of the helix is approximately 30 amino acids or 8.7 nm. Three of the chains twist together into a right handed triple helix. There are at least 9 distinct alpha chains; most are approximately 1400 amino acids in length. This results in a diameter of 1.5nm and a length of about 300nm for a single collagen molecule. The structure is established by hydrogen bonds between chains. Collagens are also a group of glycoproteins. Most carbohydrate part is a chain of glucose and galactose.

Collagen from livestock cows is a well-known ingredient for cooking. When it is heated, its structure is lost. The triple helix unwinds and the chains separate. Then, as this tangled chains cools down, they capture all of the surrounding water and form gelatin.

3.2.3 Proteoglycan

Proteoglycans are a group of glycoproteins that make up most of the network that surrounds the collagen molecules in animals. Proteoglycans contain a very high percentage (~95%) of carbohydrates, which makes them special among glycoproteins. The proteoglycans are structured around a long polypeptide core, and carbohydrate chains are linked to the core (see **Figure 9**).

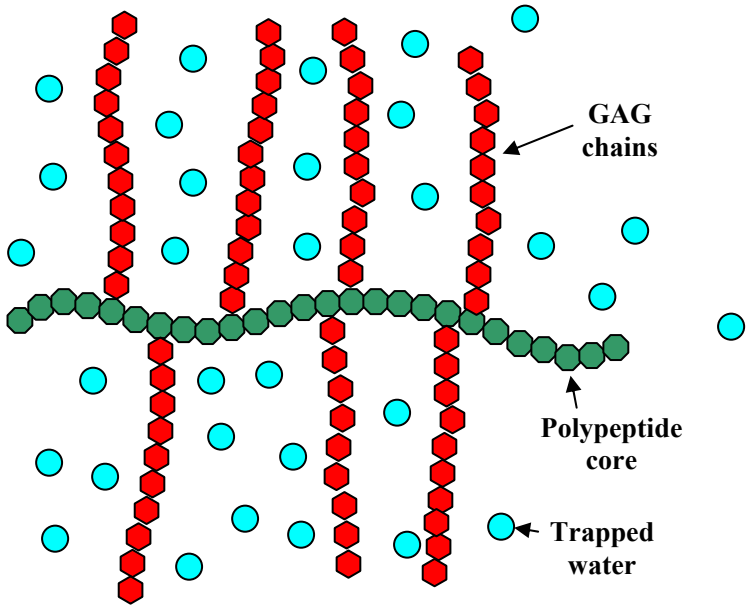

Figure 9: Proteoglycan. Glycosaminoglycan (GAG) chains trap large amount of water, making the structure nearly incompressible.

The main carbohydrate in proteoglycan structure is the glycosaminoglycans (GAGs). These consist of unbranched sugar chains. Networks formed by proteoglycans trap and prevent water molecules from flowing. They may trap 50 times their weight in water. This trapped water allows the complexes to be resistant to compression. Proteoglycans occurs in the extracellular matrix and contribute to the incompressibility of tissues.

3.2.4 Hierarchical Structure and Mechanical Functions

To illustrate the hierarchical structure of biological tissue, tendon tissue will be described in great detail. There are six structural levels in a tendon [Fung F.C. 1993, Kastelic et al. 1978]. The first (finest) level is the collagen molecule, and the sixth level is the tendon itself. The intermediate levels, the second to fifth levels, are the microfibrils, the subfibrils, the fibrils, and the fascicles. These six structural levels are discussed in ascending order.

As described above, a collagen molecule is a hierarchical structure; it is a right handed helix of an alpha-chain, which is a left handed helix itself. It is a superhelix (a helix of helices) with alternative twist handedness. If twist handedness is the same from one level to the other, the structure can be relatively easily unwound. The alternated handedness makes

twisting more difficult, and the tighter structure provides more tensile strength. This twisting technique is also found in artificial structures such as ropes and cables.

Five of the collagen molecules twist together into a left handed helix by electrostatic forces to build a microfibril. Since a collagen molecule is a superhelix, a microfibril is a super-superhelix. And again, it twists with opposite handedness. The diameter of microfibril is 3.5~4nm. The length is unknown. In a microfibril, the helical length of a collagen molecule is 291nm. Between two consecutive collagen molecules, there are 44nm gaps. Together, they make a 335nm (=291+44nm) period. Five sequences of collagens are arranged such that their periods are staggered. Thus every 67nm (=335/5), there is a 44nm gap where there are only 4 collagen molecules in a cross section. Microfibrils are coiled into a subfibril, a right-handed super-super-superhelix with a pitch of approximately 1μm. The diameter varies between 40 to 300 nm. The structure is held in place by the cross-links between collagen molecules. Subfibrils are bundled into a tendon fibril, whose diameter is 1-10μm. The tendon fibrils are assembled into fascicles. Finally fascicles are bundled into a tendon. There is no mechanical attachment between fascicles; fascicles can slide freely against each other. Since tendons are mainly exposed to tensile stress, this structure has a mechanical advantage. Similar lateral decoupling is found in ligaments.

The microscopic structures of tissue give valuable insight on their mechanical properties. For example, collagen fiber orientations are closely related to the anisotropy of material stiffness. The water trapped by proteoglycans is the primary reason of tissue incompressibility. The super-helix structure (i.e., hierarchy of coiling) is often used to explain the exponential growth of material stiffness in incremental stretch experiments. But, as mentioned earlier, the overwhelming complexity of tissue structures prohibits modeling visible-size tissue behavior in terms of microscopic phenomena. Therefore, it is necessary to describe material behaviors at a macroscopic level by using partial differential equations.

3.3 Load Map toward Finite Element Methods

The rest of this chapter introduces basic concepts of continuum solid mechanics and derives the differential equations that describe the macroscopic behaviors of materials. The equations

will be written in *weak forms* so that they can be discretized by finite element methods easily in chapter 4. The discretized equations are solved by using the Newton-Raphson method. The Newton-Raphson method repeatedly solves linearized equations until convergence is achieved. It is certainly possible to linearize the discretized equation. It is also possible to linearize the weak form first and apply discretization. This dissertation uses the latter path of derivation. It is why linearization of various quantities is discussed in this chapter. The discretization of linearized weak forms is explained in chapter 4. Appendix A explains the nomenclature that is used to describe numerous mathematical objects and operators in this dissertation.

3.4 Kinematics

The first topic in continuum mechanics is *Kinematics*. Kinematics studies the deformation of objects without considering the causes (forces and energies). It is a study of geometry as a function of time.

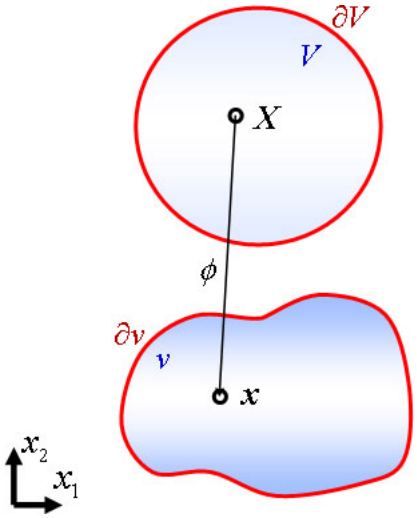

Figure 10: Deformation function.

3.4.1 Deformation

To track the motion of an object, each material point is labeled by a position $\boldsymbol{X}$ in the initial object shape. The material point is mapped by ϕ to a new position $\boldsymbol{x}$. $\boldsymbol{x}$ varies as a function of time t.

$$\boldsymbol{x} = \phi(\boldsymbol{X}, t); \quad \boldsymbol{x}, \boldsymbol{X} \in \mathbb{R}^3 \tag{3.1}$$

ϕ determines the geometry of the object. The geometry is referred as a *configuration*. The initial (undeformed) configuration is referred as the *material* configuration. The deformed configuration is referred as the *current* or, since it is the actual configuration in space, the *spatial* configuration.

3.4.2 Deformation Gradient

The partial derivative of ϕ is given by

$$\boldsymbol{F} = \frac{\partial \phi}{\partial \boldsymbol{X}} = \nabla_0 \phi \tag{3.2}$$

where ∇_0 denotes the partial derivative with respect to the material position. It is often convenient to write $\boldsymbol{x}$ as a function of $\boldsymbol{X}$ and t

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{X}, t) \tag{3.3}$$

So $\boldsymbol{F}$ can also be written as

$$\boldsymbol{F} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} \tag{3.4}$$

or componentwise,

$$F_{i,I} = \frac{\partial x_i}{\partial X_I}; \quad i, I = 1,2,3 \tag{3.5}$$

Consider an elementary vector $d\boldsymbol{X}$ in the material configuration and the corresponding elementary vector $d\boldsymbol{x}$ in current configuration. Obviously

$$d\boldsymbol{x} = \boldsymbol{F} d\boldsymbol{X} \tag{3.6}$$

The inverse of $\boldsymbol{F}$ is the gradient of the inverse of ϕ

$$\boldsymbol{F}^{-1} = \frac{\partial \boldsymbol{X}}{\partial \boldsymbol{x}} = \nabla \phi^{-1} \tag{3.7}$$

Thus $d\boldsymbol{X}$ and $d\boldsymbol{x}$ are transformed from one configuration to another by simple matrix operations

$$d\boldsymbol{x} = \phi_*\left[d\boldsymbol{X}\right] = \boldsymbol{F}d\boldsymbol{X} \tag{3.8}$$

$$d\boldsymbol{X} = \phi_*^{-1}\left[d\boldsymbol{x}\right] = \boldsymbol{F}^{-1}d\boldsymbol{x} \tag{3.9}$$

where $\phi_*\left[\cdot\right]$ is called the *push forward* operation, which is an *operand dependent* operation that transforms a quantity in the material configuration (in this case $d\boldsymbol{X}$) to a quantity in the current configuration (in this example $d\boldsymbol{x}$). Likewise, $\phi_*^{-1}\left[\cdot\right]$ is called the *pull back* operation that transforms a quantity from the current configuration to the material configuration.
Thus, in general, the push forward and pull back operations between material vectors and spatial vectors can be summarized as

$$\boxed{spatial\,vector} = \phi_*\left[\boxed{material\,vector}\right] = \boldsymbol{F}\boxed{material\,vector} \tag{3.10}$$

$$\boxed{material\,vector} = \phi_*^{-1}\left[\boxed{spatial\,vector}\right] = \boldsymbol{F}^{-1}\boxed{spatial\,vector} \tag{3.11}$$

3.4.3 Strain

The geometric properties such as angles and lengths are examined by scalar products of two elementary vectors, and the value stays the same regardless to the choice of coordinate systems. So, to examine the change of geometry, it is worthwhile to see how scalar products are affected by the deformation ϕ. Suppose elementary vectors $d\boldsymbol{x}_1$ and $d\boldsymbol{x}_2$ in the current (spatial) configuration corresponds to $d\boldsymbol{X}_1$ and $d\boldsymbol{X}_2$ in the material configuration. Then the spatial scalar product $d\boldsymbol{x}_1 \cdot d\boldsymbol{x}_2$ is found in terms of the material vectors $d\boldsymbol{X}_1$ and $d\boldsymbol{X}_2$ as

$$d\boldsymbol{x}_1 \cdot d\boldsymbol{x}_2 = d\boldsymbol{X}_1 \cdot \boldsymbol{C}d\boldsymbol{X}_2 \tag{3.12}$$

where $\boldsymbol{C}$ is the *right Cauchy-Green deformation tensor*

$$\boldsymbol{C} = \boldsymbol{F}^T\boldsymbol{F} \tag{3.13}$$

This implies that the local geometric change depends only on $\boldsymbol{C}$.
Similarly, the material scalar product $d\boldsymbol{X}_1 \cdot d\boldsymbol{X}_2$ is found in terms of the spatial vectors $d\boldsymbol{x}_1$ and $d\boldsymbol{x}_2$ as

$$d\boldsymbol{X}_1 \cdot d\boldsymbol{X}_2 = d\boldsymbol{x}_1 \cdot \boldsymbol{b}^{-1}d\boldsymbol{x}_2 \tag{3.14}$$

where $\boldsymbol{b}$ is the *left Cauchy-Green deformation tensor*

$$\boldsymbol{b} = \boldsymbol{F}\boldsymbol{F}^T \tag{3.15}$$

The change of dot product is

$$\frac{1}{2}\left(d\boldsymbol{x}_1 \cdot d\boldsymbol{x}_2 - d\boldsymbol{X}_1 \cdot d\boldsymbol{X}_2\right) = d\boldsymbol{X}_1 \cdot \boldsymbol{E} d\boldsymbol{X}_2 \tag{3.16}$$

where $\boldsymbol{E}$ is the *Lagrangian* or *Green strain tensor*

$$\boldsymbol{E} = \frac{1}{2}\left(\boldsymbol{C} - \boldsymbol{I}\right) \tag{3.17}$$

The same difference can be written in terms of the spatial vectors as

$$\frac{1}{2}\left(d\boldsymbol{x}_1 \cdot d\boldsymbol{x}_2 - d\boldsymbol{X}_1 \cdot d\boldsymbol{X}_2\right) = d\boldsymbol{x}_1 \cdot \boldsymbol{e} d\boldsymbol{x}_2 \tag{3.18}$$

where $\boldsymbol{e}$ is the *Eulerian* or *Almansi strain tensor*

$$\boldsymbol{e} = \frac{1}{2}\left(\boldsymbol{I} - \boldsymbol{b}^{-1}\right) \tag{3.19}$$

$\boldsymbol{E}$ works exclusively on material vectors. Such a tensor is called a *material tensor*. On the other hand, $\boldsymbol{e}$ operates only on the spatial vectors. Such a tensor is called a *spatial tensor*. The push forward and pull back operations between $\boldsymbol{E}$ and $\boldsymbol{e}$ are

$$\boldsymbol{e} = \phi_*\left[\boldsymbol{E}\right] = \boldsymbol{F}^{-T}\boldsymbol{E}\boldsymbol{F}^{-1} \tag{3.20}$$

$$\boldsymbol{E} = \phi_*\left[\boldsymbol{e}\right] = \boldsymbol{F}^{T}\boldsymbol{e}\boldsymbol{F} \tag{3.21}$$

Thus, in general, the push forward and pull back operations between material (second order) tensors and spatial (second order) tensors can be summarized as

$$\boxed{spatial\,tensor} = \phi_*\left[\boxed{material\,tensor}\right] = \boldsymbol{F}^{-T}\boxed{material\,tensor}\boldsymbol{F}^{-1} \tag{3.22}$$

$$\boxed{material\,tensor} = \phi_*^{-1}\left[\boxed{spatial\,tensor}\right] = \boldsymbol{F}^{T}\boxed{spatial\,tensor}\boldsymbol{F} \tag{3.23}$$

3.4.4 Polar Decomposition

$\boldsymbol{F}$ contains rigid body motion and "real" deformation. As shown below, $\boldsymbol{F}$ can be decomposed into a rigid motion (rotation tensor) and an intrinsic deformation (stretch tensor).

First, suppose λ_1^2, λ_2^2 and λ_3^2 are the eigenvalues of $\boldsymbol{C}$ and $\begin{bmatrix}\boldsymbol{N}_1 & \boldsymbol{N}_2 & \boldsymbol{N}_3\end{bmatrix}$ is the eigenvector triad

$$\boldsymbol{C}\begin{bmatrix}\boldsymbol{N}_1 & \boldsymbol{N}_2 & \boldsymbol{N}_3\end{bmatrix}=\begin{bmatrix}\boldsymbol{N}_1 & \boldsymbol{N}_2 & \boldsymbol{N}_3\end{bmatrix}\begin{bmatrix}\lambda_1^2 & 0 & 0\\ 0 & \lambda_2^2 & 0\\ 0 & 0 & \lambda_3^2\end{bmatrix} \tag{3.24}$$

Since $\boldsymbol{C}$ is symmetric and positive definite, λ_1^2, λ_2^2 and λ_3^2 are all positive real numbers, and $\begin{bmatrix}\boldsymbol{N}_1 & \boldsymbol{N}_2 & \boldsymbol{N}_3\end{bmatrix}$ can be chosen to be orthonormal. Thus (3.24) is rewritten as

$$\boldsymbol{C}=\sum_{\alpha=1}^{3}\lambda_\alpha^2\boldsymbol{N}_\alpha\otimes\boldsymbol{N}_\alpha \tag{3.25}$$

Now consider a stretch tensor $\boldsymbol{U}$

$$\boldsymbol{U}=\sum_{\alpha=1}^{3}\lambda_\alpha\boldsymbol{N}_\alpha\otimes\boldsymbol{N}_\alpha \tag{3.26}$$

This is the square root of $\boldsymbol{C}$ ($\boldsymbol{U}=\sqrt{\boldsymbol{C}}$) since

$$\begin{aligned}\boldsymbol{U}^2=\boldsymbol{U}\boldsymbol{U}&=\sum_{\alpha,\beta=1}^{3}\left(\lambda_\alpha\boldsymbol{N}_\alpha\otimes\boldsymbol{N}_\alpha\right)\left(\lambda_\beta\boldsymbol{N}_\beta\otimes\boldsymbol{N}_\beta\right)\\ &=\sum_{\alpha,\beta=1}^{3}\lambda_\alpha\lambda_\beta\left(\boldsymbol{N}_\alpha\otimes\boldsymbol{N}_\alpha\right)\left(\boldsymbol{N}_\beta\otimes\boldsymbol{N}_\beta\right)\\ &=\sum_{\alpha,\beta=1}^{3}\lambda_\alpha\lambda_\beta\left(\left(\boldsymbol{N}_\alpha\otimes\boldsymbol{N}_\alpha\right)\boldsymbol{N}_\beta\otimes\boldsymbol{N}_\beta\right)\\ &=\sum_{\alpha,\beta=1}^{3}\lambda_\alpha\lambda_\beta\left(\left(\boldsymbol{N}_\beta\cdot\boldsymbol{N}_\alpha\right)\boldsymbol{N}_\alpha\otimes\boldsymbol{N}_\beta\right)\\ &=\sum_{\alpha,\beta=1}^{3}\lambda_\alpha\lambda_\beta\left(\delta_{\alpha\beta}\boldsymbol{N}_\alpha\otimes\boldsymbol{N}_\beta\right)\\ &=\sum_{\alpha=1}^{3}\lambda_\alpha^2\left(\boldsymbol{N}_\alpha\otimes\boldsymbol{N}_\alpha\right)\\ &=\boldsymbol{C}\end{aligned} \tag{3.27}$$

A tensor defined as

$$\boldsymbol{R}=\boldsymbol{F}\boldsymbol{U}^{-1} \tag{3.28}$$

is a rotation tensor since

$$\boldsymbol{R}^T\boldsymbol{R}=\boldsymbol{U}^{-T}\boldsymbol{F}^T\boldsymbol{F}\boldsymbol{U}^{-1}=\boldsymbol{U}^{-T}\boldsymbol{C}\boldsymbol{U}^{-1}=\boldsymbol{U}^{-T}\boldsymbol{U}\boldsymbol{U}\boldsymbol{U}^{-1}=\boldsymbol{I} \tag{3.29}$$

Thus $\boldsymbol{F}$ is decomposed as

$$\boldsymbol{F} = \boldsymbol{R}\boldsymbol{U} \tag{3.30}$$

This decomposition is interpreted as follows. A material vector $d\boldsymbol{X}$ is transformed to a spatial vector $d\boldsymbol{x}$ as

$$d\boldsymbol{x} = \boldsymbol{F}d\boldsymbol{X} = \boldsymbol{R}\left(\boldsymbol{U}d\boldsymbol{X}\right) \tag{3.31}$$

The first multiplication $\boldsymbol{U}d\boldsymbol{X}$ stretches $d\boldsymbol{X}$ with the factors λ_1, λ_2, and λ_3 in the principal directions $\boldsymbol{N}_1, \boldsymbol{N}_2$, and $\boldsymbol{N}_2$. Then the second multiplication with $\boldsymbol{R}$ rotates the vector to the final orientation.

3.4.5 Linearized Deformation Gradient

Since the Newton-Raphson iteration is used later to solve the nonlinear system, linearization of various quantities must be considered. Here the deformation gradient is linearized. By applying the definition of the directional derivative, $\boldsymbol{F}$ is linearized as

$$\begin{aligned} D\boldsymbol{F}\left(\phi_t\right)\left[\boldsymbol{u}\right] &= \left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} \boldsymbol{F}\left(\phi_t + \varepsilon\boldsymbol{u}\right) \\ &= \left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} \frac{\partial\left(\phi_t + \varepsilon\boldsymbol{u}\right)}{\partial\boldsymbol{X}} \\ &= \left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} \left(\frac{\partial\phi_t}{\partial\boldsymbol{X}} + \varepsilon\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{X}}\right) \\ &= \frac{\partial\boldsymbol{u}}{\partial\boldsymbol{X}} \\ &= \left(\nabla\boldsymbol{u}\right)\boldsymbol{F} \end{aligned} \tag{3.32}$$

Or alternatively

$$D\boldsymbol{F}\left[\boldsymbol{u}\right] = \frac{\partial\boldsymbol{u}\left(\boldsymbol{X}\right)}{\partial\boldsymbol{X}} = \nabla_0\boldsymbol{u} \tag{3.33}$$

3.4.6 Linearized Strain

The material strain tensor is linearized as

$$
\begin{aligned}
D\boldsymbol{E}[\boldsymbol{u}] &= \tfrac{1}{2}\left(\boldsymbol{F}^T D\boldsymbol{F}[\boldsymbol{u}] + D\boldsymbol{F}^T[\boldsymbol{u}]\boldsymbol{F}\right) \\
&= \tfrac{1}{2}\left(\boldsymbol{F}^T \nabla \boldsymbol{u}\boldsymbol{F} + \boldsymbol{F}^T \left(\nabla \boldsymbol{u}\right)^T \boldsymbol{F}\right) \\
&= \tfrac{1}{2}\boldsymbol{F}^T \left(\nabla \boldsymbol{u} + \left(\nabla \boldsymbol{u}\right)^T\right)\boldsymbol{F}
\end{aligned}
\tag{3.34}
$$

Here, the *small strain tensor* $\boldsymbol{\varepsilon}$ is introduced as,

$$
\boldsymbol{\varepsilon} = \frac{1}{2}\left[\nabla \boldsymbol{u} + \left(\nabla \boldsymbol{u}\right)^T\right] \tag{3.35}
$$

Then, the directional derivative of $\boldsymbol{E}$ can be written as the pull back of $\boldsymbol{\varepsilon}$

$$
D\boldsymbol{E}[\boldsymbol{u}] = \phi_*^{-1}[\boldsymbol{\varepsilon}] = \tfrac{1}{2}\boldsymbol{F}^T\boldsymbol{\varepsilon}\boldsymbol{F} \tag{3.36}
$$

3.4.7 Velocity

As will be explained later, the final balance equation is derived from the principle of virtual work. It facilitates the derivation to describe virtual work in a per unit time basis (therefore, it can be seen as the principle of virtual power). Therefore, it is useful to describe various quantities in terms of their rate or velocity. The simplest example is the velocity of a particle (material point). Recall that the position of a particle is

$$
\boldsymbol{x} = \phi\left(\boldsymbol{X}, t\right) \tag{3.37}
$$

Now the velocity of the particle is

$$
\boldsymbol{v}\left(\boldsymbol{X}, t\right) = \frac{\partial \phi\left(\boldsymbol{X}, t\right)}{\partial t} \tag{3.38}
$$

Although $\boldsymbol{v}$ is a spatial vector, it is written as a function of the material position $\boldsymbol{X}$. Instead, $\boldsymbol{v}$ can be written as a function of the spatial position $\boldsymbol{x}$

$$
\boldsymbol{v}\left(\boldsymbol{x}, t\right) = \boldsymbol{v}\left(\phi^{-1}\left(\boldsymbol{x}, t\right), t\right) \tag{3.39}
$$

3.4.8 Velocity Gradient

The velocity gradient $\boldsymbol{l}$ is the partial derivative of the velocity with respect to the spatial coordinates

$$
\boldsymbol{l} = \frac{\partial \boldsymbol{v}\left(\boldsymbol{x}, t\right)}{\partial \boldsymbol{x}} = \nabla \boldsymbol{v} \tag{3.40}
$$

By manipulating the time derivative of the deformation gradient, an interesting relationship is revealed

$$\dot{F} = \frac{d}{dt}\left(\frac{\partial \phi}{\partial \boldsymbol{X}}\right) = \frac{\partial}{\partial \boldsymbol{X}}\left(\frac{\partial \phi}{\partial t}\right) = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{X}} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}}\frac{\partial \phi}{\partial \boldsymbol{X}} = \boldsymbol{l}\boldsymbol{F} \tag{3.41}$$

$$\boldsymbol{l} = \dot{\boldsymbol{F}}\boldsymbol{F}^{-1} \tag{3.42}$$

3.4.9 Rate of Deformation

In 3.4.3, the strain was derived by examining the change of the scalar product of elementary vectors. Similarly the strain rate is derived from the time derivative of the scalar product.
As seen in 3.4.3,

$$d\boldsymbol{x}_1 = \boldsymbol{F}d\boldsymbol{X}_1; \quad d\boldsymbol{x}_2 = \boldsymbol{F}d\boldsymbol{X}_2 \tag{3.43}$$

$$d\boldsymbol{x}_1 \cdot d\boldsymbol{x}_2 = d\boldsymbol{X}_1 \cdot \boldsymbol{C}d\boldsymbol{X}_2 \tag{3.44}$$

Now, by taking time derivative and noting $\boldsymbol{E} = \frac{1}{2}(\boldsymbol{C} - \boldsymbol{I})$

$$\frac{d}{dt}\left(d\boldsymbol{x}_1 \cdot d\boldsymbol{x}_2\right) = d\boldsymbol{X}_1 \cdot \dot{\boldsymbol{C}}d\boldsymbol{X}_2 = 2d\boldsymbol{X}_1 \cdot \dot{\boldsymbol{E}}d\boldsymbol{X}_2 \tag{3.45}$$

where $\dot{\boldsymbol{E}}$ is the *material strain rate tensor*

$$\dot{\boldsymbol{E}} = \tfrac{1}{2}\dot{\boldsymbol{C}} = \tfrac{1}{2}\left(\dot{\boldsymbol{F}}^T\boldsymbol{F} + \boldsymbol{F}^T\dot{\boldsymbol{F}}\right) \tag{3.46}$$

By substituting

$$d\boldsymbol{X}_1 = \boldsymbol{F}^{-1}d\boldsymbol{x}_1; \quad d\boldsymbol{X}_2 = \boldsymbol{F}^{-1}d\boldsymbol{x}_2 \tag{3.47}$$

into (3.45)

$$\frac{1}{2}\frac{d}{dt}\left(d\boldsymbol{x}_1 \cdot d\boldsymbol{x}_2\right) = d\boldsymbol{x}_1 \cdot \left(\boldsymbol{F}^{-T}\dot{\boldsymbol{E}}\boldsymbol{F}^{-1}\right)d\boldsymbol{x}_2 \tag{3.48}$$

Here, the rate of deformation tensor $\boldsymbol{d}$ is introduced

$$\boldsymbol{d} = \phi_*\left[\dot{\boldsymbol{E}}\right] = \boldsymbol{F}^{-T}\boldsymbol{E}\boldsymbol{F}^{-1}; \quad \boldsymbol{E} = \phi_*^{-1}\left[\boldsymbol{d}\right] = \boldsymbol{F}^T\boldsymbol{d}\boldsymbol{F} \tag{3.49}$$

It turns out that $\boldsymbol{d}$ is the symmetric part of $\boldsymbol{l}$

$$\boldsymbol{d} = \tfrac{1}{2}\left(\boldsymbol{l} + \boldsymbol{l}^T\right) \tag{3.50}$$

since

$$
\begin{aligned}
\boldsymbol{d} &= \phi_*\left[\dot{\boldsymbol{E}}\right] \\
&= \boldsymbol{F}^{-T}\boldsymbol{E}\boldsymbol{F}^{-1} \\
&= \tfrac{1}{2}\boldsymbol{F}^{-T}\left(\dot{\boldsymbol{F}}^{T}\boldsymbol{F} + \boldsymbol{F}^{T}\dot{\boldsymbol{F}}\right)\boldsymbol{F}^{-1} \\
&= \tfrac{1}{2}\left(\boldsymbol{F}^{-T}\dot{\boldsymbol{F}}^{T} + \dot{\boldsymbol{F}}\boldsymbol{F}^{-1}\right) \\
&= \tfrac{1}{2}\left(\boldsymbol{l} + \boldsymbol{l}^{T}\right)
\end{aligned}
\tag{3.51}
$$

3.5 Stress and Equilibrium

In the previous section, the geometric aspect of deformation was discussed. In this section, the cause of the deformation, force or stress, is discussed.

3.5.1 Cauchy stress tensor

In a continuum medium, adjacent parts push or pull each other via some surface areas. Such a force, called the *traction force*, should be measured per unit area. The traction force depends on the orientation of the surface. Given the spatial normal vector $\boldsymbol{n}$ of the surface, the spatial traction force $\boldsymbol{t}$ is

$$
\boldsymbol{t} = \boldsymbol{\sigma}\boldsymbol{n} \tag{3.52}
$$

where the spatial tensor $\boldsymbol{\sigma}$ is called the *Cauchy stress tensor*.

In indicial form

$$
\begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} \tag{3.53}
$$

This equation is interpreted as follows. Column I of $\boldsymbol{\sigma}$, $\begin{bmatrix} \sigma_{1I} \\ \sigma_{2I} \\ \sigma_{3I} \end{bmatrix}$, corresponds to the traction force on a surface perpendicular to the coordinate axis X_I. Thus the traction force on an arbitrary oriented surface is the linear combination of these three traction forces scaled by components of the surface's normal vectors.

3.5.2 Equilibrium

The total force acting on an elementary volume is the summation of the off-balance internal force (the divergence of the stress) and the external volumetric force

$$\operatorname{div}\boldsymbol{\sigma} + \boldsymbol{f}$$

In the equilibrium state, this force, the residual force, vanishes

$$\boldsymbol{r} = \operatorname{div}\boldsymbol{\sigma} + \boldsymbol{f} = 0 \tag{3.54}$$

3.5.3 Principle of Virtual Work

The equation (3.54) is, in general, not easy to satisfy. The difficulty stems from the fact that (3.54), often referred to as a *strong form*, is a condition for every elementary volume in the material. The *weak form* or *the principle of virtual work* is obtained by converting the strong form into a condition for the entire *body* made of the material. The weak form enables the derivation of the finite element method that provides approximate solution of the equation. The finite element method is discussed in the next chapter. Here the principle of virtual work is derived from (3.54).

Suppose the object is moving with an arbitrary or *virtual* velocity $\delta\boldsymbol{v}$. At the equilibrium state, the residual force $\boldsymbol{r}$ does not do any work

$$\delta w = \boldsymbol{r} \cdot \delta\boldsymbol{v} = 0 \tag{3.55}$$

The total virtual work δW done in the volume v is also zero

$$\delta W = \int_v \left(\operatorname{div}\boldsymbol{\sigma} + \boldsymbol{f}\right) \cdot \delta\boldsymbol{v} dv = 0 \tag{3.56}$$

By applying the chain rule

$$\operatorname{div}\left(\boldsymbol{\sigma}\delta\boldsymbol{v}\right) = \left(\operatorname{div}\boldsymbol{\sigma}\right) \cdot \delta\boldsymbol{v} + \boldsymbol{\sigma} : \nabla\delta\boldsymbol{v} \tag{3.57}$$

Using this and the Gauss theorem

$$\begin{aligned} &\int_v \left(\operatorname{div}\boldsymbol{\sigma} + \boldsymbol{f}\right) \cdot \delta\boldsymbol{v} dv \\ &= \int_v \left(\operatorname{div}\left(\boldsymbol{\sigma}\delta\boldsymbol{v}\right) - \boldsymbol{\sigma} : \nabla\delta\boldsymbol{v}\right) dv + \int_v \boldsymbol{f} \cdot \delta\boldsymbol{v}\, dv \\ &= \int_{\partial v} \boldsymbol{n} \cdot \boldsymbol{\sigma}\delta\boldsymbol{v}\, da - \int_v \boldsymbol{\sigma} : \nabla\delta\boldsymbol{v}\, dv + \int_v \boldsymbol{f} \cdot \delta\boldsymbol{v}\, dv \end{aligned} \tag{3.58}$$

By (3.52) and (3.40)

$$\int_v \boldsymbol{\sigma} : \delta\boldsymbol{l}\, dv - \int_v \boldsymbol{f} \cdot \delta\boldsymbol{v}\, dv - \int_{\partial v} \boldsymbol{t} \cdot \delta\boldsymbol{v}\, da \tag{3.59}$$

Noting the symmetry of $\boldsymbol{\sigma}$ and (3.51)

$$\delta W = \int_v \boldsymbol{\sigma} : \delta \boldsymbol{d}\, dv - \int_v \boldsymbol{f} \cdot \delta \boldsymbol{v}\, dv - \int_{\partial v} \boldsymbol{t} \cdot \delta \boldsymbol{v}\, da = 0 \tag{3.60}$$

Since this equation is using only spatial quantities, it is called the Eulerian principle of virtual work.

Note that qualifier "virtual" is used to make physical quantities hypothetical ones. Virtual velocity is not the actual velocity of the body; hence virtual work is not an actual energy increase, either. As shown in section 4.5, they are mathematical instruments to derive nodal forces in finite element methods. In the final simulation code, there are no virtual quantities.

3.5.4 Work Conjugacy

As seen in (3.60), the product of $\boldsymbol{\sigma}$ and $\boldsymbol{d}$ gives work per unit volume per unit time. Such relationship is termed *work conjugacy*. It can be also phrased as "$\boldsymbol{\sigma}$ and $\boldsymbol{d}$ are *work conjugate.*" or "$\boldsymbol{d}$ is work conjugate to $\boldsymbol{\sigma}$."

3.5.5 First Piola-Kirchhoff Stress

(3.59) and (3.60) are Eulerian (spatial) virtual works. It is often useful to have a Lagrangian (material) counterpart of the virtual work. In this and next section, the material virtual work is derived. First, the first terms of (3.59) and (3.60) are called *internal virtual work* and denoted by δW_{int}

$$\delta W_{\text{int}} = \int_v \boldsymbol{\sigma} : \delta \boldsymbol{l}\, dv = \int_v \boldsymbol{\sigma} : \delta \boldsymbol{d}\, dv \tag{3.61}$$

This is integrated over the spatial volume v. Changing the integration region to the material volume V and also using the relationship $\boldsymbol{l} = \dot{\boldsymbol{F}}\boldsymbol{F}^{-1}$ introduced in (3.42)

$$\begin{aligned} \delta W_{\text{int}} &= \int_V J\boldsymbol{\sigma} : \delta \boldsymbol{l}\, dV \\ &= \int_V J\boldsymbol{\sigma} : \left(\delta \dot{\boldsymbol{F}}\boldsymbol{F}^{-1}\right) dV \\ &= \int_V tr\left(J\boldsymbol{F}^{-1}\boldsymbol{\sigma}\delta \dot{\boldsymbol{F}}\right) dV \\ &= \int_V \boldsymbol{P} : \delta \dot{\boldsymbol{F}}\, dV \end{aligned} \tag{3.62}$$

where the *first Piola-Kirchhoff stress* $\boldsymbol{P}$ is defined as

$$\boldsymbol{P} = J\boldsymbol{\sigma}\boldsymbol{F}^{-T} \tag{3.63}$$

Now a new virtual principle of work is obtained as

$$\delta W = \int_V \boldsymbol{P} : \delta \dot{\boldsymbol{F}}\, dV - \int_V \boldsymbol{f}_0 \cdot \delta \boldsymbol{v}\, dV - \int_{\partial V} \boldsymbol{t}_0 \cdot \delta \boldsymbol{v}\, dA = 0 \tag{3.64}$$

where $\boldsymbol{f}_0$ is the external volume force per unit initial volume, $\boldsymbol{t}_0$ is the external traction force per unit initial area.

As $\boldsymbol{\sigma}$ and $\boldsymbol{d}$ in the previous section, the product of $\boldsymbol{P}$ and $\dot{\boldsymbol{F}}$ gives work done per unit volume per unit time. Therefore, they are also said to be work conjugate.

3.5.6 The Second Piola-Kirchhoff Stress

An alternative form of virtual work can be derived as

$$\begin{aligned} \delta W_{\text{int}} &= \int_v \boldsymbol{\sigma} : \delta \boldsymbol{d}\, dv \\ &= \int_V J\boldsymbol{\sigma} : \left(\boldsymbol{F}^{-T} \delta \dot{\boldsymbol{E}} \boldsymbol{F}^{-1} \right) dV \\ &= \int_V tr\left(\boldsymbol{F}^{-1} J\boldsymbol{\sigma} \boldsymbol{F}^{-T} \delta \dot{\boldsymbol{E}} \right) dV \\ &= \int_V \boldsymbol{S} : \delta \dot{\boldsymbol{E}}\, dV \end{aligned} \tag{3.65}$$

where $\boldsymbol{S}$ is the *second Piola-Kirchhoff stress* defined as

$$\boldsymbol{S} = J\boldsymbol{F}^{-1}\boldsymbol{\sigma}\boldsymbol{F}^{-T} \tag{3.66}$$

The new material principle of virtual work is given as

$$\delta W = \int_V \boldsymbol{S} : \delta \dot{\boldsymbol{E}}\, dV - \int_V \boldsymbol{f}_0 \cdot \delta \boldsymbol{v}\, dV - \int_{\partial V} \boldsymbol{t}_0 \cdot \delta \boldsymbol{v}\, dA = 0 \tag{3.67}$$

This equation is completely described by material quantities and, therefore, is called the Eulerian principle of virtual work. This equation is used to derive the linearized virtual work in Section 3.7.1.

$\boldsymbol{S}$ and $\dot{\boldsymbol{E}}$ also have work conjugacy relationship.

3.6 Material models

Section 3.4 explained the kinematics, which is purely a geometric observation, and section 3.5 discussed stress and equilibrium, which can be derived from Newton's laws without consulting the actual material behavior. To complete the description of materials, the stress (forces) must be linked to strain (geometry). This link cannot be established without

acquiring experimental data from real materials. Then the stress-strain relationships or *constitutive laws* are derived.

3.6.1 Hyperelasticity and Elastic Potential Energy

The stress of an elastic material only depends on the current state of the deformation. Therefore, the first Piola-Kirchhoff tensor $\boldsymbol{P}$ at material point $\boldsymbol{X}$ is a function of the deformation gradient $\boldsymbol{F}$ at the point.

$$\boldsymbol{P}(\boldsymbol{X}) = \boldsymbol{P}\left(\boldsymbol{F}(\boldsymbol{X})\right) \tag{3.68}$$

If the material is hyperelastic, the work done by the stress during a deformation depends only on the initial state at time t_0 and the final configuration at time t. $\boldsymbol{P}$ is work conjugate with the rate of deformation gradient $\dot{\boldsymbol{F}}$, so $\boldsymbol{P} : \dot{\boldsymbol{F}}$ is the work done by the stress per unit time. The stored strain energy function or elastic potential Ψ can be written as

$$\Psi\left(\boldsymbol{F}(\boldsymbol{X})\right) = \int_{t_0}^{t} \boldsymbol{P}\left(\boldsymbol{F}(\boldsymbol{X})\right) : \dot{\boldsymbol{F}}\, dt; \quad \dot{\Psi} = \boldsymbol{P} : \dot{\boldsymbol{F}} \tag{3.69}$$

On the other hand

$$\dot{\Psi} = \frac{\partial \Psi}{\partial \boldsymbol{F}} : \dot{\boldsymbol{F}} \tag{3.70}$$

Combining the two formulas above leads to

$$\boldsymbol{P} = \frac{\partial \Psi}{\partial \boldsymbol{F}} \tag{3.71}$$

Ψ must be frame indifferent; a rigid motion should not change Ψ's value. Thus Ψ depends only on the rotation invariant components of $\boldsymbol{F}$

$$\Psi\left(\boldsymbol{F}(\boldsymbol{X})\right) = \Psi\left(\boldsymbol{C}(\boldsymbol{X})\right) \tag{3.72}$$

$\dot{\boldsymbol{E}} (= \frac{1}{2}\dot{\boldsymbol{C}})$ and $\boldsymbol{S}$ (the second Piola-Kirchhoff stress) are work conjugate.

$$\dot{\Psi} = \boldsymbol{S} : \dot{\boldsymbol{E}} \tag{3.73}$$

Also

$$\dot{\Psi} = \frac{\partial \Psi}{\partial \boldsymbol{E}} : \dot{\boldsymbol{E}} \tag{3.74}$$

Hence, the second Piola-Kirchhoff stress can be derived from the elastic potential function

$$\boldsymbol{S} = \frac{\partial \Psi}{\partial \boldsymbol{E}} \tag{3.75}$$

3.6.2 Elasticity Tensor

The linearization of the second Piola-Kirchhoff stress tensor is given as

$$\begin{aligned} DS_{IJ}[\boldsymbol{u}] &= \left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} S_{IJ}\left(\boldsymbol{E}\left(\phi + \varepsilon \boldsymbol{u}\right)\right) \\ &= \sum_{K,L=1}^{3} \frac{\partial S_{IJ}}{\partial E_{KL}} \left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} E_{KL}\left(\phi + \varepsilon \boldsymbol{u}\right) \\ &= \sum_{K,L=1}^{3} \frac{\partial S_{IJ}}{\partial E_{KL}} DE_{KL}[\boldsymbol{u}] \end{aligned} \tag{3.76}$$

$$D\boldsymbol{S}[\boldsymbol{u}] = \boldsymbol{\mathcal{C}} : D\boldsymbol{E}[\boldsymbol{u}] \tag{3.77}$$

where $\boldsymbol{\mathcal{C}}$ is the *Lagrangian* or *material elasticity tensor*, a symmetric fourth-order tensor whose components are given by

$$\mathcal{C}_{IJKL} = \frac{\partial S_{IJ}}{\partial E_{KL}} = \frac{4\partial^2 \Psi}{\partial C_{IJ} \partial C_{KL}} = \mathcal{C}_{KLIJ} \tag{3.78}$$

This formula will be used in linearization of the balance equation in section 3.7.1.

3.6.3 Isotropic Materials

If a material is isotropic, the principal directions of the stretch do not affect the amount of stored energy. For such a material, the potential energy function Ψ is a function of the invariants of $\boldsymbol{C}$:

$$\Psi(\boldsymbol{C}) = \Psi\left(I_C, II_C, III_C\right) \tag{3.79}$$

where invariants are defined as

$$\begin{aligned} I_C &= tr\boldsymbol{C} = \boldsymbol{C} : \boldsymbol{I} \\ II_C &= tr\boldsymbol{C}\boldsymbol{C} = \boldsymbol{C} : \boldsymbol{C} \\ III_C &= \det \boldsymbol{C} = J^2 \end{aligned} \tag{3.80}$$

The second Piola-Kirchhoff stress is conveniently computed as

$$\boldsymbol{S} = 2\frac{\partial \Psi}{\partial I_C}\boldsymbol{I} + 4\frac{\partial \Psi}{\partial II_C}\boldsymbol{C} + 2J^2 \frac{\partial \Psi}{\partial III_C}\boldsymbol{C}^{-1} \tag{3.81}$$

3.6.4 The Choice of Materials

The properties of organic tissues are being actively studied in biomechanics, and several models have been proposed based on stress-strain data obtained from in vivo and in vitro experiments. However, due to the limitations of measurement technology, those models have not been rigorously validated [Miller 2000].

Nevertheless, some of the important properties are known about most biological tissues. Those properties include 1) infinite energy with zero volume, 2) highly nonlinear stress-strain relationship, and 3) anisotropy. Every material used in this research has an elastic potential energy that tends to infinity as the volume compression ratio III_C tends to zero. This is not only a realistic assumption but also important in the numerical sense: this property effectively prevents the element reversal phenomenon in the finite element solution steps described in Section 4.7.2. The nonlinearity issue is addressed in Section 3.6.6, where the Veronda material is explained. Section 3.6.7 discusses a fiber-reinforced material—a typical anisotropic material.

3.6.5 Mooney-Rivlin Material

The first model is the Mooney-Rivlin material [Mooney 1940, Ciarlet 1998] whose elastic potential is

$$\Psi\left(I_C, II_C, III_C\right) = C_1(I_C - 3) + C_2(II_C - 3) + a(III_C - 1) - (C_1 + 2C_2 + a)\ln III_C \quad (3.82)$$

C_1 and C_2 are constants to control stiffness. a determines compressibility. This model is based on experimental data from different rubbers and exhibits a relatively linear strain-stress curve. The constant term, the energy at the resting shape, is chosen to be zero. This choice makes sense physically, but experimentally the energy itself is not measurable as an absolute value. In practice the elastic potential energy is used as a device to derive stress formula. Therefore, (3.82) can be simplified as

$$\Psi\left(I_C, II_C, III_C\right) = C_1 I_C + C_2 II_C + a III_C - (C_1 + 2C_2 + a)\ln III_C \quad (3.83)$$

By applying (3.81), the second Piola-Kirchhoff stress is obtained as

$$\boldsymbol{S} = 2C_1\boldsymbol{I} + 4C_2\boldsymbol{C} - \left(2C_1 + 4C_2 + 2a\left(1 - J^2\right)\right)\boldsymbol{C}^{-1} \quad (3.84)$$

3.6.6 Veronda Material

Biological tissues are often characterized by high nonlinearity: the stiffness (modulus) dramatically increases as they are stretched. The second model, the Veronda material, expresses this nonlinearity with an exponential function [Veronda 1970, Pioletti 1998]

$$\Psi\left(I_C, II_C, III_C\right) = \alpha \exp\left(\beta(I_C - 3)\right) - \frac{\alpha\beta}{2} II_C + a\, III_C - a \ln III_C \tag{3.85}$$

where α controls the overall stiffness and β governs the rate of stiffness increase. The second Piola-Kirchhoff stress for the material is

$$\boldsymbol{S} = 2\frac{\partial \Psi}{\partial I_C}\boldsymbol{I} + 4\frac{\partial \Psi}{\partial II_C}\boldsymbol{C} + 2J^2\frac{\partial \Psi}{\partial III_C}\boldsymbol{C}^{-1} \tag{3.86}$$

Using

$$\begin{cases} \dfrac{\partial \Psi}{\partial I_C} = \alpha\beta \exp\left(\beta(I_C - 3)\right) \\ \dfrac{\partial \Psi}{\partial II_C} = -\dfrac{\alpha\beta}{2} \\ \dfrac{\partial \Psi}{\partial III_C} = a - a\dfrac{1}{III_C} \end{cases} \tag{3.87}$$

the following is obtained

$$\begin{aligned} \boldsymbol{S} &= 2\left(\alpha\beta \exp\left(\beta(I_C - 3)\right)\right)\boldsymbol{I} - 2\left(\alpha\beta\right)\boldsymbol{C} + 2J^2\left(a - a\frac{1}{III_C}\right)\boldsymbol{C}^{-1} \\ &= 4\alpha\beta \exp\left(\beta(I_C - 3)\right)\boldsymbol{I} + 4\alpha\beta\boldsymbol{C} + 2a\left(J^2 - 1\right)\boldsymbol{C}^{-1} \end{aligned} \tag{3.88}$$

3.6.7 Fiber-Reinforced Material

In addition to the nonlinearity, many tissues show anisotropy due to their microscopic fiber structures [Hirokawa 2000, Klisch 1999]. The fiber-reinforcement model is

$$\Psi\left(\boldsymbol{C}\right) = \alpha \exp(\beta(\lambda_f{}^2 - 1)) - \alpha\beta\lambda_f{}^2 \tag{3.89}$$

where $\lambda_f{}^2 = \boldsymbol{d}_f{}^T \boldsymbol{C}\, \boldsymbol{d}_f$ is the fiber stretch along the fiber direction $\boldsymbol{d}_f$, and α and β are material constants similar to the ones in the Veronda model. With this model alone, the material is sensitive to the stretch in a single direction, but real fibrous materials contain not

only fibers but also a ground substance. Thus this model is blended with other isotropic ones such as the Veronda and Mooney-Rivlin materials to compose a more realistic model.

3.7 Linearized Equilibrium Equation

As derived in section 3.5.3 and 3.5.6 the Eulerian and Lagrangian principles of virtual work are

$$\delta W(\phi, \delta \boldsymbol{v}) = \int_v \boldsymbol{\sigma} : \delta \boldsymbol{d}\, dv - \int_v \boldsymbol{f} \cdot \delta \boldsymbol{v}\, dv - \int_{\partial v} \boldsymbol{t} \cdot \delta \boldsymbol{v}\, da = 0 \tag{3.90}$$

$$\delta W(\phi, \delta \boldsymbol{v}) = \int_V \boldsymbol{S} : \delta \dot{\boldsymbol{E}}\, dV - \int_V \boldsymbol{f}_0 \cdot \delta \boldsymbol{v}\, dV - \int_{\partial V} \boldsymbol{t}_0 \cdot \delta \boldsymbol{v}\, dA = 0 \tag{3.91}$$

Each term of these equations can be categorized into three parts

$$\delta W(\phi, \delta \boldsymbol{v}) = \delta W_{\text{int}}(\phi, \delta \boldsymbol{v}) - \delta W_{ext}^{f}(\phi, \delta \boldsymbol{v}) - \delta W_{ext}^{p}(\phi, \delta \boldsymbol{v}) = 0 \tag{3.92}$$

where $\delta W_{\text{int}}(\phi, \delta \boldsymbol{v})$ is the internal virtual work

$$\delta W_{\text{int}}(\phi, \delta \boldsymbol{v}) = \int_v \boldsymbol{\sigma} : \delta \boldsymbol{d}\, dv = \int_V \boldsymbol{S} : \delta \dot{\boldsymbol{E}}\, dV \tag{3.93}$$

$\delta W_{ext}^{f}(\phi, \delta \boldsymbol{v})$ is the virtual work done by the external volume force

$$\delta W_{ext}^{f}(\phi, \delta \boldsymbol{v}) = \int_v \boldsymbol{f} \cdot \delta \boldsymbol{v}\, dv = \int_V \boldsymbol{f}_0 \cdot \delta \boldsymbol{v}\, dV \tag{3.94}$$

and $\delta W_{ext}^{p}(\phi, \delta \boldsymbol{v})$ is the virtual work done by the external traction force

$$\delta W_{ext}^{p}(\phi, \delta \boldsymbol{v}) = \int_{\partial v} \boldsymbol{t} \cdot \delta \boldsymbol{v}\, da = \int_{\partial V} \boldsymbol{t}_0 \cdot \delta \boldsymbol{v}\, dA \tag{3.95}$$

As mentioned earlier, the numerical method used to solve the resulting nonlinear system is Newton-Raphson iteration. The method requires linearized formulae at each iteration. In the next two sections, in preparation for applying this method, the linearized forms of virtual work are derived.

3.7.1 Lagrangian Linearized Internal Virtual Work

First, the linearization of the Lagrangian internal virtual work is considered

$$
\begin{aligned}
D\delta W_{\text{int}}(\phi,\delta\boldsymbol{v})[\boldsymbol{u}] &= \int_V D\left(\delta\dot{\boldsymbol{E}} : \boldsymbol{S}\right)[\boldsymbol{u}]dV \\
&= \int_V \delta\dot{\boldsymbol{E}} : D\boldsymbol{S}[\boldsymbol{u}]dV + \int_V \boldsymbol{S} : D\delta\dot{\boldsymbol{E}}[\boldsymbol{u}]dV \\
&= \int_V \delta\dot{\boldsymbol{E}} : \boldsymbol{\mathcal{C}} : D\boldsymbol{E}[\boldsymbol{u}]dV + \int_V \boldsymbol{S} : D\delta\dot{\boldsymbol{E}}[\boldsymbol{u}]dV
\end{aligned} \tag{3.96}
$$

where $D\boldsymbol{E}[\boldsymbol{u}]$ is given by (3.36), and $\boldsymbol{\mathcal{C}}$ is defined by (3.78).

The expression for $D\delta\dot{\boldsymbol{E}}[\boldsymbol{u}]$ is derived as follows.

First, $\delta\dot{\boldsymbol{E}}$ is expanded as

$$
\delta\dot{\boldsymbol{E}} = \tfrac{1}{2}\left(\delta\dot{\boldsymbol{F}}^T\boldsymbol{F} + \boldsymbol{F}^T\delta\dot{\boldsymbol{F}}\right) \tag{3.97}
$$

$\delta\dot{\boldsymbol{F}}$ can be transformed to

$$
\delta\dot{\boldsymbol{F}} = \frac{\partial\delta\boldsymbol{v}}{\partial\boldsymbol{X}} = \nabla_0\delta\boldsymbol{v} \tag{3.98}
$$

Therefore, $\delta\dot{\boldsymbol{F}}$ is not a function of $\boldsymbol{u}$, so

$$
D\delta\dot{\boldsymbol{E}}[\boldsymbol{u}] = \tfrac{1}{2}\left(\left(\nabla_0\delta\boldsymbol{v}\right)^T D\boldsymbol{F}[\boldsymbol{u}] + \left(D\boldsymbol{F}[\boldsymbol{u}]\right)^T\nabla_0\delta\boldsymbol{v}\right) \tag{3.99}
$$

Noting that $D\boldsymbol{F}[\boldsymbol{u}] = \nabla_0\boldsymbol{u}$

$$
D\delta\dot{\boldsymbol{E}}[\boldsymbol{u}] = \tfrac{1}{2}\left(\left(\nabla_0\delta\boldsymbol{v}\right)^T\nabla_0\boldsymbol{u} + \left(\nabla_0\boldsymbol{u}\right)^T\nabla_0\delta\boldsymbol{v}\right) \tag{3.100}
$$

Therefore, noting the symmetry of $\boldsymbol{S}$, the Lagrangian linearized internal virtual work can be given as

$$
D\delta W_{\text{int}}(\phi,\delta\boldsymbol{v})[\boldsymbol{u}] = \int_V \delta\dot{\boldsymbol{E}} : \boldsymbol{\mathcal{C}} : D\boldsymbol{E}[\boldsymbol{u}]dV + \int_V \boldsymbol{S} : \left[\left(\nabla_0\boldsymbol{u}\right)^T\nabla_0\delta\boldsymbol{v}\right]dV \tag{3.101}
$$

A virtual time derivative is the directional derivative of virtual velocity, i.e., $\delta\dot{\boldsymbol{E}} = D\boldsymbol{E}[\delta\boldsymbol{v}]$.

Therefore, (3.101) can be rewritten in a symmetric form as

$$
D\delta W_{\text{int}}(\phi,\delta\boldsymbol{v})[\boldsymbol{u}] = \int_V D\boldsymbol{E}[\delta\boldsymbol{v}] : \boldsymbol{\mathcal{C}} : D\boldsymbol{E}[\boldsymbol{u}]dV + \int_V \boldsymbol{S} : \left[\left(\nabla_0\boldsymbol{u}\right)^T\nabla_0\delta\boldsymbol{v}\right]dV \tag{3.102}
$$

3.7.2 Eulerian Linearized Internal Virtual Work

The Lagrangian linearized internal virtual work derived in the previous section is also useful for formulating a finite element method. However, for some materials, a special

simplification is available for the Eulerian counterpart. For this reason, it is worthwhile to derive the Eulerian linearized virtual work.

Instead of linearizing the Eulerian principle of virtual work given by (3.60), the Eulerian linearized virtual work is derived from the Lagrangian linearized virtual work (3.102). The first term $D\boldsymbol{E}[\delta\boldsymbol{v}]:\boldsymbol{\mathcal{C}}:D\boldsymbol{E}[\boldsymbol{u}]dV$ is converted as

$$\begin{aligned}
&D\boldsymbol{E}[\delta\boldsymbol{v}]:\boldsymbol{\mathcal{C}}:D\boldsymbol{E}[\boldsymbol{u}]dV \\
&=\left(\boldsymbol{F}^T\delta\boldsymbol{d}\boldsymbol{F}\right):\boldsymbol{\mathcal{C}}:\left(\boldsymbol{F}^T\boldsymbol{\varepsilon}\boldsymbol{F}\right)\frac{1}{J}dv \\
&=\sum_{I,J,K,L=1}^{3}\left(\boldsymbol{F}^T\delta\boldsymbol{d}\boldsymbol{F}\right)_{IJ}:\mathcal{C}_{IJKL}:\left(\boldsymbol{F}^T\boldsymbol{\varepsilon}\boldsymbol{F}\right)_{KL}\frac{1}{J}dv \\
&=\sum_{I,J,K,L=1}^{3}\left(\sum_{i,j=1}^{3}F_{iI}\delta d_{ij}F_{jJ}\right)\mathcal{C}_{IJKL}\left(\sum_{k,l=1}^{3}F_{kK}\varepsilon_{kl}F_{lL}\right)\frac{1}{J}dv \\
&=\sum_{i,j,k,l=1}^{3}\delta d_{ij}\left(\frac{1}{J}\sum_{I,J,K,L=1}^{3}F_{iI}F_{jJ}F_{kK}F_{lL}\mathcal{C}_{IJKL}\right)\varepsilon_{kl}dv \\
&=\sum_{i,j,k,l=1}^{3}\delta d_{ij}\mathcal{C}_{ijkl}\varepsilon_{kl}dv \\
&=\delta\boldsymbol{d}:\boldsymbol{\mathcal{C}}:\boldsymbol{\varepsilon}\,dv
\end{aligned} \tag{3.103}$$

where $\boldsymbol{\mathcal{C}}$ is the *spatial elasticity tensor* defined as

$$\boldsymbol{\mathcal{C}}=\frac{1}{J}\phi_*[\boldsymbol{\mathcal{C}}];\quad \mathcal{C}_{ijkl}=\frac{1}{J}\sum_{I,J,K,L=1}^{3}F_{iI}F_{jJ}F_{kK}F_{lL}\mathcal{C}_{IJKL} \tag{3.104}$$

The second term $\boldsymbol{S}:\left[\left(\nabla_0\boldsymbol{u}\right)^T\nabla_0\delta\boldsymbol{v}\right]dV$ is transformed as

$$\begin{aligned}
&\boldsymbol{S}:\left[\left(\nabla_0\boldsymbol{u}\right)^T\nabla_0\delta\boldsymbol{v}\right]dV \\
&=J\phi_*^{-1}[\boldsymbol{\sigma}]:\left[\left(\left(\nabla\boldsymbol{u}\right)\boldsymbol{F}\right)^T\left(\left(\nabla\delta\boldsymbol{v}\right)\boldsymbol{F}\right)\right]\frac{1}{J}dv \\
&=\boldsymbol{F}^{-1}\boldsymbol{\sigma}\boldsymbol{F}^{-T}:\left[\boldsymbol{F}^T\left(\nabla\boldsymbol{u}\right)^T\left(\nabla\delta\boldsymbol{v}\right)\boldsymbol{F}\right]dv,
\end{aligned}$$

which since $\left(\boldsymbol{F}^{-1}\boldsymbol{\sigma}\boldsymbol{F}^{-T}\right)^T=\boldsymbol{F}^{-1}\boldsymbol{\sigma}\boldsymbol{F}^{-T}$ is equal to

$$\mathrm{tr}\left(\boldsymbol{F}^{-1}\boldsymbol{\sigma}\boldsymbol{F}^{-T}\boldsymbol{F}^T\left(\nabla\boldsymbol{u}\right)^T\left(\nabla\delta\boldsymbol{v}\right)\boldsymbol{F}\right)dv$$

Using the relationship $\mathrm{tr}\left(\boldsymbol{F}^{-1}A\boldsymbol{F}\right)=\boldsymbol{F}^{-1}:\boldsymbol{F}^T\boldsymbol{A}^T=\mathrm{tr}(\boldsymbol{F}^{-T}\boldsymbol{F}^T\boldsymbol{A}^T)=\mathrm{tr}\left(\boldsymbol{A}\right)$, the formula above is further simplified as

$$\begin{aligned}
&\mathrm{tr}\left(\boldsymbol{F}^{-1}\boldsymbol{\sigma}\left(\nabla\boldsymbol{u}\right)^{T}\left(\nabla\delta\boldsymbol{v}\right)\boldsymbol{F}\right)dv \\
&=\mathrm{tr}\left(\boldsymbol{\sigma}\left(\nabla\boldsymbol{u}\right)^{T}\left(\nabla\delta\boldsymbol{v}\right)\right)dv \\
&=\boldsymbol{\sigma}:\left[\left(\nabla\boldsymbol{u}\right)^{T}\nabla\delta\boldsymbol{v}\right]dv
\end{aligned}$$

As a result, the Eulerian linearized internal virtual work is obtained as

$$D\delta W_{\mathrm{int}}\left(\phi,\delta\boldsymbol{v}\right)\left[\boldsymbol{u}\right]=\int_{v}\delta\boldsymbol{d}:\boldsymbol{\mathcal{C}}:\boldsymbol{\varepsilon}\,dv+\int_{v}\boldsymbol{\sigma}:\left[\left(\nabla\boldsymbol{u}\right)^{T}\nabla\delta\boldsymbol{v}\right]dv \qquad (3.105)$$

This concludes the derivations of principles of virtual work (weak form differential equations) and their linearized forms. Now the discretization of those forms can be discussed.

4 FINITE ELEMENT METHOD

4.1 Overview

In the previous chapter, the equilibrium equation, in terms of a spatial (3.60) or a material (3.67) description, was derived. For example, (3.67) can be expressed as

$$\delta W = \int_V \boldsymbol{S} : \delta \dot{\boldsymbol{E}}\, dV - \int_V \boldsymbol{f}_0 \cdot \delta \boldsymbol{v}\, dV - \int_{\partial V} \boldsymbol{t}_0 \cdot \delta \boldsymbol{v}\, dA = 0 \qquad (4.1)$$

To simulate deformation, a configuration (a particle position $\boldsymbol{x}$ or a displacement $\boldsymbol{u}$) that satisfies (4.1) for any $\delta \boldsymbol{v}$, must be found. But, because of its complex boundary conditions and nonlinearity, (4.1) cannot be solved analytically. Instead, the equilibrium equation must be discretized resulting in a series of algebraic equations. These algebraic equations can be used to obtain an approximate solution. The finite element method is used as the discretization method in this dissertation work.

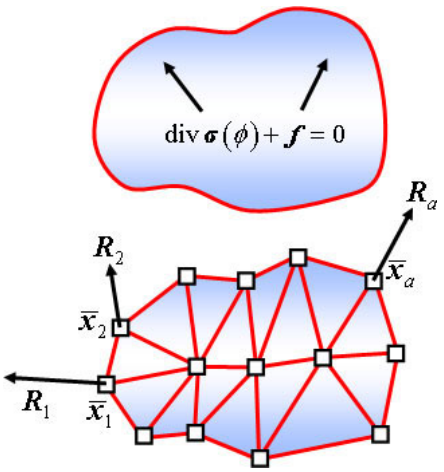

Figure 11: Finite Element Method. A continuous object is discretized by the finite element method. The partial differential equation, $\operatorname{div}\boldsymbol{\sigma}(\phi) + \boldsymbol{f} = 0$ is replaced by algebraic equations in terms of the nodal residual forces $R_1, R_2, \cdots$.

The first step of the finite element method is discretizing the kinematics. Then the equilibrium equation is discretized by substituting the original quantities with discretized versions. To apply the Newton-Raphson method, the linearized equilibrium equations should be discretized in the same manner.

4.2 Discretized Kinematics

In continuum mechanics, an object contains an infinite number of particles. Thus the position $\boldsymbol{x}$ (or a displacement $\boldsymbol{u}$) has infinite degrees of freedom. The principal idea of discretization is to approximate $\boldsymbol{x}$ by an interpolation of just a handful of numbers.

In finite element methods, the object is partitioned into *elements*, and each element has a few *nodal points* or *nodes*. To solve problems in solid mechanics, it is often convenient to define this partitioning or *meshing* in the material configuration. Then each nodal point is attached to a particle, so boundary is unchanged by deformation. The movement of a particle is obtained as an interpolation of the movements of the nodal points:

$$\boldsymbol{x}(\boldsymbol{X},t) = \sum_{a=1}^{n} N_a(\boldsymbol{X})\,\bar{\boldsymbol{x}}_a(t) \tag{4.2}$$

where $\bar{\boldsymbol{x}}_a(t)$ is the current position of the nodal points, n is the number of nodal points, $N_a(\boldsymbol{X})$ is the *shape function*, which defines the influence of the node a at the material point $\boldsymbol{X}$. Since the shapes of the elements are not uniform, $N_a(\boldsymbol{X})$ must be defined differently for each element.

Isoparametric elements share a single 3D coordinate system $\boldsymbol{\xi} = [\xi_1 \quad \xi_2 \quad \xi_3]$ and use a single set of shape functions $N_a(\boldsymbol{\xi})$ where $\boldsymbol{\xi}$ is defined as a function of $\boldsymbol{X}$, thus

$$\boldsymbol{x}(\boldsymbol{X},t) = \sum_{a=1}^{n} N_a\big(\boldsymbol{\xi}(\boldsymbol{X})\big)\,\bar{\boldsymbol{x}}_a(t) \tag{4.3}$$

or simply

$$\boldsymbol{x} = \sum_{a=1}^{n} N_a\,\bar{\boldsymbol{x}}_a(t) \tag{4.4}$$

$\boldsymbol{\xi}(\boldsymbol{X})$ can be obtained as follows.

Let the particle corresponding to the nodal point be $\bar{\boldsymbol{X}}_a = \bar{\boldsymbol{x}}_a(t)$. In the initial configuration,

$$\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{\xi}) = \sum_{a=1}^{n} N_a(\boldsymbol{\xi})\bar{\boldsymbol{X}}_a \tag{4.5}$$

$\boldsymbol{\xi}(\boldsymbol{X})$ is then found as the inverse of $\boldsymbol{X}(\boldsymbol{\xi})$.

By differentiating both sides of (4.4) by time, the velocity is obtained as

$$\boldsymbol{v} = \sum_{a=1}^{n} N_a \bar{\boldsymbol{v}}_a(t) \tag{4.6}$$

The displacement is also interpolated as

$$\boldsymbol{u} = \sum_{a=1}^{n} N_a \bar{\boldsymbol{u}}_a(t) \tag{4.7}$$

where $\bar{\boldsymbol{u}}_a$ is the a^{th} particle's displacement.

The deformation gradient is

$$\boldsymbol{F} = \sum_{a=1}^{n} \bar{\boldsymbol{x}}_a \otimes \nabla_0 N_a \tag{4.8}$$

where $\nabla_0 N_a$ is

$$\nabla_0 N_a = \frac{\partial N_a}{\partial \boldsymbol{X}} = \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{X}}\frac{\partial N_a}{\partial \boldsymbol{\xi}} = \left(\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{\xi}}\right)^{-T}\frac{\partial N_a}{\partial \boldsymbol{\xi}}; \qquad \frac{\partial \boldsymbol{X}}{\partial \boldsymbol{\xi}} = \sum_{a=1}^{n} \bar{\boldsymbol{X}}_a \otimes \nabla_\xi N_a \tag{4.9}$$

where ∇_ξ denotes $\dfrac{\partial}{\partial \boldsymbol{\xi}}$.

By introducing (4.6) into (3.50),

$$\boldsymbol{d} = \frac{1}{2}\sum_{a=1}^{n}\left(\bar{\boldsymbol{v}}_a \otimes \nabla N_a + \nabla N_a \otimes \bar{\boldsymbol{v}}_a\right) \tag{4.10}$$

$$\delta\boldsymbol{d} = \frac{1}{2}\sum_{a=1}^{n}\left(\delta\bar{\boldsymbol{v}}_a \otimes \nabla N_a + \nabla N_a \otimes \delta\bar{\boldsymbol{v}}_a\right) \tag{4.11}$$

Similarly, by introducing (4.7) into (3.35),

$$\boldsymbol{\varepsilon} = \frac{1}{2}\sum_{a=1}^{n}\left(\bar{\boldsymbol{u}}_a \otimes \nabla N_a + \nabla N_a \otimes \bar{\boldsymbol{u}}_a\right) \tag{4.12}$$

In the above three equations, ∇N_a is obtained as

$$\nabla N_a = \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}\right)^{-T} \frac{\partial N_a}{\partial \boldsymbol{\xi}}; \qquad \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}} = \sum_{a=1}^{n} \bar{\boldsymbol{x}}_a \otimes \nabla_{\xi} N_a \tag{4.13}$$

4.3 Tetrahedral Element

In this dissertation, the linear tetrahedral element is used. A tetrahedral element has 4 nodal points located at vertices of a tetrahedron. Four shape functions are defined as

$$\begin{aligned} N_1(\xi_1,\xi_2,\xi_3) &= \xi_1 \\ N_2(\xi_1,\xi_2,\xi_3) &= \xi_2 \\ N_3(\xi_1,\xi_2,\xi_3) &= \xi_3 \\ N_4(\xi_1,\xi_2,\xi_3) &= 1-\xi_1-\xi_2-\xi_3 \end{aligned} \tag{4.14}$$

and their derivatives are given as

$$\begin{aligned} &\nabla_{\xi} N_1(\xi_1,\xi_2,\xi_3) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \nabla_{\xi} N_2(\xi_1,\xi_2,\xi_3) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \nabla_{\xi} N_3(\xi_1,\xi_2,\xi_3) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \\ &\nabla_{\xi} N_4(\xi_1,\xi_2,\xi_3) = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \end{aligned} \tag{4.15}$$

The approximation of $\boldsymbol{x}(\boldsymbol{X})$ using this shape function belongs to Hilbert space H^1 (Sobolev space with L^2 norm). Therefore, it is sufficiently smooth to solve second order differential equations, which include all PDEs used in this work.

4.4 Mesh Generation

Mesh generation or *meshing* is the task of partitioning the domains of given objects into elements. Numerous mesh generation techniques have been proposed [George 1996]. Describing each technique is beyond the scope of this dissertation. Nevertheless, meshing has great impacts on the discretization errors of finite element simulations. Adaptive meshing combined with accurate error estimation is left for future work.

4.5 Discretized Equilibrium Equations

The equilibrium equation (3.60) is

$$\delta W(\phi, \delta \boldsymbol{v}) = \int_v \boldsymbol{\sigma} : \delta \boldsymbol{d}\, dv - \int_v \boldsymbol{f} \cdot \delta \boldsymbol{v}\, dv - \int_{\partial v} \boldsymbol{t} \cdot \delta \boldsymbol{v}\, da$$

The goal of discretization is to derive the equivalent nodal forces. The force at a node can be found by examining the virtual work caused by the virtual velocity of a single nodal point. The total virtual work is the summation of the contributions from all the elements, which is

$$\begin{aligned} \delta W^{(e)}(\phi, N_a \delta \overline{\boldsymbol{v}}_a) = & \int_{v^{\{e\}}} \boldsymbol{\sigma} : (\delta \overline{\boldsymbol{v}}_a \otimes \nabla N_a) dv \\ & - \int_{v^{(e)}} \boldsymbol{f} \cdot (N_a \delta \overline{\boldsymbol{v}}_a)\, dv - \int_{\partial v^{(e)}} \boldsymbol{t} \cdot (N_a \delta \overline{\boldsymbol{v}}_a)\, da \end{aligned} \tag{4.16}$$

where $\delta \overline{\boldsymbol{v}}_a$ is the virtual velocity of the a^{th} nodal point and e denotes the e^{th} element, which contains the a^{th} nodal point. Equation (4.16) can be further simplified as

$$\delta W^{(e)}(\phi, N_a \delta \overline{\boldsymbol{v}}_a) = \delta \overline{\boldsymbol{v}}_a \cdot \left(\int_{v^{\{e\}}} \boldsymbol{\sigma} \nabla N_a dv - \int_{v^{(e)}} N_a \boldsymbol{f}\, dv - \int_{\partial v^{(e)}} N_a \boldsymbol{t}\, da \right) \tag{4.17}$$

or

$$\delta W^{(e)}(\phi, N_a \delta \overline{\boldsymbol{v}}_a) = \delta \overline{\boldsymbol{v}}_a \cdot (\boldsymbol{T}_a^{(e)} - \boldsymbol{F}_a^{(e)}) \tag{4.18}$$

where

$$\boldsymbol{T}_a^{(e)} = \int_{v^{(e)}} \boldsymbol{\sigma} \nabla N_a dv; \qquad T_{a,i}^{(e)} = \sum_{j=1}^{3} \int_{v^{\{e\}}} \sigma_{ij} \frac{\partial N_a}{\partial x_j} dv \tag{4.19}$$

is the internal nodal equivalent force and

$$\boldsymbol{F}_a^{(e)} = \int_{v^{(e)}} N_a \boldsymbol{f}\, dv + \int_{\partial v^{(e)}} N_a \boldsymbol{t}\, da \tag{4.20}$$

is the external nodal equivalent force. In this dissertation, there are two external forces that are considered. One is the gravity force, which is included in the first term $\int_{v^{(e)}} N_a \boldsymbol{f}\, dv$, the other is the contact force described in Chapter 5. A nodal gravity force is

$$\int_{V^{(e)}} N_a \begin{bmatrix} 0 \\ 0 \\ -\rho \end{bmatrix} dV \tag{4.21}$$

where ρ is mass density in the reference configuration.

The total force at the a^{th} nodal point is the summation of contributions from all elements that contains the nodal point a,

$$\delta W(\phi, N_a \delta \overline{\boldsymbol{v}}_a) = \sum_{a \in e} \delta W^{(e)}(\phi, N_a \delta \overline{\boldsymbol{v}}_a) = \delta \overline{\boldsymbol{v}}_a \cdot (\boldsymbol{T}_a - \boldsymbol{F}_a) \tag{4.22}$$

where the total equivalent nodal forces are

$$\boldsymbol{T}_a = \sum_{a \in e} \boldsymbol{T}^{(e)}; \quad \boldsymbol{F}_a = \sum_{a \in e} \boldsymbol{F}^{(e)} \tag{4.23}$$

The discretized principle of virtual work is the summation of the contributions from all the nodal points,

$$\delta W(\phi, \delta \boldsymbol{v}) = \sum_{a=1}^{n} \delta W(\phi, N_a \delta \overline{\boldsymbol{v}}_a) = \sum_{a=1}^{n} \delta \overline{\boldsymbol{v}}_a \cdot (\boldsymbol{T}_a - \boldsymbol{F}_a) = 0 \tag{4.24}$$

This relationship should hold for arbitrary $\overline{\boldsymbol{v}}_a$, so

$$\boldsymbol{R}_a = \boldsymbol{T}_a - \boldsymbol{F}_a = 0 \tag{4.25}$$

where $\boldsymbol{R}_a$ is called the *nodal residual force*.

It is convenient to concatenate all the nodal forces into a single array

$$\mathbf{T} = \begin{bmatrix} \boldsymbol{T}_1 \\ \boldsymbol{T}_2 \\ \vdots \\ \boldsymbol{T}_n \end{bmatrix}; \quad \mathbf{F} = \begin{bmatrix} \boldsymbol{F}_1 \\ \boldsymbol{F}_2 \\ \vdots \\ \boldsymbol{F}_n \end{bmatrix}; \quad \mathbf{R} = \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \\ \vdots \\ \boldsymbol{R}_n \end{bmatrix} \tag{4.26}$$

Using these arrays, the discretized virtual equation is written as

$$\delta W(\phi, \delta \boldsymbol{v}) = \delta \mathbf{v}^T \mathbf{R} = \delta \mathbf{v}^T (\mathbf{T} - \mathbf{F}) = 0 \tag{4.27}$$

where $\delta \mathbf{v}^T$ is the virtual velocity vector $\begin{bmatrix} \delta v_1^T & \delta v_2^T & \dots & \delta v_n^T \end{bmatrix}$.

Similarly the current position of the nodal points called the mesh configuration is built into an array

$$\mathbf{x} = \begin{bmatrix} \overline{\boldsymbol{x}}_1 \\ \overline{\boldsymbol{x}}_2 \\ \vdots \\ \overline{\boldsymbol{x}}_n \end{bmatrix} \tag{4.28}$$

The nodal forces are nonlinear equations of the nodal positions, so a system of equations of the form is obtained:

$$\mathbf{R}(\mathbf{x}) = \mathbf{T}(\mathbf{x}) - \mathbf{F}(\mathbf{x}) = 0 \tag{4.29}$$

4.6 Discretization of the Linearized Equilibrium Equations

To solve the nonlinear system (4.29), the Newton-Raphson iteration is employed. The Newton-Raphson iteration requires the computation of a tangent matrix

$$\mathbf{K} = \frac{\partial \mathbf{R}(\mathbf{x})}{\partial \mathbf{x}} \tag{4.30}$$

This tangent matrix is the slope of the force with respect to the displacement. Therefore, it is also called a *stiffness matrix*. This can be written as an $n \times n$ matrix

$$\mathbf{K} = \begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{12} & \cdots & \boldsymbol{K}_{1n} \\ \boldsymbol{K}_{21} & \boldsymbol{K}_{22} & & \vdots \\ \vdots & & \ddots & \\ \boldsymbol{K}_{n1} & \cdots & & \boldsymbol{K}_{nn} \end{bmatrix} \tag{4.31}$$

where each element is a 3×3 matrix

$$\boldsymbol{K}_{ab} = \frac{\partial \boldsymbol{R}_a}{\partial \boldsymbol{x}_b}; \quad \left[\boldsymbol{K}_{ab}\right]_{ij} = \frac{\partial R_{ai}}{\partial x_{bj}} \tag{4.32}$$

$\boldsymbol{K}_{ab}$ describes how the force on nodal point a changes as nodal point b moves. Thus this matrix "links" node a and b.

The nodal force itself is found as the virtual work caused by the virtual velocity of node a. The derivative of this virtual work in the direction induced by movement of node b is

$$D\delta W(\phi, N_a \delta \overline{\boldsymbol{v}}_a)[N_b \overline{\boldsymbol{u}}_b] = D\left(\delta \overline{\boldsymbol{v}}_a \cdot \boldsymbol{R}_a\right)[N_b \overline{\boldsymbol{u}}_b] = \delta \overline{\boldsymbol{v}}_a \cdot \frac{\partial \boldsymbol{R}_a}{\partial \boldsymbol{x}_b} \overline{\boldsymbol{u}}_b \tag{4.33}$$

Thus $\mathbf{K} = \partial \mathbf{R}(\mathbf{x}) / \partial \mathbf{x}$ can be found by examining the directional derivative of virtual work. The virtual work consists of the internal and external parts as

$$\delta W(\phi, \delta \boldsymbol{v})[\boldsymbol{u}] = \delta W_{\text{int}}(\phi, \delta \boldsymbol{v})[\boldsymbol{u}] - \delta W_{\text{ext}}(\phi, \delta \boldsymbol{v})[\boldsymbol{u}] \tag{4.34}$$

Therefore, the directional derivative can be computed separately:

$$D\delta W(\phi, \delta \boldsymbol{v})[\boldsymbol{u}] = D\delta W_{\text{int}}(\phi, \delta \boldsymbol{v})[\boldsymbol{u}] - D\delta W_{\text{ext}}(\phi, \delta \boldsymbol{v})[\boldsymbol{u}] \tag{4.35}$$

The internal virtual work is again divided into the constitutive term and the initial stress term as,

$$
\begin{aligned}
&D\delta W_{\text{int}}(\phi,\delta\boldsymbol{v})[\boldsymbol{u}] = D\delta W_{\boldsymbol{\mathcal{C}}}(\phi,\delta\boldsymbol{v})[\boldsymbol{u}] + D\delta W_{\sigma}(\phi,\delta\boldsymbol{v})[\boldsymbol{u}] \\
&= \int_v \delta\boldsymbol{d} : \boldsymbol{\mathcal{C}} : \boldsymbol{\varepsilon}\, d\boldsymbol{v} + \int_v \boldsymbol{\sigma} : [(\nabla\boldsymbol{u})^T \nabla\delta\boldsymbol{v}]\, dv
\end{aligned}
\tag{4.36}
$$

Thus the internal part of the stiffness matrix $\mathbf{K}$ is made of the constitutive contribution and the initial stress contribution.

4.6.1 Constitutive Component

The constitutive contribution to $\boldsymbol{K}_{ab}$ from each element is derived as follows:

$$
\begin{aligned}
&D\delta W^{(e)}_{\boldsymbol{\mathcal{C}}}(\phi, N_a\delta\overline{\boldsymbol{v}}_a)[N_b\overline{\boldsymbol{u}}_b] \\
&= \int_{v^{(e)}} \frac{1}{2}\left(\delta\overline{\boldsymbol{v}}_a \otimes \nabla N_a + \nabla N_a \otimes \delta\overline{\boldsymbol{v}}_a\right) : \boldsymbol{\mathcal{C}} : \frac{1}{2}\left(\overline{\boldsymbol{u}}_a \otimes \nabla N_a + \nabla N_a \otimes \overline{\boldsymbol{u}}_a\right) dv
\end{aligned}
\tag{4.37}
$$

The constitutive component $\boldsymbol{K}^{(e)}_{\boldsymbol{\mathcal{C}},ab}$ is

$$
D\delta W^{(e)}_{\boldsymbol{\mathcal{C}}}(\phi, N_a\delta\overline{\boldsymbol{v}}_a)[N_b\overline{\boldsymbol{u}}_b] = \delta\overline{\boldsymbol{v}}_a \cdot \boldsymbol{K}^{(e)}_{\boldsymbol{\mathcal{C}},ab}\overline{\boldsymbol{u}}_b \tag{4.38}
$$

By careful comparison between two equations above,

$$
\left[\boldsymbol{K}^{(e)}_{\boldsymbol{\mathcal{C}},ab}\right]_{ij} = \int_{v^{(e)}} \sum_{k,l=1}^{3} \frac{\partial N_a}{\partial x_k} \mathcal{C}^{sym}_{ikjl} \frac{\partial N_b}{\partial x_l} dv \tag{4.39}
$$

where the symmetrized constitutive tensor is

$$
\mathcal{C}^{sym}_{ikjl} = \frac{1}{4}(\mathcal{C}_{ikjl} + \mathcal{C}_{iklj} + \mathcal{C}_{kijl} + \mathcal{C}_{kilj}) \tag{4.40}
$$

4.6.2 Initial Stress Component

The gradients of $\delta\boldsymbol{v}$ and $\boldsymbol{u}$ are

$$
\nabla\delta\boldsymbol{v} = \sum_{a=1}^{n} \delta\boldsymbol{v}_a \otimes \nabla N_a \tag{4.41}
$$

$$
\nabla\boldsymbol{u} = \sum_{b=1}^{n} \boldsymbol{u}_b \otimes \nabla N_b \tag{4.42}
$$

By introducing these to (4.36),

$$
\begin{aligned}
D\delta W_{\sigma}^{(e)}(\phi, N_a\delta\bar{\boldsymbol{v}}_a)[N_b\bar{\boldsymbol{u}}_b] &= \int_{v^{(e)}} \boldsymbol{\sigma} : \left[\left(\nabla\bar{\boldsymbol{u}}_b\right)^T \nabla\delta\bar{\boldsymbol{v}}_a \right] dv \\
&= \int_{v^{(e)}} \boldsymbol{\sigma} : \left[\left(\delta\bar{\boldsymbol{v}}_a \cdot \bar{\boldsymbol{u}}_b\right) \nabla N_b \otimes \nabla N_a \right] dv \\
&= \left(\delta\bar{\boldsymbol{v}}_a \cdot \bar{\boldsymbol{u}}_b\right) \int_{v^{(e)}} \nabla N_a \cdot \boldsymbol{\sigma} \nabla N_b dv
\end{aligned}
\tag{4.43}
$$

On the other hand, using the initial stress component $\boldsymbol{K}_{\sigma,ab}^{(e)}$ the directional derivative is

$$
D\delta W_{\sigma}(\phi, N_a\delta\boldsymbol{v}_a)[N_b\boldsymbol{u}_b] = \delta\boldsymbol{v}_a \cdot \boldsymbol{K}_{\sigma,ab}^{(e)}\boldsymbol{u}_b \tag{4.44}
$$

Comparison of above forms reveals

$$
\boldsymbol{K}_{\boldsymbol{\sigma},ab}^{(e)} = \int_{v^{(e)}} (\nabla N_a \cdot \boldsymbol{\sigma} \nabla N_b) \boldsymbol{I} dv \,, \tag{4.45}
$$

or, in indicial form,

$$
\left[\boldsymbol{K}_{\boldsymbol{\sigma},ab}^{(e)} \right]_{ij} = \int_{v^{(e)}} \sum_{k,l=1}^{3} \frac{\partial N_a}{\partial x_k} \sigma_{kl} \frac{\partial N_b}{\partial x_l} \delta_{ij} dv \tag{4.46}
$$

4.6.3 External Force Components

As seen in (4.21), the gravity forces are independent of deformation. Therefore, the gravity forces do not contribute to the stiffness matrix. Due to the complexity of contact forces, their stiffness matrix contribution is not considered in this work.

4.6.4 Total stiffness matrix

Finally the total stiffness matrix is the summation of all components described above:

$$
\boldsymbol{K}_{ab}^{(e)} = \boldsymbol{K}_{\boldsymbol{\sigma},ab}^{(e)} + \boldsymbol{K}_{\boldsymbol{\sigma},ab}^{(e)} \tag{4.47}
$$

4.7 Solution of the nonlinear system

By solving (4.29), the mesh configuration $\mathbf{x}$ in an equilibrium state is obtained. Since (4.29) is a large nonlinear system, a sophisticated solver should be used. As mentioned earlier, Newton-Raphson iteration is employed for this purpose. In this section, various aspects of the nonlinear solver are examined.

4.7.1 Selecting a search direction

Nearly all solution methods start with an initial guess and proceed in a given search direction in a step-by-step manner. Finding the proper direction in the high-dimensional search space is critical for the algorithm's efficiency. Several different methods exist to determine the best search direction.

The maximum gradient descent method chooses the direction of the forces (i.e., the negative of the residual $\mathbf{R}(\mathbf{x})$) for each step. This method turns out to be very slow because it takes many steps for a local force to propagate through the entire mesh. Furthermore it is impossible for this method to predict rapid force changes caused by the deformation of objects.

On the other hand, the Newton-Raphson method uses the derivative of the forces (i.e., the stiffness matrix), which provides information about how the forces vary as a function of deformation. Each Newton step consists of the computation of the residual, computation of the stiffness matrix, and solution of a linear system. The process continues until the residual drops below a given tolerance.

4.7.2 Avoiding illegal steps

There are two questions that remain to be answered: how to obtain the initial guess and how far to proceed in a selected search direction. If there are no displacement boundary conditions, the object is deformed only by external forces. In this case, an obvious choice of the initial guess is the initial shape, i.e., $\mathbf{x} = 0$. But if the displacement boundary conditions are specified, some components of $\mathbf{x}$ are externally given. In such cases, $\mathbf{x}$ initially contains zeros for free components and final values for constrained components. The corresponding mesh may contain a tetrahedral element whose orientation is reversed, resulting in an illegal configuration. For this reason, constrained components must be gradually incremented, and as soon as an illegal configuration is detected, the step size must be reduced (adaptive incremental loading). Each incremental step performs a Newton iteration, and the solution is cascaded into the next step. After the second step, the initial guess is computed by extrapolating the previous two solutions (two-point predictor).

Given an initial guess, one must determine how far to proceed in a given search direction. If $\mathbf{R}(\mathbf{x})$ is smooth, and the initial guess is close to the solution, a full step towards the solution of the linear system can safely be taken. However, the nonlinearity of the system often causes a full step to lead to divergence. A full Newton step can also bring the mesh into an illegal configuration. My method checks if the full Newton step is legal and if the residual decreases. If both of these are true, the step is taken. If not, the step size is halved until the two conditions are satisfied. The scaling value of the step size is called a *damping* factor; hence this method is sometimes called *damped Newton method*. This line search strategy greatly improves the robustness of the method.

The resulting algorithm can be summarized as three nested loops as shown in Figure 12.

Loop1: Adaptive Incremental Loading
  2-Point Prediction
  Loop2: Newton Iteration, $i = 0,1,2,\cdots$
    Compute $\mathbf{K}$ and $\mathbf{R}$
    if $\left\|\mathbf{R}\left(\mathbf{x}^{(i)}\right)\right\| < \varepsilon$
      terminate Newton Iteration
    $\Delta\mathbf{x} = -\mathbf{K}\left(\mathbf{x}^{(i)}\right)^{-1}\mathbf{R}\left(\mathbf{x}^{(i)}\right)$ // Linear System Solution
    $\mathbf{x}^{(\text{newton})} = \mathbf{x}^{(i)} + \Delta\mathbf{x}$ // Full Newton Step
    Loop3: Line Search, $\alpha = 1, ½, ¼, \cdots$
      if $\left\|\mathbf{R}\left((1-\alpha)\mathbf{x}^{(i)} + \alpha\mathbf{x}^{(\text{newton})}\right)\right\| < \left\|\mathbf{R}\left(\mathbf{x}^{(i)}\right)\right\|$
        $\mathbf{x}^{(i+1)} = (1-\alpha)\mathbf{x}^{(i)} + \alpha\mathbf{x}^{(\text{newton})}$
        terminate Line Search

Figure 12: Algorithm of Nonlinear System Solver

4.7.3 Linear system solution

Newton iteration depends on the stable solution of linear systems. In the linear system solution, the accuracy of the solution is traded for speed. Since the degree of nonlinearity of

the equation is high, the residual is not greatly reduced by a single Newton iteration (one iteration of Loop2 in **Figure 12)**. Consequently, computing an exact solution for each linear system does not improve the convergence rate of the Newton iteration. Given a large error tolerance, iterative linear system solution methods converge more quickly than direct methods. For this reason, my method uses an iterative linear system solver.

The linear systems constructed in my finite element method are symmetric, sparse, and usually positive definite. Around a bifurcation point, however, the matrix of the system is sometimes not only indefinite but also nearly singular. The (bi)conjugate gradient method tends to behave in an erratic manner. I found that an implementation of the Generalized Minimum Residual method (GMRES) with diagonal preconditioning [Seager 1988] is both efficient and stable. The iteration is terminated when either the residual reduces to one tenth or a predetermined iteration limit is reached. This strategy keeps the computation time for solving the linear systems relatively low without slowing down the convergence of the Newton iteration.

I made another adjustment to the above technique to prevent failure in the linear system solver. Each node is assigned a "viscosity" proportional to the sizes of the surrounding elements. Let $\dot{\mathbf{x}}$ be the node velocity vector, let $\mathbf{D}$ be the viscosity coefficient matrix, and let $\boldsymbol{D}_a$ be the 3×3 diagonal viscosity matrix for a^{th} node. The viscous force is

$$\mathbf{D}\dot{\mathbf{x}} = \begin{bmatrix} \boldsymbol{D}_1 & & & 0 \\ & \boldsymbol{D}_2 & & \\ & & \ddots & \\ 0 & & & \boldsymbol{D}_n \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \\ \vdots \\ \dot{\boldsymbol{x}}_n \end{bmatrix} \tag{4.48}$$

The balance equation with this viscous force is

$$\mathbf{R}(\mathbf{x}) + \mathbf{D}\dot{\mathbf{x}} = 0 \tag{4.49}$$

This equation is linearized as follows:

$$\begin{aligned} &\mathbf{R}\left(\mathbf{x}^{(i)} + \left(\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}\right)\right) + \mathbf{D}\dot{\mathbf{x}}^{(i+1)} = 0 \\ &\mathbf{R}\left(\mathbf{x}^{(i)}\right) + \mathbf{K}\left(\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}\right) + \mathbf{D}\frac{\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}}{\Delta t} = 0 \end{aligned} \tag{4.50}$$

where i is the index of the Newton iteration, Δt is the "time step" associated with each Newton iteration, Δt is usually set to 1. The net effect to the original Newton iteration is simply adding viscous factors to the diagonal elements of $\mathbf{K}$:

$$\mathbf{K} \to \mathbf{K} + \frac{1}{\Delta t}\mathbf{D} \tag{4.51}$$

Around a bifurcation point, the stiffness matrix $\mathbf{K}$ becomes indefinite or, in the worst case, singular. The viscous component tends to make the matrix nonsingular and positive definite and gives the algorithm the tendency to pick an energy-minimizing solution at the bifurcation point, i.e., to prefer a stable equilibrium point.

4.8 Dynamic and quasistatic analysis

My method was extended to quasistatic (i.e., inertia ignored) and dynamic analysis. The equilibrium equation now includes inertial and viscous forces. Time derivatives are discretized by the implicit Euler scheme. In **Figure 12**, the adaptive incremental loading steps are replaced with adaptive time steps. The Newton iteration solves for nodal velocities. The velocities are then used to update nodal positions.

5 THE CONTACT PROBLEM

5.1 Overview

This chapter is dedicated to the contact problem, which is the main topic of this dissertation. Section 5.2 explains the kinematics and balance equations of the contact problem using Signorini's problem where one side of the contact is stationary. Then section 5.3 describes more general cases that include self-contact. This section pays particularly close attention to various geometric situations where the direction of contact forces is sensitive to the object deformation or are too ambiguous to define mathematically. In previous literature, the discontinuity of boundary normals has been blamed for the ambiguity and sensitivity problems. But I will point out that the problem occurs even if boundaries are smooth. Finally, section 5.3 presents my new algorithm that addresses these problems using a new gap function computation method (material depth) and the area integration of the contact force.

5.2 Stationary Rigid Obstacle

Contact occurs between two objects (or two parts of an object). Both sides deform as a result of the contact forces between them. But in many situations where one side is very hard compared to the other side, one can assume that the harder side is rigid. In this case, the rigid side is simply a fixed obstacle for the flexible side. This problem has been called *Signorini's problem* and has been extensively studied in mechanical engineering. In this section, this problem is considered to introduce a few important concepts, which are also useful in understanding the general contact cases described in later sections. These concepts include gap functions, contact forces, and penalty formulations.

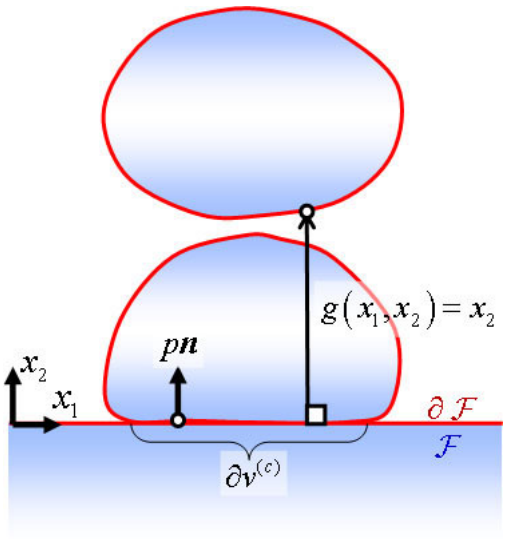

Figure 13: Geometric configuration of a deforming object and fixed obstacle.

5.2.1 Contact Kinematics

First, it is necessary to define the basic kinematics of contact problems (**Figure 13**). There is a deformable body, whose interior (open and bounded) is V and whose boundary is ∂V in the initial (undeformed) configuration. The object undergoes a deformation ϕ and the deformed interior is $v = \phi(V)$, and the deformed boundary is ∂v. There is a fixed and rigid obstacle, whose interior is $\mathcal{F}$, and its exterior is $\partial\mathcal{F}$. $\mathcal{F}$ is called a *foundation* since it is often a supporting structure for flexible objects in mechanical or architectural constructions. The distance to $\mathcal{F}$ is defined by a scalar function $g(\boldsymbol{x})$. $g(\boldsymbol{x})$ is called a *gap function* and measures the clearance or *gap* for avoiding collision with the obstacle $\mathcal{F}$. The sign of $g(\boldsymbol{x})$ is positive if $\boldsymbol{x}$ is outside of the obstacle:

$$\begin{cases} g(\boldsymbol{x}) < 0; & \forall \boldsymbol{x} \notin \mathcal{F} \cup \partial\mathcal{F} \\ g(\boldsymbol{x}) = 0; & \forall \boldsymbol{x} \in \partial\mathcal{F} \\ g(\boldsymbol{x}) > 0; & \forall \boldsymbol{x} \in \mathcal{F} \end{cases} \tag{5.1}$$

For example, if the obstacle is a ground plane at level zero (as shown in **Figure 13**), then $g(\boldsymbol{x}) = x_2$.

No interior point in the flexible object should be located in the obstacle, so

$$g(\boldsymbol{x}) > 0 \quad \forall \boldsymbol{x} \in v \tag{5.2}$$

This is called the *impenetrability constraint*.

5.2.2 Contact Force and the Balance Equation

The contact dealt with in this dissertation is frictionless. Therefore, the contact force is normal to the surface. The normal is the partial derivative of g with respect to spatial coordinates. Thus the contact force is

$$\boldsymbol{t} = p\boldsymbol{n} = p\frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \tag{5.3}$$

This is the only traction force considered in this dissertation. Now the balance equation **(3.60)** becomes

$$\delta W = \int_v \boldsymbol{\sigma} : \delta \boldsymbol{d}\, dv - \int_v \boldsymbol{f} \cdot \delta \boldsymbol{v}\, dv - \int_{\partial v^{(c)}} p\frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \cdot \delta \boldsymbol{v}\, da = 0 \tag{5.4}$$

where the integration region $\partial v^{(c)}$ is the interface or *contact surface* of two participating media and $\partial v^{(c)} \subset \partial v$.

5.2.3 Penalty Formulation

The penalty method replaces the pressure p with the penalty $\varepsilon_p g$ as (see also **Figure 14**)

$$\delta W = \int_v \boldsymbol{\sigma} : \delta \boldsymbol{d}\, dv - \int_v \boldsymbol{f} \cdot \delta \boldsymbol{v}\, dv - \int_{\partial v^{(c)}} \varepsilon_p g\frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \cdot \delta \boldsymbol{v}\, da = 0 \tag{5.5}$$

Note that penalty force $\varepsilon_p g\, \partial g(\boldsymbol{x})/\partial \boldsymbol{x}$ is the partial derivative of penalty-potential energy $\psi = \frac{1}{2}\varepsilon_p g^2$, namely

$$\frac{\partial \psi}{\partial \boldsymbol{x}} = \frac{\partial}{\partial \boldsymbol{x}}\left(\frac{1}{2}\varepsilon_p g^2\right) = \varepsilon_p g\frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \tag{5.6}$$

This relationship is later used to derive penalty forces.

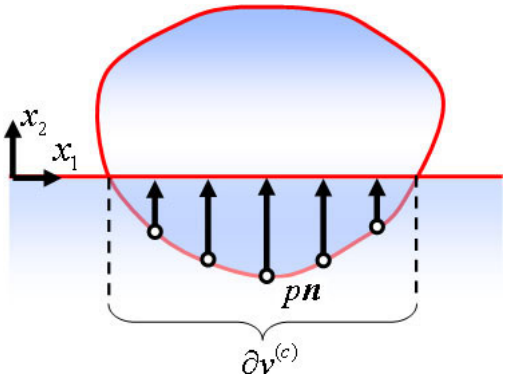

Figure 14: Penalty formulation.

5.3 Self Contact

Now, contact between flexible objects is considered (**Figure 15**). Contact can occur between two separate objects or between two parts of a single connected object. But in general, if an object can be deformed, it is possible that two parts of the object make a contact. In general, self-contacting cases should be considered. Once the possibility of self-contact is considered, the distinction between self-contact and the other kind is pointless. Therefore, in the later section, flexible-flexible contact and self-contact are deemed to mean the same thing.

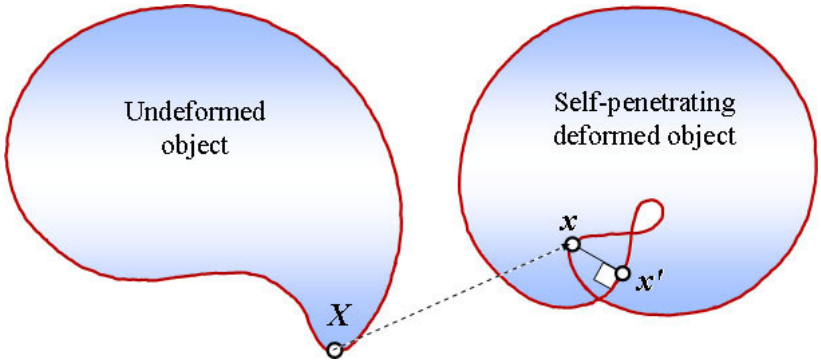

Figure 15: Geometric configuration of self-penetrating object.

5.3.1 Injectivity

Two particles in v cannot be mapped to the same location. The impenetrability constraint is equivalent to the injectivity of ϕ as

$$\forall \boldsymbol{X}, \boldsymbol{Y} \in V,\ \boldsymbol{X} \neq \boldsymbol{Y} \Rightarrow \boldsymbol{\phi}(\boldsymbol{X}) \neq \boldsymbol{\phi}(\boldsymbol{Y}) \tag{5.7}$$

5.3.2 Gap Function and Impenetrability Constraints

If penetration occurs, multiple particles may share the same location $\boldsymbol{x}$. The location alone is not enough information to determine which particle is examined. To prevent this ambiguity, the gap function must be defined as a function of the coordinates in the initial configuration as $g(\boldsymbol{X})$.

If self-contact is considered, the "obstacle" may be the deforming body itself. Unlike in Signorini's problem, the gap function cannot be defined as the distance between the point and boundaries since it is always zero. Instead of the distance, conventional methods use the distance from $\boldsymbol{x}(=\boldsymbol{\phi}(\boldsymbol{X}))$ to its projections on object boundaries. A projection of $\boldsymbol{x}$ on a surface is a point $\boldsymbol{x}'$ such that $\boldsymbol{x}-\boldsymbol{x}'$ is normal to the surface (**Figure 15**). A projection is a local minimizer of distance.

5.3.3 Balance Equations for Self Contacting Case

The balance equation for the self contact problem can be defined as

$$\delta W = \int_v \boldsymbol{\sigma} : \delta \boldsymbol{d}\, dv - \int_v \boldsymbol{f} \cdot \delta \boldsymbol{v}\, dv - \int_{\partial V^{(c)}} p \frac{\partial g(\boldsymbol{X})}{\partial \boldsymbol{x}} \cdot \delta \boldsymbol{v}\, da = 0 \tag{5.8}$$

The integration is over $\partial V^{(c)}$, which is the contact surface mapped back to the initial configuration so that the gap function $g(\boldsymbol{X})$ is uniquely evaluated. On the other hand the integrand consists of only spatial quantities including the elementary surface area da.

On the contact area, two opposing contact forces are action and reaction forces. According to Newton's third law, they satisfy

$$p_1 \frac{\partial g}{\partial \boldsymbol{x}}\bigg|_{X=X_1} = -p_2 \frac{\partial g}{\partial \boldsymbol{x}}\bigg|_{X=X_2}, \quad \boldsymbol{X}_1, \boldsymbol{X}_2 \in \partial V^{(C)}\ \ \boldsymbol{\phi}(\boldsymbol{X}_1) = \boldsymbol{\phi}(\boldsymbol{X}_2) \tag{5.9}$$

where distinct particles $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are in contact on the contact surface and p_1 and p_2 are the pressures on both sides.

5.3.4 Discretization of the Contact Force

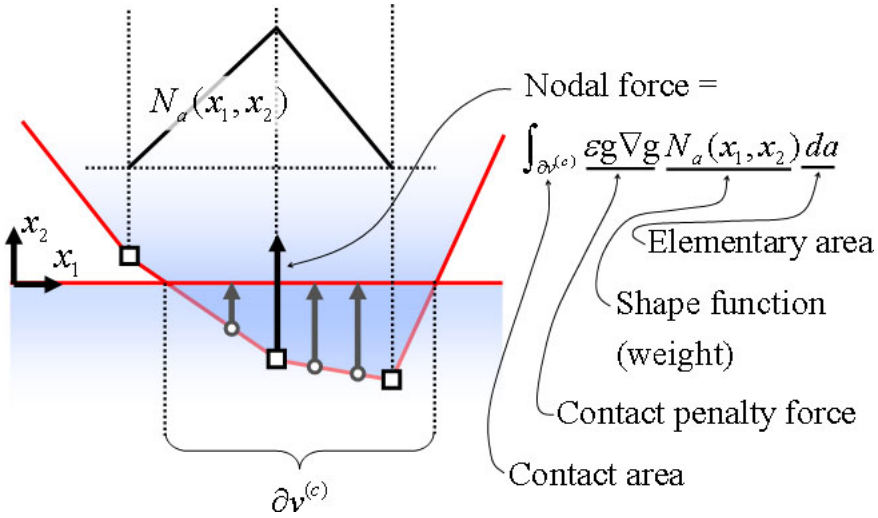

Figure 16: 2D illustration of contact force discretization: The upper object intrudes into the region of lower object. The upper object's boundary is discretized as line segments (in 3D cases, it is a triangle mesh). The squares are nodal points of the line segments. The nodal contact force of the a^{th} node is weighted average of the contact penalty force ($\varepsilon_p \nabla g$). The force is averaged over the contact area ($\partial v^{(c)}$), and the shape function of the node (N_a) is used as a weight.

The balance equation (5.8) is discretized by an FEM. The quantities in this FEM are illustrated in **Figure 16**. The discretization of $\delta W_{\text{int}}(\delta \boldsymbol{v}) = \int_v \boldsymbol{\sigma} : \delta \boldsymbol{d} \, dv$, the internal virtual work, and $\delta W_{ext}^{f}(\delta \boldsymbol{v}) = \int_v \boldsymbol{f} \cdot \delta \boldsymbol{v} \, dv$, the virtual work done by external body force, were explained in chapter 4. Here, the discretization of $\delta W_{ext}^{p}(\delta \boldsymbol{v}) = \int_{\partial V^{(c)}} p \frac{\partial g(\boldsymbol{X})}{\partial \boldsymbol{x}} \cdot \delta \boldsymbol{v} \, da$, the virtual work done by the penalty force, is discussed.

The virtual work caused by a single virtual velocity of node *a* is

$$\begin{aligned} \delta W_{ext}^{p}(N_a \delta \bar{\boldsymbol{v}}_a) &= \int_{\partial V^{(c)}} p \frac{\partial g}{\partial \boldsymbol{x}} \cdot N_a \delta \bar{\boldsymbol{v}}_a \, da \\ &= \delta \bar{\boldsymbol{v}}_a \cdot \int_{\partial V^{(c)}} p N_a \frac{\partial g}{\partial \boldsymbol{x}} \, da \\ &= \delta \bar{\boldsymbol{v}}_a \cdot \boldsymbol{F}_a^{cont} \end{aligned} \tag{5.10}$$

where $\boldsymbol{F}_a^{cont} = \int_{\partial V^{(c)}} N_a p \frac{\partial g}{\partial \boldsymbol{x}} da$ is the contact penalty force component of the equivalent force on node *a*. N_a is the shape function or weight assigned to node *a* (see (4.2)). In the penalty formulation, it is (see also **Figure 16**)

$$\boldsymbol{F}_a^{cont} = \int_{\partial V^{(c)}} \varepsilon_p g \nabla g N_a da \tag{5.11}$$

The partial differential of the spatial coordinates $\boldsymbol{x}$ with respect to the nodal coordinates $\bar{\boldsymbol{x}}_a$ is

$$\frac{\partial \boldsymbol{x}}{\partial \bar{\boldsymbol{x}}_a} = \begin{bmatrix} \partial x_1 / \partial \bar{x}_{a,1} & 0 & 0 \\ 0 & \partial x_2 / \partial \bar{x}_{a,2} & 0 \\ 0 & 0 & \partial x_3 / \partial \bar{x}_{a,3} \end{bmatrix} = N_a \boldsymbol{I} \tag{5.12}$$

Therefore,

$$\begin{aligned} \boldsymbol{F}_a^{cont} &= \int_{\partial V^{(c)}} \varepsilon_p g N_a \frac{\partial g}{\partial \boldsymbol{x}} da \\ &= \int_{\partial V^{(c)}} \varepsilon_p g N_a \boldsymbol{I} \frac{\partial g}{\partial \boldsymbol{x}} da \\ &= \int_{\partial V^{(c)}} \varepsilon_p g \left(\frac{\partial \boldsymbol{x}}{\partial \bar{\boldsymbol{x}}_a} \right)^T \frac{\partial g}{\partial \boldsymbol{x}} da \\ &= \int_{\partial V^{(c)}} \varepsilon_p g \frac{\partial g}{\partial \bar{\boldsymbol{x}}_a} da \\ &= \frac{\partial}{\partial \bar{\boldsymbol{x}}_a} \int_{\partial V^{(c)}} \psi da \end{aligned} \tag{5.13}$$

Thus the nodal contact penalty force is obtained as the partial derivative of the total penalty energy with respect to the nodal coordinates.

5.3.5 Point Collocation

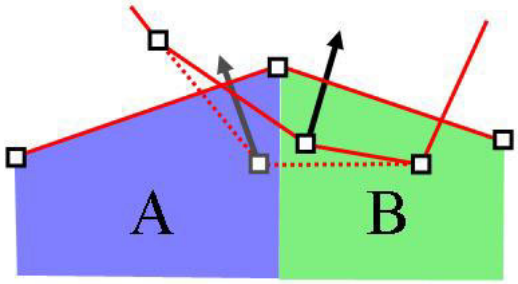

Figure 17: Point collocation and contact force discontinuity. The region A and B have different gradients creating a discontinuous boundary of the nodal contact force.

Now the numerical integration of equation (5.11) is considered. Since it is difficult to obtain an analytical expression of g, this integration cannot be performed easily. To simplify the computation, most conventional methods evaluate g only at nodal points (or a limited number of points) and perform the integration by the point collocation as

$$\boldsymbol{F}_a^{cont} = \varepsilon_p g(\boldsymbol{X}_a) \frac{\partial g}{\partial \boldsymbol{x}}\bigg|_{\boldsymbol{X}=\boldsymbol{X}_a} \int_{\partial v} N_a da \tag{5.14}$$

With this method, the nodal force depends only on the gap value at the nodal point and becomes discontinuous (see **Figure 17**). As shown below, this dependency causes a numerical problem.

5.3.6 Sensitive Projection Location and Gap Discontinuity

The gap function is evaluated by finding projections. By the definition of projection, the search algorithm has to rely on the surface normals. Unfortunately, this leads to a few problems, which can be summarized as two points:

1. The gap function becomes discontinuous with respect to the configuration because the location of the projection is highly sensitive to the deformation.
2. The criteria for selecting a projection among many candidates are difficult to define.

First, suppose that the gap function can be defined as the distance to the *closest* projection from a particle $\boldsymbol{x}$. If a surface has high curvature, the algorithm has to check many candidate

projections (**Figure 18**). As objects deform, the closest projection jumps between competing candidates, or a projection can emerge or disappear abruptly (**Figure 19**). Due to this location sensitivity, the derivative of the gap function, or even the gap function itself, is not a continuous function of the object configuration. Such a case often occurs when buckling creates wrinkles on a surface.

But the discussion above applies to only rather simple cases. In fact, it is not clear that choosing the closest projection is always desirable. As seen in **Figure 20**, the closest projection might be found in a neighbor, but the desirable projection might be found somewhere else in a farther area. Thus the criteria for choosing the "best" projection are difficult to define. It is not even obvious that one of the candidates can be uniquely selected because a particle at $\boldsymbol{x}$ may intrude into multiple objects. When a multi-object intrusion occurs, multiple projections should be used (**Figure 21**). Thus, to find projections, a global and exhaustive search algorithm is required.

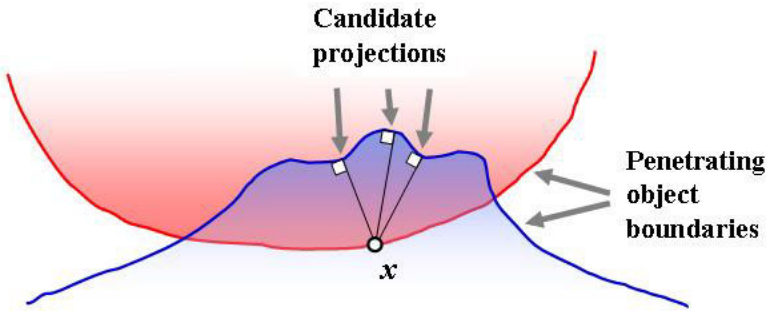

Figure 18: Derivative discontinuous gap function: A bumpy surface has many projections for a given particle *x*. In such a case, a small movement of *x* results in the jump of the closest projection. In this case, the gap function (defined as the closest projection) is continuous with respect to *x*, but its derivative with respect to *x* is not continuous.

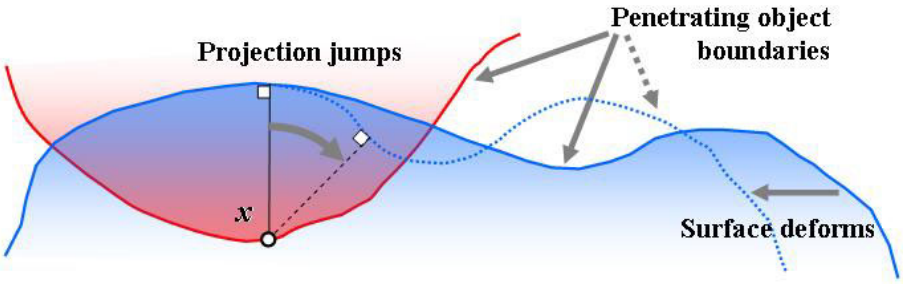

Figure 19: Discontinuous gap function: A small deformation creates a new projection, which can be closer than the old projection causing an abrupt jump of the gap function value. This case is worse than Figure 18 since the gap function, itself, is discontinuous.

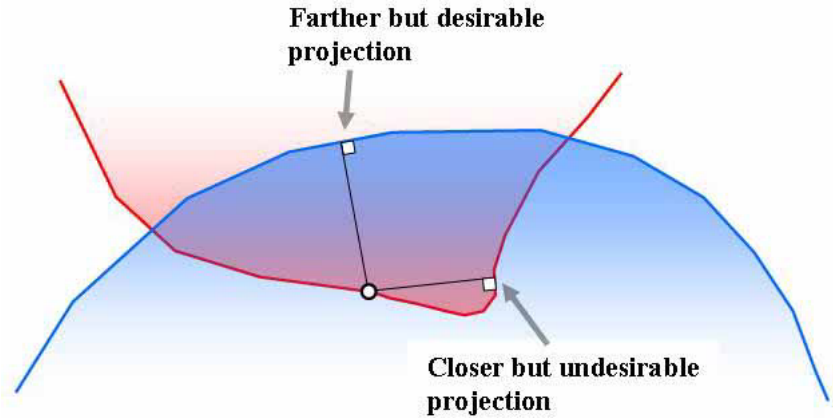

Figure 20: Closest but undesirable projection: The closest projection found in the neighbor is intuitively not desirable. However it is hard to define criteria that make the algorithm prefer the desirable projection. With the presence of self-contact, both red and blue regions may belong to a single object. Therefore, it is not possible to mechanically preclude projections on red regions.

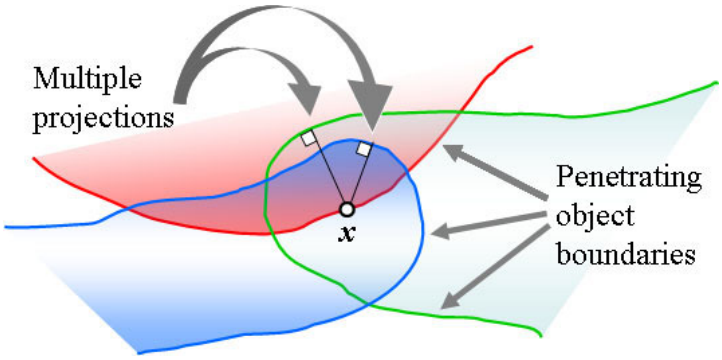

Figure 21: Multiple valid projections: Since the particle x intrudes on more than one region of nearby objects, it makes more sense to use multiple projections. The red, blue, and green regions may be just parts of a single object.

5.3.7 Convergence Problem

The point collocation technique (5.14) samples the discontinuous function *g*. The resulting penalty force $\boldsymbol{F}_a^{cont}$ is, therefore, discontinuous with respect to the mesh configuration (the mesh configuration is $\mathbf{x}$ defined by (4.28)). $\boldsymbol{F}_a^{cont}$ is a component of the residual force $\mathbf{R}(\mathbf{x})$ defined in (4.26). Thus the discontinuity of $\boldsymbol{F}_a^{cont}$ implies the discontinuity of $\mathbf{R}(\mathbf{x})$. This discontinuity often manifests itself as the convergence problem (oscillation) in nonlinear system solving because, with the presence of a discontinuity, the equation $\mathbf{R}(\mathbf{x}) = 0$ often lacks a solution. Such a convergence problem is by far more serious than the performance issue because it is very difficult to decide when the computation should be terminated.

In dynamic simulations, sudden changes of contact forces cause strange phenomena. If the force is pointing toward a discontinuous border, the nodal point is pushed by the alternating contact force and oscillation occurs. This is often cited as the "contact chatter" problem. If the force is pointing away from the border, the border works as a "slingshot" and the nodal point would be launched by the suddenly flipped contact force when it crosses the border.

5.4 The New Algorithm

The culprit in the convergence problem is the discontinuity of the contact force. The gap function is intrinsically nonsmooth. As long as point collocation, (5.14), is used, discontinuous contact forces are inevitable. A possible remedy to the discontinuity problem is to perform area integration, (5.11), which is repeated here:

$$\boldsymbol{F}_a^{cont} = \int_{\partial V^{(e)}} \varepsilon_p g N_a \frac{\partial g}{\partial \boldsymbol{x}} da \qquad (5.15)$$

Area integration averages various values of g and $\partial g / \partial \boldsymbol{x}$, resulting in a smooth contact force.

By sampling many points in the area, similar effects can be achieved, but that is computationally expensive. Therefore, a precise analytical integration is desirable. However,

if a projection is used to evaluate g and $\partial g/\partial \boldsymbol{x}$, the formulation of the analytical integration is not easy. As described in 5.3.6, it is already difficult to uniquely determine an optimum projection for a point. Analytically integrating a quantity that contains such values seems impossible. A more stable and simpler substitute for the projection is required. The following sections explain a new algorithm based on the use of *material depth* for evaluating the gap function.

5.4.1 Material Depth and Gap Continuity

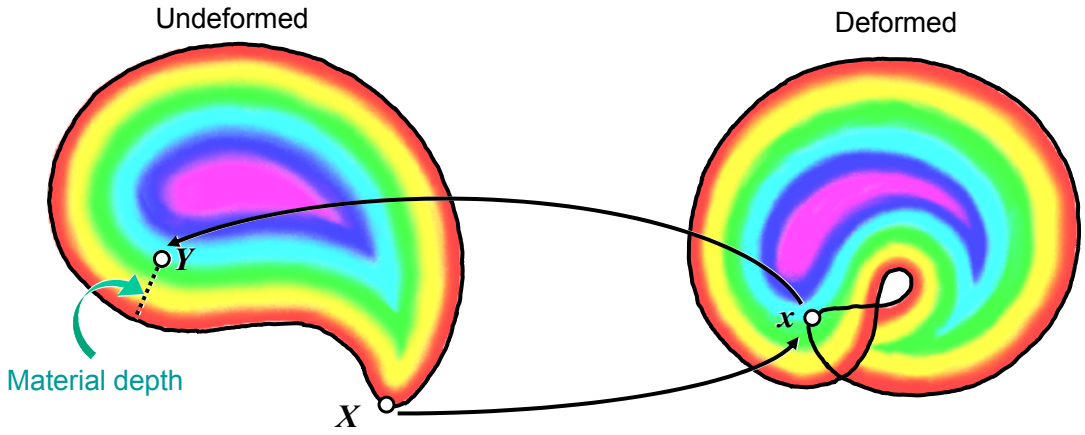

Figure 22: Material depth.

In this section, *material depth* is introduced as a new way to compute the gap function. **Figure 22** shows this notion. The deformation maps particle $\boldsymbol{X}$ on a boundary to the current position $\boldsymbol{x}$. As a result, $\boldsymbol{X}$ collides with particle $\boldsymbol{Y}$. The material depth is the distance from $\boldsymbol{Y}$ to the boundary in the reference configuration. It maintains a constant intrinsic value for a particle (or material point); hence the term "material depth". Since there is no self-penetration in the reference configuration, the material depth can be computed regardless of self-penetration in the current configuration. Also, unlike the distance to a projection, material depth never changes abruptly due to deformation.

Using the material depth as the gap function, as described above, is still not simple enough to allow analytical integration of (5.15). The new algorithm approximates the material depth by the linear interpolation of the material depths at nodal points of the FE

mesh (see **Figure 23**). The detailed derivation of analytical integration is deferred to the section 5.4.6.

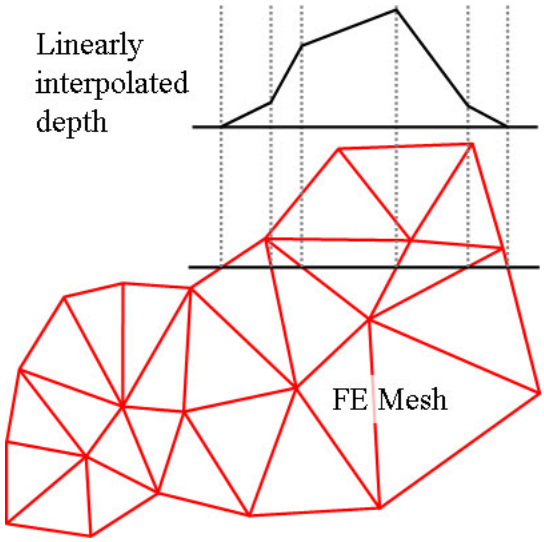

Figure 23: Linear interpolation of material depths.

Material depth is an approximation of the distance field in the current configuration. As an object is stretched or compressed, the depth value deviates from the actual distance. However, the inaccuracy does not cause a significant problem in terms of the maximum penetration allowed since in practice a large penalty factor can be used without causing numerical instability [CARSTENSEN 1999]. Thus the magnitude of the depth value is not a significant issue. A more important aspect of material depth is its gradient, which is used as an approximate surface normal. Since the material depth deviates from the spatial distance, the gradient also deviates from the actual normal. But this is not a serious problem either. The direction of the gradient is in fact the same as the exact normal on the boundary. Thus the gradient of the material depth is indeed a good approximation of the surface normal, and the directional accuracy increases as the algorithm converges to a solution with small penetration.

The gradient of material depth flips across the medial sheet of an object. So, if the depth of penetration is more than half the thickness of the object, my algorithm can no longer compute proper contact forces. This problem and a possible remedy for it are discussed in Section 9.3.1 and 10.1.1.

5.4.2 Algorithm Summary

The algorithm performs the following steps:

1. Compute the material depth for each node in the reference configuration.
2. Find the collision between a boundary element (triangle) and a volumetric element (tetrahedron).
3. Compute the intersection of the triangle and the tetrahedron.
4. Integrate the penalty force over the area of the intersection polygon and add their contributions to the global residual and the stiffness matrix.

Section 5.4.4, 5.4.5, and 5.4.6 describe each step in more detail.

5.4.3 Area Integration and Gap Gradient Continuity

The area of the intersection of the boundary triangles and the interior's tetrahedron varies continuously. The material depth is also continuous. Thus the penalty force is mostly a smooth function (C^1) of the mesh configuration $\mathbf{x}$. There are two exceptions. One occurs when an edge of a triangle and a side of the tetrahedron are coplanar, in which case the penalty force is no longer smooth but it is still continuous (C^0). The other exception occurs when the triangle is coplanar with a side of the tetrahedron. In this case the continuity no longer holds, but for this to occur, three vertices of the triangle must fall into the side of the tetrahedron, a behavior is not likely to happen in practice. Furthermore, if the side of tetrahedron is one of boundary triangles, continuity still holds. Thus the penalty forces are continuous in all plausible situations. This is a crucial point with respect to equation solving since a solution may not even exist without continuous penalty forces. This continuity is the major advantage of the new method over traditional methods. A more rigorous analysis is given in chapter 6.

5.4.4 Initial Depth Computation

For an object with few triangles, an exhaustive search algorithm is efficient enough for pre-computation of material depths at the node points. For more complex objects, I use the fast marching level set method, which quickly computes distance values [Sethian 1999]. By utilizing the tetrahedral mesh, I could use the finite element version of the fast marching method [Barth 1998], which can compute distances even for self-intersecting objects. This approach is left for future work.

5.4.5 Collision Detection

The algorithm needs to know which boundary elements (triangles) intersect with which volume elements (tetrahedral). In my application, the number of elements reaches tens to hundreds of thousands. An efficient search algorithm is required. The problem can be simply formulated as finding intersections between tetrahedra and triangles. It might appear that most query acceleration techniques from computational geometry are applicable to finding these intersections. However, special attention should be paid to the fact that those tetrahedra and triangles are parts of a finite element mesh. The first point of interest is the extremely high density of "objects." In other applications such as rigid body dynamics, objects are more complicated than tetrahedra, but usually there is some space between the objects. In finite element meshes, most elements have adjacent elements without any spaces between them. So the *object density* is measured not only in terms of the *number* of objects per unit volume but also the *object volume* per unit volume. If there is no space between objects, the density in terms of object volume is 100%. For a finite element mesh, this value can be very close to the maximum value.

Whether bucket sort or bounding volume methods are used, the typical strategy of query acceleration is to use some simple geometry that contains objects (e.g., bounding spheres, octree nodes) for avoiding unnecessary checks. High object (volume) density implies that those simple geometric shapes contain many objects. The methods most negatively affected by this characteristic are those based on sorting lists along each axis. For example, the sweep and prune algorithm is practically unusable in my application even with an implementation highly optimized for finite element meshes. In fact, even before considering the performance,

the algorithm must be rejected for its extremely high memory consumption (which eventually hits the performance as well). In the algorithm, coordinate overlap information is maintained to utilize temporal coherency. This means it needs to record all overlaps between tetrahedra. Because of the high volume density, there are very many tetrahedra that overlap with each other in a coordinate. If the element distribution is uniform, the number of tetrahedra that overlap a tetrahedron is $O(N^{2/3})$ where N is the number of tetrahedra in the mesh. There are O(N) tetrahedra in the mesh, so the overlap data consumes $O(N^{5/3})$ memory space. My implementation is optimized for finite element meshes, so it records only overlaps between nodal points and tetrahedra. However, this optimization does not change the asymptotic behavior. Superlinear memory consumption is not acceptable for large N. There is, however, a remedy for this problem. Brown et al [2000] divide the space into small partitions, and a simple sort list method is applied to the small number of objects in each partition.

However, Brown's algorithm is primarily developed for massively parallel machines. So, it is an over-kill for my purpose. To simplify implementation, I used hierarchical bounding volumes. The frequent deformation of the finite element mesh demands that the algorithm update the tree quickly. Therefore, a simple bounding volume, an interval in a coordinate, is chosen. The hierarchy is stored as a binary tree. A node of the tree represents an X, Y, or Z coordinate interval that bounds the intervals of all descendant nodes. The interval of a leaf node contains a tetrahedral element. The overlaps between the intervals of each triangle element and the intervals of tree nodes are examined, and final candidates are collected from tetrahedra in overlapping leaf nodes.

Each candidate tetrahedron is checked if it actually intersects with the triangle. Since there are usually many candidates thanks to the high volume density, trivial rejection is very important. The check is done by successively clipping the triangle by a plane of each side of the candidate tetrahedron. As soon as the triangle is found outside of the tetrahedron, the process skips to the next candidate. The algorithm can be further optimized by giving special treatment to the tetrahedra that are directly adjacent to the triangle. This is left for the future work.

The tree structure is built by a top-down partitioning of elements in the direction of their greatest extent. I usually divide an "analysis session" of a phenomenon (e.g., flexion of a joint) into sequential simulation runs. Since I set up the simulation runs such that the mesh

does not deform much in a single run, it is sufficient to build the tree structure only once at the beginning of each run. The interval values are efficiently updated in a bottom-up manner at every solution step. As a result, the collision detection occupies a rather minor part of the total computation time (see section 7.5.3).

5.4.6 Contact Force Computation

As described in the previous sections, the total contact penalty force is the sum of the contributions from all the pairs of intersecting tetrahedral triangular elements. In this section, the actual formula of the penalty force for such a pair is derived.

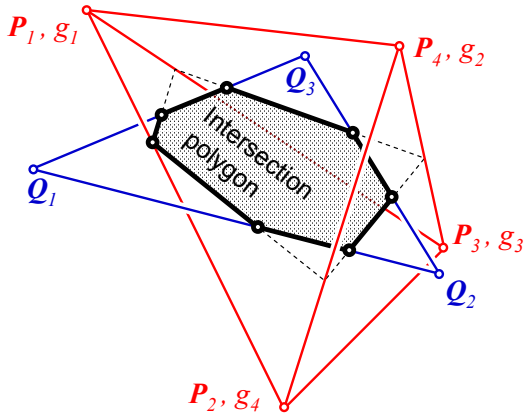

Figure 24: Intersecting Elements

The triangular element consists of 3 node points. Their coordinates are defined as three vectors:

$$\boldsymbol{Q}_i = \begin{bmatrix} Q_{i1} \\ Q_{i2} \\ Q_{i3} \end{bmatrix}, \quad i = 1,2,3\,. \tag{5.16}$$

In matrix form

$$\boldsymbol{Q} = \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} \\ Q_{21} & Q_{22} & Q_{23} \\ Q_{31} & Q_{32} & Q_{33} \end{bmatrix}. \tag{5.17}$$

Each row of the matrix corresponds to a node point.

A point $\boldsymbol{x}$ on the triangle is specified by the barycentric coordinates

$$\boldsymbol{x} = \zeta_1 \begin{bmatrix} Q_{11} \\ Q_{12} \\ Q_{13} \end{bmatrix} + \zeta_2 \begin{bmatrix} Q_{21} \\ Q_{22} \\ Q_{13} \end{bmatrix} + \zeta_3 \begin{bmatrix} Q_{11} \\ Q_{12} \\ Q_{13} \end{bmatrix} \tag{5.18}$$

where

$$\zeta_1 + \zeta_2 + \zeta_3 = 1, \quad \zeta_1, \zeta_2, \zeta_3 > 0 . \tag{5.19}$$

The tetrahedral element consists of 4 node points, whose coordinates are

$$\boldsymbol{P}_i = \begin{bmatrix} P_{i1} \\ P_{i2} \\ P_{i3} \end{bmatrix}, \ i = 1,2,3,4 . \tag{5.20}$$

In matrix form

$$\boldsymbol{P} = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \\ P_{41} & P_{42} & P_{43} \end{bmatrix} \tag{5.21}$$

A point $\boldsymbol{x}$ in the tetrahedral element is written as

$$\boldsymbol{x} = \xi_1 \begin{bmatrix} P_{11} \\ P_{12} \\ P_{13} \end{bmatrix} + \xi_2 \begin{bmatrix} P_{21} \\ P_{22} \\ P_{23} \end{bmatrix} + \xi_3 \begin{bmatrix} P_{31} \\ P_{32} \\ P_{33} \end{bmatrix} + \xi_4 \begin{bmatrix} P_{41} \\ P_{42} \\ P_{43} \end{bmatrix} \tag{5.22}$$

where

$$\xi_1 + \xi_2 + \xi_3 + \xi_4 = 1, \quad \xi_1, \xi_2, \xi_3, \xi_4 > 0 . \tag{5.23}$$

The gap function is approximated as the interpolation of material depths $g_1, g_2, \cdots, g_4$ assigned to node points of the tetrahedral element:

$$g = \begin{bmatrix} g_1 & g_2 & g_3 & g_4 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{bmatrix}$$

$$= \sum_{a=1}^{4} g_a \xi_a \tag{5.24}$$

Now by combining (5.18), (5.22), and (5.23)

$$\begin{bmatrix} P_{11} & P_{21} & P_{31} & P_{41} \\ P_{12} & P_{22} & P_{32} & P_{42} \\ P_{13} & P_{23} & P_{33} & P_{43} \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{21} & Q_{31} & 0 \\ Q_{12} & Q_{22} & Q_{32} & 0 \\ Q_{13} & Q_{23} & Q_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \\ 1 \end{bmatrix}. \tag{5.25}$$

Therefore,

$$\begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{bmatrix} = \bar{\boldsymbol{P}}^{-1} \bar{\boldsymbol{Q}} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \\ 1 \end{bmatrix} \tag{5.26}$$

where

$$\bar{\boldsymbol{P}} = \begin{bmatrix} P_{11} & P_{21} & P_{31} & P_{41} \\ P_{12} & P_{22} & P_{32} & P_{42} \\ P_{13} & P_{23} & P_{33} & P_{43} \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{5.27}$$

$$\bar{\boldsymbol{Q}} = \begin{bmatrix} Q_{11} & Q_{21} & Q_{31} & 0 \\ Q_{12} & Q_{22} & Q_{32} & 0 \\ Q_{13} & Q_{23} & Q_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.28}$$

So (5.24) can be rewritten as

$$\begin{aligned} g &= \sum_{a=1}^{4} g_a \sum_{b=1}^{4} \left(\sum_{c=1}^{4} \left(\bar{\boldsymbol{P}}^{-1} \right)_{ac} \bar{Q}_{cb} \right) \zeta_b \\ &= \sum_{b=1}^{4} \zeta_b \sum_{a=1}^{4} g_a \sum_{c=1}^{4} \left(\bar{\boldsymbol{P}}^{-1} \right)_{ac} \bar{Q}_{cb} \end{aligned} \tag{5.29}$$

where $\zeta_4 = 1$ is defined for convenience.

The penalty energy per unit area at $\boldsymbol{x}$ is

$$\psi = \frac{1}{2} \varepsilon_p g^2 . \tag{5.30}$$

The contribution to the nodal equivalent forces at node points from this point is

$$\begin{aligned}\psi_{\boldsymbol{P}} &= \frac{\partial}{\partial \boldsymbol{P}}\psi \\ &= \varepsilon_p g \frac{\partial g}{\partial \boldsymbol{P}}\end{aligned} \tag{5.31}$$

$$\begin{aligned}\psi_{\boldsymbol{Q}} &= \frac{\partial}{\partial \boldsymbol{Q}}\psi \\ &= \varepsilon_p g \frac{\partial g}{\partial \boldsymbol{Q}}\end{aligned} \tag{5.32}$$

In the following derivation, it is more convenient to use a componentwise description. Thus

$$\begin{aligned}\psi_{P_{ij}} &= \frac{\partial}{\partial P_{ij}}\psi \\ &= \varepsilon_p g \frac{\partial g}{\partial P_{ij}}\end{aligned} \tag{5.33}$$

$$\begin{aligned}\psi_{Q_{ij}} &= \frac{\partial}{\partial Q_{ij}}\psi \\ &= \varepsilon_p g \frac{\partial g}{\partial Q_{ij}}.\end{aligned} \tag{5.34}$$

Now

$$\begin{aligned}\frac{\partial g}{\partial P_{ij}} &= \sum_{b=1}^{4}\zeta_b \sum_{a=1}^{4} g_a \sum_{c=1}^{4} \frac{\partial}{\partial P_{ij}}\left(\bar{\boldsymbol{P}}^{-1}\right)_{ac}\bar{Q}_{cb} \\ &= \sum_{b=1}^{4}\zeta_b \sum_{a=1}^{4} g_a \sum_{c=1}^{4}\left(\sum_{d,e=1}^{4}\frac{\partial \bar{P}_{de}}{\partial P_{ij}}\frac{\partial\left(\bar{\boldsymbol{P}}^{-1}\right)_{ac}}{\partial \bar{P}_{de}}\right)\bar{Q}_{cb}, \quad i=1,2,3 \quad j=1,2,3,4\end{aligned} \tag{5.35}$$

By examining components of (5.27)

$$\frac{\partial \bar{P}_{de}}{\partial P_{ij}} = \delta_{ie}\delta_{jd}\,. \tag{5.36}$$

Therefore,

$$\begin{aligned}\frac{\partial g}{\partial P_{ij}} &= \sum_{b=1}^{4}\zeta_b \sum_{a=1}^{4} g_a \sum_{c=1}^{4}\left(\sum_{d,e=1}^{4}\delta_{ie}\delta_{jd}\frac{\partial\left(\bar{\boldsymbol{P}}^{-1}\right)_{ac}}{\partial \bar{P}_{de}}\right)\bar{Q}_{cb} \\ &= \sum_{b=1}^{4}\zeta_b \sum_{a=1}^{4} g_a \sum_{c=1}^{4}\frac{\partial\left(\bar{\boldsymbol{P}}^{-1}\right)_{ac}}{\partial \bar{P}_{ji}}\bar{Q}_{cb}\end{aligned} \tag{5.37}$$

Now given an invertible matrix $\boldsymbol{A}$,

$$
\begin{aligned}
&\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I} \\
&\frac{\partial\left(\boldsymbol{A}^{-1}\right)}{\partial A_{ij}}\boldsymbol{A} + \boldsymbol{A}^{-1}\frac{\partial \boldsymbol{A}}{\partial A_{ij}} = \boldsymbol{0} \\
&\frac{\partial\left(\boldsymbol{A}^{-1}\right)}{\partial A_{ij}}\boldsymbol{A} = -\boldsymbol{A}^{-1}\frac{\partial \boldsymbol{A}}{\partial A_{ij}} \\
&\frac{\partial\left(\boldsymbol{A}^{-1}\right)}{\partial A_{ij}} = -\boldsymbol{A}^{-1}\frac{\partial \boldsymbol{A}}{\partial A_{ij}}\boldsymbol{A}^{-1}
\end{aligned}
$$

Thus

$$
\begin{aligned}
\frac{\partial\left(\boldsymbol{A}^{-1}\right)_{kl}}{\partial A_{ij}} &= -\sum_{a}\left(\boldsymbol{A}^{-1}\right)_{ka}\left(\sum_{b}\left(\frac{\partial \boldsymbol{A}}{\partial A_{ij}}\right)_{ab}\left(\boldsymbol{A}^{-1}\right)_{bl}\right) \\
&= -\sum_{a}\left(\boldsymbol{A}^{-1}\right)_{ka}\left(\sum_{b}\delta_{ia}\delta_{jb}\left(\boldsymbol{A}^{-1}\right)_{bl}\right) \\
&= -\left(\boldsymbol{A}^{-1}\right)_{ki}\left(\boldsymbol{A}^{-1}\right)_{jl}
\end{aligned}
\tag{5.38}
$$

By applying this formula, (5.37) is rewritten as

$$
\begin{aligned}
\frac{\partial g}{\partial P_{ij}} &= \sum_{b=1}^{4}\zeta_b\sum_{a=1}^{4}g_a\sum_{c=1}^{4}\left(-\left(\bar{\boldsymbol{P}}^{-1}\right)_{aj}\left(\bar{\boldsymbol{P}}^{-1}\right)_{ic}\right)\bar{Q}_{cb} \\
&= \sum_{b=1}^{4}\zeta_b\left(-\sum_{a=1}^{4}g_a\left(\bar{\boldsymbol{P}}^{-1}\right)_{aj}\sum_{c=1}^{4}\left(\bar{\boldsymbol{P}}^{-1}\right)_{ic}\bar{Q}_{cb}\right)
\end{aligned}
\tag{5.39}
$$

Also

$$
\begin{aligned}
\frac{\partial g}{\partial Q_{ij}} &= \sum_{b=1}^{4}\zeta_b\sum_{a=1}^{4}g_a\sum_{c=1}^{4}\left(\bar{\boldsymbol{P}}^{-1}\right)_{ac}\frac{\partial \bar{Q}_{cb}}{\partial Q_{ij}} \\
&= \sum_{b=1}^{4}\zeta_b\sum_{a=1}^{4}g_a\sum_{c=1}^{4}\left(\bar{\boldsymbol{P}}^{-1}\right)_{ac}\delta_{ib}\delta_{jc} \\
&= \zeta_i\sum_{a=1}^{4}g_a\left(\bar{\boldsymbol{P}}^{-1}\right)_{aj}, \quad i,j = 1,2,3
\end{aligned}
\tag{5.40}
$$

The results can be summarized as

$$
g = \sum_{b=1}^{4}G_b\zeta_b \tag{5.41}
$$

$$
\frac{\partial g}{\partial P_{ij}} = \sum_{b=1}^{4}G'_{b,ij}\zeta_b \tag{5.42}
$$

$$\frac{\partial g}{\partial Q_{ij}} = G_j''\zeta_i, \quad i,j = 1,2,3 \tag{5.43}$$

where

$$G_b = \sum_{a=1}^{4} g_a \sum_{c=1}^{4} \left(\bar{\boldsymbol{P}}^{-1}\right)_{ac} \bar{Q}_{cb}, \; b = 1,2,3,4 \tag{5.44}$$

$$G'_{b,ij} = -\sum_{a=1}^{4} g_a \left(\bar{\boldsymbol{P}}^{-1}\right)_{aj} \sum_{c=1}^{4} \left(\bar{\boldsymbol{P}}^{-1}\right)_{ic} \bar{Q}_{cb} \tag{5.45}$$

$$G_j'' = \sum_{a=1}^{4} g_a \left(\bar{\boldsymbol{P}}^{-1}\right)_{aj} . \tag{5.46}$$

Finally, (5.33) and (5.34) are

$$\psi_{P_{ij}} = \varepsilon_p \sum_{b=1}^{4} G_b \zeta_b \times \sum_{b=1}^{4} G'_{b,ij} \zeta_b \tag{5.47}$$

$$\psi_{Q_{ij}} = \varepsilon_p \sum_{b=1}^{4} G_b \zeta_b \times G_j'' \zeta_i \tag{5.48}$$

Noting (5.19) and $\zeta_4 = 1$, the above equations can be written as a quadratic polynomial of the form:

$$f = C_1\zeta_2^2 + \left(C_2\zeta_1 + C_3\right)\zeta_2 + C_4\zeta_1^2 + C_5\zeta_1 + C_6 \tag{5.49}$$

where coefficients $C_1, C_2, \cdots, C_6$ are quadratic polynomials of G_b, $G'_{b,ij}$ and G_j''.

f must be integrated over intersecting region Γ to obtain nodal forces:

$$\int_\Gamma f \, da = 2A_{tri} \int_\Gamma f \, d\zeta_1 d\zeta_2 \tag{5.50}$$

where A_{tri} is the area of the triangle element:

$$A_{tri} = \frac{1}{2}\left|\left(\boldsymbol{Q}_1 - \boldsymbol{Q}_3\right) \times \left(\boldsymbol{Q}_2 - \boldsymbol{Q}_3\right)\right| . \tag{5.51}$$

The intersection region Γ is a convex polygon with up to 7 sides in $\zeta_1 - \zeta_2$ space.

An edge of the polygon is described by a tuple of 4 parameters:

$$\left[\zeta_{1,1}, \zeta_{1,2}, \alpha, \beta\right] \tag{5.52}$$

where $\left[\zeta_{1,1}, \zeta_{1,2}\right]$ is an interval in the ζ_1 direction and α and β define the line equation $\zeta_2 = \alpha\zeta_1 + \beta$. Let Θ be a set of all edges, except for the edges that are parallel to the ζ_1 axis,

and let the elements of the edges be expressed in this tuple format. The integration of the force over the intersection polygon is a sum of integrations over the regions between each segments and the ζ_1 axis:

$$2A_{tri}\sum_{(\zeta_{1,1},\zeta_{1,2},\alpha,\beta)\in\Theta}\int_{\zeta_{1,1}}^{\zeta_{1,2}}\int_0^{\alpha\zeta_1+\beta} f\,d\zeta_2 d\zeta_1 . \tag{5.53}$$

The edges that are parallel to the ζ_1 axis do not have any contribution to the integration, so they are ignored.

The double integration is trivially performed as

$$\begin{aligned}
&\int_{\zeta_{1,1}}^{\zeta_{1,2}}\int_0^{\alpha\zeta_1+\beta} f\,d\zeta_2 d\zeta_1 \\
&= \int_{\zeta_{1,1}}^{\zeta_{1,2}}\int_0^{\alpha\zeta_1+\beta}\left(C_1\zeta_2^2+\left(C_2\zeta_1+C_3\right)\zeta_2+C_4\zeta_1^2+C_5\zeta_1+C_6\right)d\zeta_2 d\zeta_1 \\
&= \int_{\zeta_{1,1}}^{\zeta_{1,2}}\left(\frac{C_1}{3}\zeta_2^3+\frac{C_2\zeta_1+C_3}{2}\zeta_2^2+\left(C_4\zeta_1^2+C_5\zeta_1+C_6\right)\zeta_2\right)\Bigg|_{\zeta_2=\alpha\zeta_1+\beta} d\zeta_1 \\
&= \int_{\zeta_{1,1}}^{\zeta_{1,2}}\left(C_1'\zeta_1^3+C_2'\zeta_1^2+C_3'\zeta_1+C_4'\right)d\zeta_1 \\
&= \frac{C_1'}{4}\left(\zeta_{1,2}^4-\zeta_{1,1}^4\right)+\frac{C_2'}{3}\left(\zeta_{1,2}^3-\zeta_{1,1}^3\right)+\frac{C_3'}{2}\left(\zeta_{1,2}^2-\zeta_{1,1}^2\right)+C_4'\left(\zeta_{1,2}-\zeta_{1,1}\right)
\end{aligned} \tag{5.54}$$

Thus nodal penalty forces are analytically integrated over the intersection area.

6 ALGORITHM ANALYSIS

6.1 Overview

In the previous section a contact handling algorithm was explained. In this chapter the algorithm is analyzed mathematically and numerically. The first topic is the existence of a solution. Due to their point sampling nature, many methods encounter the cases where they do not converge because of the lack of a solution, i.e., the zero force points. Therefore, it is important to examine the *existence of a solution* in the formulation of my algorithm.

To prove the existence of a solution, it is necessary to show that the force is a *continuous* function of the mesh configuration. As shown in the previous chapter, the total force is the summation of the elastic force and the contact penalty force. While it is trivially shown that the elastic force is continuous with respect to the mesh configuration, the proof of the contact penalty force continuity is far more complicated because it is integrated over a region that also varies as the mesh configuration changes. Section 6.3 is devoted to this proof.

The algorithm uses the penalty method. So it allows some amount of penetration between objects. The next section shows that the amount of penetration approaches zero as the penalty factor goes to infinity. In fact, it is shown that at the limit point the solution of the penalty formulation is also the solution of the constrained problem. The implication of this is that, besides numerical issues, the penetration can be reduced to a desirable level by increasing the penalty factor.

The final section presents a numerical comparison of a point sampling based method and my area integration based algorithm to illustrate the superior convergence characteristics of my method. I also examine the influence of the penalty factor to convergence speed and maximum penetration depth.

6.2 Existence of Solution

As shown in the previous chapters, the total force is derived as the first derivative of the energy function with respect to the mesh configuration. If the energy function has a minimum point and is continuous at least at the minimum point, then the derivative of the energy (i.e., the force) vanishes. Therefore, the minimum point is the solution of the nonlinear system. Thus the proof is carried on in two parts:

1. Existence of the energy minimum
2. Continuity of the force function

As shown in (4.28) the mesh configuration consists of n nodal points as

$$\mathbf{x} = \begin{bmatrix} \bar{\boldsymbol{x}}_1 \\ \bar{\boldsymbol{x}}_2 \\ \vdots \\ \bar{\boldsymbol{x}}_n \end{bmatrix} \tag{6.1}$$

The initial configuration and nodal positions are denoted as $\mathbf{x}^{(0)}$ and $\bar{\boldsymbol{x}}_1^{(0)}, \bar{\boldsymbol{x}}_2^{(0)}, \cdots, \bar{\boldsymbol{x}}_n^{(0)}$.

The energy W is a function of the mesh configuration

$$W(\mathbf{x}) \tag{6.2}$$

It is assumed that *at least one node point is fixed in space*, or more formally

$$\exists i, \;\; \bar{\boldsymbol{x}}_i \equiv \bar{\boldsymbol{x}}_i^{(0)} \tag{6.3}$$

This is a reasonable assumption since, without such a constraint, the problem becomes ill-posed, which makes the problem itself uninteresting.

Theorem I: If the assumption (6.3) is satisfied, the energy function $W(\mathbf{x})$ has a minimum point in an open set that includes the initial configuration $\mathbf{x}^{(0)}$.

Proof: The key idea of the proof is to trap the minimum point in an open set Ξ. Ξ is defined as follows:

$$\Xi = \left\{ \mathbf{x}; W(\mathbf{x}) < W\left(\mathbf{x}^{(0)}\right) + \varepsilon \right\} \tag{6.4}$$

where ε is an arbitrary positive number $\varepsilon \in \mathbb{R}, \varepsilon > 0$. Ξ contains every $\mathbf{x}$ that makes $W(\mathbf{x})$ less than $W(\mathbf{x}^{(0)})$, so if the minimum point exists, it must be included in Ξ. To complete the proof of the existence of the minimum point, two things have to be shown:

Property A: Ξ is a bounded set.

Property B: $\inf_{\Xi} W(\mathbf{x})$ is not a value of $W(\mathbf{x})$ where $\mathbf{x}$ is on the boundary of Ξ.

Property A is shown as follows. $W(\mathbf{x})$ is the sum of the elastic potential energy and the contact penalty energy. Since at least one node point is fixed, as a coordinate of another node point goes to infinity, at least one of the tetrahedral elements must be infinitely stretched, so the later energy function also approaches infinity. On the other hand, the contact penalty energy is bounded everywhere. Therefore, there exists δ such that

$$\forall \mathbf{x}, \mathbf{x} \in \Xi \quad \left\| \mathbf{x} - \mathbf{x}^{(0)} \right\| < \delta \tag{6.5}$$

Thus Ξ is both open and bounded.

Property B is proven by contradiction. Assume the contrary of **Theorem I**: there is no minimum in Ξ. By definition $W(\mathbf{x}) \geq 0$, so $W(\mathbf{x})$ is bounded. So $\inf_{\Xi} W(\mathbf{x})$ exists. Since $W(\mathbf{x})$ does not have a minimum in Ξ, $\inf_{\Xi} W(\mathbf{x})$ must be on the open boundary $\partial\Xi$. On the other hand, $W(\mathbf{x})$ goes to $W(\mathbf{x}^{(0)}) + \varepsilon > W(\mathbf{x}^{(0)})$ on $\partial\Xi$. Therefore, $\inf_{\Xi} W(\mathbf{x})$ cannot be on $\partial\Xi$. Thus the energy function $W(\mathbf{x})$ must have a minimum point in Ξ.

The total force is the summation of the elastic force and the contact penalty force. So, to prove the second part, it is sufficient to show that both of these forces are continuous. From the material energy definitions in section 3.6, the forces are discontinuous only when the determinant of the deformation gradient is zero, i.e., the tetrahedral element has zero volume. Since the algorithm prohibits such a configuration, it can be excluded from the domain considered. The continuity of the contact penalty force is proved in the next section. Given the result of the next section, it is concluded that the solution, or a zero force configuration, exists for the problem formulated in my algorithm.

6.3 Proof of Penalty Force Continuity

As shown in (5.44)–(5.49), the force field is defined as a rational function of the triangle parameterization ζ_1, ζ_2, the tetrahedron vertices $\boldsymbol{P}$, and the triangle vertices $\boldsymbol{Q}$:

$$f = \mathrm{F}(\zeta_1, \zeta_2, \boldsymbol{P}, \boldsymbol{Q}) \tag{6.6}$$

The nodal contact force is integrated as

$$\int_\Gamma f \, da = 2A_{tri} \int_\Gamma \mathrm{F}(\zeta_1, \zeta_2, \boldsymbol{P}, \boldsymbol{Q}) d\zeta_1 d\zeta_2 \tag{6.7}$$

where A_{tri} is the area of the triangle, which is a continuous function of $\boldsymbol{Q}$. Therefore, the proof is focused on the continuity of the function:

$$\mathrm{F}'(\boldsymbol{P}, \boldsymbol{Q}) = \int_\Gamma \mathrm{F}(\zeta_1, \zeta_2, \boldsymbol{P}, \boldsymbol{Q}) d\zeta_1 d\zeta_2 \tag{6.8}$$

For the sake of concise notation, in the following argument, $\boldsymbol{P}$ and $\boldsymbol{Q}$ are often treated as a single 3×7 matrix $\boldsymbol{V}$:

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{P}_1 & \boldsymbol{P}_2 & \boldsymbol{P}_3 & \boldsymbol{P}_4 & \boldsymbol{Q}_1 & \boldsymbol{Q}_2 & \boldsymbol{Q}_3 \end{bmatrix} \tag{6.9}$$

(6.8) is rewritten as

$$\mathrm{F}'(\boldsymbol{V}) = \int_\Gamma \mathrm{F}(\zeta_1, \zeta_2, \boldsymbol{V}) d\zeta_1 d\zeta_2 \tag{6.10}$$

Γ is the intersection of the triangle and the tetrahedron in $\zeta_1 - \zeta_2$ coordinate space. Let $\boldsymbol{V}'$ have a value slightly different from $\boldsymbol{V}$. Let be the intersecting region that corresponds to $\boldsymbol{V}'$. Now $\mathrm{F}'(\boldsymbol{V}')$ is integrated over the region Γ' as

$$\mathrm{F}'(\boldsymbol{V}') = \int_{\Gamma'} \mathrm{F}(\zeta_1, \zeta_2, \boldsymbol{V}) d\zeta_1 d\zeta_2 \tag{6.11}$$

$\mathrm{F}'(\boldsymbol{V})$ is a continuous function *if and only if* the difference between $\mathrm{F}'(\boldsymbol{V}')$ and $\mathrm{F}'(\boldsymbol{V})$ goes to zero as $\boldsymbol{V}'$ approaches $\boldsymbol{V}$, i.e.,

$$\lim_{\boldsymbol{V}' \to \boldsymbol{V}} \left| \int_{\Gamma'} \mathrm{F}(\zeta_1, \zeta_2, \boldsymbol{V}') d\zeta_1 d\zeta_2 - \int_\Gamma \mathrm{F}(\zeta_1, \zeta_2, \boldsymbol{V}) d\zeta_1 d\zeta_2 \right| = 0 \tag{6.12}$$

Now the following inequality holds:

$$
\begin{aligned}
&\left|\int_{\Gamma'} \mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V}')d\zeta_1 d\zeta_2 - \int_{\Gamma} \mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V})d\zeta_1 d\zeta_2\right| \\
&= \left|\int_{\Gamma} \big(\mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V}') - \mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V})\big)d\zeta_1 d\zeta_2\right. \\
&\quad \left. + \int_{\Gamma'\cap\bar{\Gamma}} \mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V}')d\zeta_1 d\zeta_2 - \int_{\bar{\Gamma}'\cap\Gamma} \mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V})d\zeta_1 d\zeta_2\right| \\
&\le \left|\int_{\Gamma} \big(\mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V}') - \mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V})\big)d\zeta_1 d\zeta_2\right| \\
&\quad + \left|\int_{\Gamma'\cap\bar{\Gamma}} \mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V}')d\zeta_1 d\zeta_2\right| + \left|\int_{\bar{\Gamma}'\cap\Gamma} \mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V})d\zeta_1 d\zeta_2\right|
\end{aligned} \tag{6.13}
$$

For the second and third terms, the following inequalities holds.

$$
\begin{aligned}
&\left|\int_{\Gamma'\cap\bar{\Gamma}} \mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V}')d\zeta_1 d\zeta_2\right| \le \sup_{\Gamma'\cap\bar{\Gamma}}\Big(\big|\mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V}')\big|\Big)\,area\big(\Gamma'\cap\bar{\Gamma}\big) \\
&\left|\int_{\bar{\Gamma}'\cap\Gamma} \mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V})d\zeta_1 d\zeta_2\right| \le \sup_{\bar{\Gamma}'\cap\Gamma}\Big(\big|\mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V})\big|\Big)\,area\big(\bar{\Gamma}'\cap\Gamma\big)
\end{aligned} \tag{6.14}
$$

where $area(\Pi)$ denotes the area of integration region Π:

$$
area(\Pi) = \int_{\Pi} d\xi_1 d\xi_2 \,. \tag{6.15}
$$

Thus the following three equations imply the final goal (6.12):

$$
\lim_{\boldsymbol{V}'\to\boldsymbol{V}} \left|\int_{\Gamma} \mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V}')d\zeta_1 d\zeta_2 - \int_{\Gamma} \mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V})d\zeta_1 d\zeta_2\right| = 0 \tag{6.16}
$$

$$
\sup_{\Gamma'\cap\bar{\Gamma}}\Big(\big|\mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V}')\big|\Big) \lim_{\boldsymbol{V}'\to\boldsymbol{V}} area\big(\Gamma'\cap\bar{\Gamma}\big) = 0 \tag{6.17}
$$

$$
\sup_{\bar{\Gamma}'\cap\Gamma}\Big(\big|\mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V})\big|\Big) \lim_{\boldsymbol{V}'\to\boldsymbol{V}} area\big(\bar{\Gamma}'\cap\Gamma\big) = 0 \tag{6.18}
$$

The proof follows the steps below.

Step 1: Prove that $\mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V})$ is continuous (**proposition i**).

Step 2: Prove that $\mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V})$ is bounded (**proposition ii**).

Step 3: (6.16) holds because of **proposition i**.

Step 4: $\sup_{\Gamma'\cap\bar{\Gamma}}\Big(\big|\mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V}')\big|\Big)$ in (6.17) and $\sup_{\bar{\Gamma}'\cap\Gamma}\Big(\big|\mathrm{F}(\zeta_1,\zeta_2,\boldsymbol{V})\big|\Big)$ in (6.18) are bounded because of **proposition ii**.

Step 5: Prove that $\lim_{\boldsymbol{V}'\to\boldsymbol{V}} area\big(\Gamma'\cap\bar{\Gamma}\big) = 0$ and $\lim_{\boldsymbol{V}'\to\boldsymbol{V}} area\big(\bar{\Gamma}'\cap\Gamma\big) = 0$ (**proposition iii**).

Step 6: The result of **step 4** and **proposition iii** imply (6.17) and (6.18).

Step 7: (6.16), (6.17), and (6.18) imply (6.12).

Therefore, proving **proposition i**, **ii**, and **iii** will complete the continuity proof.

6.3.1 Proof of **proposition i** *and* **ii**

Proposition i and **ii** are subject to following assumption:

Assumption 1: *The tetrahedral element has a positive volume, i.e.*,

$$\det \bar{\boldsymbol{P}} = \det \begin{bmatrix} P_{11} & P_{21} & P_{31} & P_{41} \\ P_{12} & P_{22} & P_{32} & P_{42} \\ P_{13} & P_{23} & P_{33} & P_{43} \\ 1 & 1 & 1 & 1 \end{bmatrix} > 0 \tag{6.19}$$

This assumption holds because, as described in section 4.7, the algorithm rejects configurations that have reversed or flattened tetrahedral elements. Under this assumption, by examining the rational equations (5.44)–(5.49), it is clear that **proposition i** and **ii** hold.

6.3.2 Proof of **proposition iii**

6.3.2.1 Affine transformation

Proposition iii consists of two equations:

$$\lim_{\boldsymbol{V}' \to \boldsymbol{V}} area\left(\Gamma' \cap \bar{\Gamma}\right) = 0 \tag{6.20}$$

$$\lim_{\boldsymbol{V}' \to \boldsymbol{V}} area\left(\bar{\Gamma}' \cap \Gamma\right) = 0 \tag{6.21}$$

Γ (or Γ') is an intersection of the triangle and tetrahedron. The intersection is a convex polygon with up to 7 edges in the $\zeta_1 - \zeta_2$ coordinate space. $\Gamma' \cap \bar{\Gamma}$ and $\bar{\Gamma}' \cap \Gamma$ are also polygons in the $\zeta_1 - \zeta_2$ space. To prove proposition **iii**, the edges of these polygons have to be described in terms of $\boldsymbol{V}$ and $\boldsymbol{V}'$ (3D coordinates of the vertices of the triangles and tetrahedra). In the following discussion, it is convenient to use a 3D coordinate system that is aligned with $\zeta_1 - \zeta_2$ coordinate space and transform both the triangle and the tetrahedron to this new 3D coordinate system because the coordinates of the triangle vertices become constants and only remaining variables are the tetrahedron vertices after such transformation.

The axis of the third coordinate ζ_3 of this 3D coordinate system is chosen to be orthogonal to ζ_1 and ζ_2 axes.

The resulting transformation is an affine transformation T:

$$\mathbb{R}^3 \xrightarrow{\ T\ } \mathbb{R}^3$$

where

$$\boldsymbol{\zeta} = \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} = T(\boldsymbol{x}) = \boldsymbol{M}^{-1}(\boldsymbol{x} - \boldsymbol{Q}_3) \tag{6.22}$$

where

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{Q}_1 - \boldsymbol{Q}_3 & \boldsymbol{Q}_2 - \boldsymbol{Q}_3 & (\boldsymbol{Q}_1 - \boldsymbol{Q}_3) \times (\boldsymbol{Q}_2 - \boldsymbol{Q}_3) \end{bmatrix} \tag{6.23}$$

The existence of this affine transformation relies on the invertibility of $\boldsymbol{M}$. The following assumption guarantees that $\boldsymbol{M}$ is invertible.

Assumption 2: *The triangular element has a non-zero area, i.e.,*

$$\left| (\boldsymbol{Q}_1 - \boldsymbol{Q}_3) \times (\boldsymbol{Q}_2 - \boldsymbol{Q}_3) \right| > 0 \tag{6.24}$$

This is a valid assumption because the triangular element is a side of a tetrahedral element whose volume is also guaranteed to be positive.

Now the triangle and tetrahedron are transformed by T.

6.3.2.2 Transformation of the triangle and the interior points of the triangle

T maps all points on the plane that contains the triangle to points on the plane $\zeta_3 = 0$. Particularly the triangle vertices $\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{Q}_3$ are mapped as

$$T(\boldsymbol{Q}_1) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, T(\boldsymbol{Q}_2) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, T(\boldsymbol{Q}_3) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.25}$$

The set of interior points of the triangle in the transformed space is

$$\Gamma_{tri} = \left\{ \boldsymbol{\zeta} = \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} ;\ \zeta_1, \zeta_2 > 0,\ \zeta_1 + \zeta_2 < 1,\ \zeta_3 = 0 \right\} \tag{6.26}$$

6.3.2.3 Transformation of the tetrahedron and the interior points of the tetrahedron

Let $\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_4$ be the transformed points of tetrahedron vertices $\boldsymbol{P}_1, \boldsymbol{P}_2, \cdots, \boldsymbol{P}_4$ by T:

$$\boldsymbol{p}_i = T\left(\boldsymbol{P}_i\right),\ i = 1,2,3,4 \tag{6.27}$$

The set of interior points of the tetrahedron in the transformed space is the intersection of four half spaces:

$$\Omega_{tet} = \bigcap_{i=1}^{4} \Omega_i \tag{6.28}$$

where

$$\Omega_i = \left\{ \boldsymbol{\zeta} \ ;\ b_{i,1}(\boldsymbol{V})\zeta_1 + b_{i,2}(\boldsymbol{V})\zeta_2 + b_{i,3}(\boldsymbol{V})\zeta_3 + b_{i,4}(\boldsymbol{V}) < 0 \right\},\ i = 1,2,3,4 \tag{6.29}$$

Each half space corresponds to a side of the tetrahedral element. The coefficients $b_{i,1}(\boldsymbol{V}), b_{i,2}(\boldsymbol{V}), \cdots, b_{i,4}(\boldsymbol{V})$ are all continuous functions of $\boldsymbol{V}$ and defined as

$$\begin{bmatrix} b_{1,1}(\boldsymbol{V}) \\ b_{1,2}(\boldsymbol{V}) \\ b_{1,3}(\boldsymbol{V}) \end{bmatrix} = (\boldsymbol{p}_3 - \boldsymbol{p}_1) \times (\boldsymbol{p}_2 - \boldsymbol{p}_1) \qquad b_{1,4}(\boldsymbol{V}) = \begin{bmatrix} b_{1,1}(\boldsymbol{V}) \\ b_{1,2}(\boldsymbol{V}) \\ b_{1,3}(\boldsymbol{V}) \end{bmatrix} \bullet \boldsymbol{p}_1$$

$$\begin{bmatrix} b_{2,1}(\boldsymbol{V}) \\ b_{2,2}(\boldsymbol{V}) \\ b_{2,3}(\boldsymbol{V}) \end{bmatrix} = (\boldsymbol{p}_2 - \boldsymbol{p}_1) \times (\boldsymbol{p}_4 - \boldsymbol{p}_1) \qquad b_{2,4}(\boldsymbol{V}) = \begin{bmatrix} b_{2,1}(\boldsymbol{V}) \\ b_{2,2}(\boldsymbol{V}) \\ b_{2,3}(\boldsymbol{V}) \end{bmatrix} \bullet \boldsymbol{p}_1$$

$$\begin{bmatrix} b_{3,1}(\boldsymbol{V}) \\ b_{3,2}(\boldsymbol{V}) \\ b_{3,3}(\boldsymbol{V}) \end{bmatrix} = (\boldsymbol{p}_3 - \boldsymbol{p}_2) \times (\boldsymbol{p}_4 - \boldsymbol{p}_2) \qquad b_{3,4}(\boldsymbol{V}) = \begin{bmatrix} b_{3,1}(\boldsymbol{V}) \\ b_{3,2}(\boldsymbol{V}) \\ b_{3,3}(\boldsymbol{V}) \end{bmatrix} \bullet \boldsymbol{p}_2$$

$$\begin{bmatrix} b_{4,1}(\boldsymbol{V}) \\ b_{4,2}(\boldsymbol{V}) \\ b_{4,3}(\boldsymbol{V}) \end{bmatrix} = (\boldsymbol{p}_1 - \boldsymbol{p}_3) \times (\boldsymbol{p}_4 - \boldsymbol{p}_3) \qquad b_{4,4}(\boldsymbol{V}) = \begin{bmatrix} b_{4,1}(\boldsymbol{V}) \\ b_{4,2}(\boldsymbol{V}) \\ b_{4,3}(\boldsymbol{V}) \end{bmatrix} \bullet \boldsymbol{p}_3 \tag{6.30}$$

6.3.2.4 The intersection of the tetrahedron interior and the triangle plane

Γ is the intersection of the triangle interior Γ_{tri} and the tetrahedron interior Ω_{tet}. Γ_{tri} is defined in the flat land $\zeta_3 = 0$. It is convenient to discuss the tetrahedron interior in the same

flat land. The flat land version of the tetrahedron interior Γ_{tet} is defined as the intersection of the plane $\zeta_3 = 0$ and Ω_{tet} :

$$\begin{aligned}\Gamma_{tet} &= \{\boldsymbol{\zeta};\ \zeta_3 = 0\} \cap \Omega_{tet} \\ &= \bigcap_{i=1}^{4} \left(\{\boldsymbol{\zeta};\ \zeta_3 = 0\} \cap \Omega_i \right) \\ &= \bigcap_{i=1}^{4} \Gamma_i \end{aligned} \tag{6.31}$$

where

$$\Gamma_i = \left\{ \boldsymbol{\zeta}\ ;\ b_{i,1}(\boldsymbol{V})\zeta_1 + b_{i,2}(\boldsymbol{V})\zeta_2 + b_{i,4}(\boldsymbol{V}) < 0, \zeta_3 = 0 \right\} \tag{6.32}$$

Due to **assumption 1**

$$\left| \begin{bmatrix} b_{i,1}(\boldsymbol{V}) \\ b_{i,2}(\boldsymbol{V}) \\ b_{i,3}(\boldsymbol{V}) \end{bmatrix} \right| > 0 \tag{6.33}$$

Now Γ_i is examined in four mutually exclusive separate cases:

Case 1: $b_{i,1}(\boldsymbol{V}) = b_{i,2}(\boldsymbol{V}) = 0,\ b_{i,4}(\boldsymbol{V}) > 0$

Case 2: $b_{i,1}(\boldsymbol{V}) = b_{i,2}(\boldsymbol{V}) = b_{i,4}(\boldsymbol{V}) = 0$

Case 3: $b_{i,1}(\boldsymbol{V}) = b_{i,2}(\boldsymbol{V}) = 0,\ b_{i,4}(\boldsymbol{V}) < 0$

Case 4: $\left|b_{i,1}(\boldsymbol{V})\right| + \left|b_{i,2}(\boldsymbol{V})\right| > 0$

In **case 1**, the condition $b_{i,1}(\boldsymbol{V})\zeta_1 + b_{i,2}(\boldsymbol{V})\zeta_2 + b_{i,4}(\boldsymbol{V}) < 0$ is false. Therefore, $\Gamma_i = \varnothing$. Therefore there is no intersection between the triangle and the tetrahedron; hence the penalty force is always zero, which is trivially a continuous function. In **case 2**, $\Gamma_i = \varnothing$ as well, but this condition may be broken by a small change in $\boldsymbol{V}$. This case, however, can be excluded because of the following assumption.

Assumption 3: *None of the tetrahedron's sides are coplanar with the triangle*.

This assumption is justified as follows. To make a tetrahedron's side coplanar with the triangle, all three vertices of the side must be exactly on the plane defined by the triangle. To

see such a case in an actual FEM computation is highly unlikely even if the continuous sequence of deformation in time is considered. If the initial condition is arranged to have coplanarity, the configuration can be perturbed slightly to escape from the situation.

In **case 3**, Γ_i is simplified to $\{\boldsymbol{\zeta}\ ;\ \zeta_3 = 0\}$, so Γ_i does not have any effect to the intersection Γ_{tet} and can be ignored.

Thus Γ_i in **case 1-3** can be all eliminated. Now Γ_{tet} can be treated as an intersection of all Γ_i's that fall into **case 4**:

$$\Gamma_{tet} = \bigcap_{i \in \mathrm{I}} \Gamma_i \tag{6.34}$$

where I is a set of indices of tetrahedron sides that are not parallel to the triangle, or

$$\mathrm{I} = \left\{ i\ ;\ i \in \{1,2,3,4\}, \left| b_{i,1}(\boldsymbol{V}) \right| + \left| b_{i,2}(\boldsymbol{V}) \right| > 0 \right\} \tag{6.35}$$

6.3.2.5 Definition of Γ

Finally, Γ is defined as the intersecting region of the triangle and the tetrahedron.

$$\Gamma = \Gamma_{tri} \cap \Gamma_{tet} \tag{6.36}$$

6.3.2.6 Definition of Γ', Γ'_{tet}, *and* Γ'_i

In the same manner as Γ, Γ_{tet}, and Γ_i, Γ', Γ'_{tet}, and Γ'_i are defined:

$$\Gamma' = \Gamma_{tri} \cap \Gamma'_{tet} \tag{6.37}$$

$$\Gamma'_{tet} = \bigcap_{i \in \mathrm{I}'} \Gamma'_i \tag{6.38}$$

$$\mathrm{I}' = \left\{ i\ ;\ i \in \{1,2,3,4\}, \left| b_{i,1}(\boldsymbol{V}') \right| + \left| b_{i,2}(\boldsymbol{V}') \right| > 0 \right\} \tag{6.39}$$

$$\Gamma'_i = \left\{ \boldsymbol{\xi}\ ;\ b_{i,1}(\boldsymbol{V}')\xi_1 + b_{i,2}(\boldsymbol{V}')\xi_2 + b_{i,4}(\boldsymbol{V}') < 0, \xi_3 = 0 \right\} \tag{6.40}$$

$\boldsymbol{V}'$ can be chosen close enough to $\boldsymbol{V}$ so that $\mathrm{I} = \mathrm{I}'$.

6.3.2.7 Bounding $area\left(\Gamma' \cap \bar{\Gamma}\right)$

The first half of **proposition iii** is $\lim_{V' \to V} area\left(\Gamma' \cap \bar{\Gamma}\right) = 0$. To prove this, finding a simple yet tight bound of $area\left(\Gamma' \cap \bar{\Gamma}\right)$ helps. First a simpler superset of $\Gamma' \cap \bar{\Gamma}$ is derived as follows:

$$
\begin{aligned}
&\Gamma' \cap \bar{\Gamma} \\
&= \left(\Gamma_{tri} \cap \Gamma'_{tet}\right) \cap \overline{\left(\Gamma_{tri} \cap \Gamma_{tet}\right)} \\
&= \left(\Gamma_{tri} \cap \Gamma'_{tet}\right) \cap \left(\overline{\Gamma_{tri}} \cup \overline{\Gamma_{tet}}\right) \\
&= \Gamma_{tri} \cap \left(\Gamma'_{tet} \cap \overline{\Gamma_{tet}}\right) \\
&= \Gamma_{tri} \cap \left(\bigcap_i \Gamma'_i \cap \overline{\bigcap_i \Gamma_i}\right) \\
&= \Gamma_{tri} \cap \left(\bigcap_i \Gamma'_i \cap \bigcup_i \overline{\Gamma_i}\right) \\
&= \bigcup_i \left(\Gamma_{tri} \cap \left(\left(\bigcap_j \Gamma'_j\right) \cap \overline{\Gamma_i}\right)\right) \\
&\subseteq \bigcup_i \left(\Gamma_{tri} \cap \Gamma'_i \cap \overline{\Gamma_i}\right) \qquad (6.41)
\end{aligned}
$$

Therefore, the area is bounded as

$$
\begin{aligned}
&area\left(\Gamma' \cap \bar{\Gamma}\right) \\
&\le area\left(\bigcup_{i \in \mathrm{I}} \left(\Gamma_{tri} \cap \Gamma'_i \cap \overline{\Gamma_i}\right)\right) \\
&\le \sum_{i \in \mathrm{I}} area\left(\Gamma_{tri} \cap \Gamma'_i \cap \overline{\Gamma_i}\right) \qquad (6.42)
\end{aligned}
$$

Now consider replacing Γ_{tri} with a simpler set $\Gamma_0 = \left\{\boldsymbol{\zeta} \; ; \; 0 < \zeta_1 < 1, \zeta_3 = 0\right\}$. Obviously $\Gamma_{tri} \subset \Gamma_0$. Therefore RHS of (6.42) is bounded as

$$
area\left(\Gamma_{tri} \cap \Gamma'_i \cap \overline{\Gamma_i}\right) \le area\left(\Gamma_0 \cap \Gamma'_i \cap \overline{\Gamma_i}\right) \qquad (6.43)
$$

Due to (6.34) and (6.35), at least one of $b_{i,1}(\boldsymbol{V})$ and $b_{i,2}(\boldsymbol{V})$ is non zero. Without loss of generality, $b_{i,2}(\boldsymbol{V}) \neq 0$ can be assumed. There are two cases to consider: $b_{i,2}(\boldsymbol{V}) > 0$ and $b_{i,2}(\boldsymbol{V}) < 0$.

First, the case $b_{i,2}(\boldsymbol{V}) > 0$ is considered. In this case, $\boldsymbol{V}'$ can be chosen close enough to $\boldsymbol{V}$ to satisfy $b_{i,2}(\boldsymbol{V}') > 0$. Γ_i' and $\overline{\Gamma_i}$ can be rewritten as

$$\Gamma_i' = \{\boldsymbol{\zeta} \; ; \; \zeta_2 < c_1'\zeta_1 + c_2'\} \tag{6.44}$$

$$\overline{\Gamma_i} = \{\boldsymbol{\zeta} \; ; \; \zeta_2 \geq c_1\zeta_1 + c_2\} \tag{6.45}$$

where

$$c_1 = -\frac{b_{i,1}(\boldsymbol{V})}{b_{i,2}(\boldsymbol{V})}, \; c_2 = -\frac{b_{i,3}(\boldsymbol{V})}{b_{i,2}(\boldsymbol{V})}, \; c_1' = -\frac{b_{i,1}(\boldsymbol{V}')}{b_{i,2}(\boldsymbol{V}')}, \; c_2' = -\frac{b_{i,3}(\boldsymbol{V}')}{b_{i,2}(\boldsymbol{V}')} \tag{6.46}$$

Then

$$\begin{aligned}
&\Gamma_0 \cap \Gamma_i' \cap \overline{\Gamma_i} \\
&= \{\boldsymbol{\zeta} \; ; \; 0 < \zeta_1 < 1, \; c_1\zeta_1 + c_2 \leq \zeta_2 < c_1'\zeta_1 + c_2'\} \\
&\subseteq \{\boldsymbol{\zeta}; 0 < \zeta_1 < 1, \\
&\quad \min(c_1, c_1')\zeta_1 + \min(c_2, c_2') \leq \zeta_2 \leq \max(c_1, c_1')\zeta_1 + \max(c_2, c_2')\}
\end{aligned} \tag{6.47}$$

In the second case, in which $b_{i,2}(\boldsymbol{V}) < 0$, the inequalities of (6.44) and (6.45) flip, but the RHS of the above inequality stays the same. Thus for the both cases,

$$\begin{aligned}
&area\left(\Gamma_0 \cap \Gamma_i' \cap \overline{\Gamma_i}\right) \\
&= area\left(\{\boldsymbol{\zeta} \; ; \; 0 < \zeta_1 < 1, \; c_1\zeta_1 + c_2 \leq \zeta_2 < c_1'\zeta_1 + c'\}\right) \\
&\leq area\left(\{\boldsymbol{\zeta} \; ; \; 0 < \zeta_1 < 1, \right. \\
&\qquad \left. \min(c_1, c_1')\zeta_1 + \min(c_2, c_2') \leq \zeta_2 < \max(c_1, c_1')\zeta_1 + \max(c_2, c_2')\}\right) \\
&= area\left(\{\boldsymbol{\zeta} \; ; \; 0 < \zeta_1 < 1, \right. \\
&\qquad \left. 0 \leq \zeta_2 < \left(\max(c_1, c_1') - \min(c_1, c_1')\right)\zeta_1 + \left(\max(c_2, c_2') - \min(c_2, c_2')\right)\}\right)
\end{aligned}$$

$$= area\left(\left\{\boldsymbol{\zeta}\ ;\ 0<\zeta_1<1,\right.\right.$$
$$\left.\left.0\le\zeta_2<\left|c_1-c_1'\right|\zeta_1+\left|c_2-c_2'\right|\right\}\right)$$
$$=\tfrac{1}{2}\left|c_1-c_1'\right|+\left|c_2-c_2'\right|$$
$$=\frac{1}{2}\left|\frac{b_{i,1}(\boldsymbol{V})}{b_{i,2}(\boldsymbol{V})}-\frac{b_{i,1}(\boldsymbol{V}')}{b_{i,2}(\boldsymbol{V}')}\right|+\left|\frac{b_{i,3}(\boldsymbol{V})}{b_{i,2}(\boldsymbol{V})}-\frac{b_{i,3}(\boldsymbol{V}')}{b_{i,2}(\boldsymbol{V}')}\right| \tag{6.48}$$

Combined with (6.43),

$$area\left(\Gamma_{tri}\cap\Gamma_i'\cap\overline{\Gamma_i}\right)\le\frac{1}{2}\left|\frac{b_{i,1}(\boldsymbol{V})}{b_{i,2}(\boldsymbol{V})}-\frac{b_{i,1}(\boldsymbol{V}')}{b_{i,2}(\boldsymbol{V}')}\right|+\left|\frac{b_{i,3}(\boldsymbol{V})}{b_{i,2}(\boldsymbol{V})}-\frac{b_{i,3}(\boldsymbol{V}')}{b_{i,2}(\boldsymbol{V}')}\right| \tag{6.49}$$

6.3.2.8 The limits of the areas

Noting $b_{i,2}(\boldsymbol{V})\neq 0$,

$$\lim_{\boldsymbol{V}'\to\boldsymbol{V}} area\left(\Gamma_{tri}\cap\Gamma_i'\cap\overline{\Gamma_i}\right)=0 \tag{6.50}$$

By (6.42) and (6.50),

$$\lim_{\boldsymbol{V}'\to\boldsymbol{V}} area\left(\Gamma'\cap\overline{\Gamma}\right)=0. \tag{6.51}$$

With a similar argument, it is also concluded that

$$\lim_{\boldsymbol{V}'\to\boldsymbol{V}} area\left(\overline{\Gamma'}\cap\Gamma\right)=0. \tag{6.52}$$

Thus **proposition iii** is concluded.

6.4 Penalty Factor and Impenetrability Constraints

6.4.1 Introduction

The nature of the penalty method inevitably allows at least a small amount of penetration. Although in the discretized arena, small penetration on the contact area is considered no more serious than gaps between surfaces, it is also important to have a procedure to control the amount of penetration under the desired tolerance. Intuitively, if the amount of penetration is larger than the desired tolerance, one can increase the penalty factor and try another run of the simulation. Obviously, this procedure can also be automated. This section

is meant to provide a theoretical justification for such a procedure: the goal of this section is to prove that, as the penalty factor increases, the solution approaches a point that also solves the hard constraint problem and the maximum penetration allowed approaches zero.

6.4.2 Hard Constraint Problem

The gap g is a function of the particle coordinates $\boldsymbol{X}$ and the configuration ϕ

$$g(\boldsymbol{X},\phi) \tag{6.53}$$

After discretization, the configuration is a finite dimension array $\mathbf{x}$, allowing g to be written

$$g(\boldsymbol{X},\mathbf{x}) \tag{6.54}$$

Then, the equilibrium problem under the impenetrability constraint can be written as

$$\begin{cases} \text{minimize elastic potential energy } W(\mathbf{x}) \\ \text{subject to } g(\boldsymbol{X},\mathbf{x}) \geq 0,\ \forall \boldsymbol{X} \in \partial V \end{cases} \tag{6.55}$$

where ∂V is the boundary of object V.

6.4.3 Penalty Problem

Now define

$$g^{-}(\boldsymbol{X},\mathbf{x}) = \min\left(0, g(\boldsymbol{X},\mathbf{x})\right) \tag{6.56}$$

Since $g^{-}(\boldsymbol{X},\mathbf{x})$ is a continuous function of $\boldsymbol{X}$, the impenetrability constraint $g(\boldsymbol{X},\mathbf{x}) \geq 0\ \forall \boldsymbol{X} \in \partial V$ is equivalent to

$$h(\mathbf{x}) \equiv \int_{\partial V} \left[g^{-}(\boldsymbol{X},\mathbf{x}) \right]^2 da = 0 \tag{6.57}$$

On the other hand, the penalty method discussed in chapter 5 is equivalent to

$$\text{minimize } W(\mathbf{x}) + \varepsilon_p h(\mathbf{x}) \tag{6.58}$$

where ε_p is the contact penalty factor. The goal of this section is to prove that as $\varepsilon_p \to \infty$, the solution of the penalty problem (6.58) approaches the solution of the hard constraint problem (6.55).

6.4.4 Definitions

To facilitate the proof, it is useful to define a few quantities.

First, the minimum energy of the constraint problem (6.55) is

$$\mu_0 = \min\left\{W(\mathbf{x}) : g(\boldsymbol{X}, \mathbf{x}) \geq 0,\ \forall \boldsymbol{X} \in \partial V\right\} \tag{6.59}$$

Positive integers are used for the penalty factor

$$\varepsilon_p = 1, 2, \cdots, k, \cdots \tag{6.60}$$

Then, a sequence of objective (energy) functions are defined:

$$\begin{aligned} W_1(\mathbf{x}) &= W(\mathbf{x}) + 1 \times h(\mathbf{x}) \\ W_2(\mathbf{x}) &= W(\mathbf{x}) + 2 \times h(\mathbf{x}) \\ &\cdots \\ W_k(\mathbf{x}) &= W(\mathbf{x}) + kh(\mathbf{x}) \\ &\cdots \end{aligned} \tag{6.61}$$

Let $\mathbf{x}_k$ be the minimizer of the k^{th} energy function

$$\mathbf{x}_k = \operatorname{argmin}\left(W_k(\mathbf{x})\right) \tag{6.62}$$

so $W_k(\mathbf{x}_k)$ is the minimum of $W_k(\mathbf{x})$.

6.4.5 Outline of the Proof

The proof proceeds in the following steps:

1. Prove that $W_k(\mathbf{x}_k)$ is nondecreasing:

$$W_k(\mathbf{x}_k) \leq W_{k+1}(\mathbf{x}_{k+1}) \tag{6.63}$$

2. Prove that $W_k(\mathbf{x}_k)$ is bounded by the minimum energy of constraint problem:

$$W_k(\mathbf{x}_k) \leq \mu_0 \tag{6.64}$$

3. Prove that the limit of the penalty energy is zero:

$$\lim_{k\to\infty} kh(\mathbf{x}_k) = 0 \tag{6.65}$$

4. The final goal: Prove that $\lim_{k\to\infty} \mathbf{x}_k$, the limit of the minimizer of the penalty problem, also solves the constraint problem (6.55).

6.4.6 Step 1

The inequality (6.63) is proved as follows:

$$\begin{aligned} W_{k+1}(\mathbf{x}_{k+1}) &= W(\mathbf{x}_{k+1}) + (k+1)h(\mathbf{x}_{k+1}) \\ &\geq W(\mathbf{x}_{k+1}) + kh(\mathbf{x}) \\ &= W_k(\mathbf{x}_{k+1}) \\ &\geq W_k(\mathbf{x}_k) \end{aligned} \tag{6.66}$$

6.4.7 Step 2

Let $\mathbf{x}_0$ be the solution of the problem (6.55) then

$$\mu_0 = W(\mathbf{x}_0) = W(\mathbf{x}_0) + kh(\mathbf{x}_0) = W_k(\mathbf{x}_0) \geq W_k(\mathbf{x}_k) \tag{6.67}$$

6.4.8 Step 3

It is easier to prove $\limsup_{k\to\infty} kh(\mathbf{x}_k) = 0$ instead of $\lim_{k\to\infty} kh(\mathbf{x}_k) = 0$. Since $kh(\mathbf{x}_0) \geq 0$, such a replacement is allowed. First assume the contrary that $\limsup_{k\to\infty} kh(\mathbf{x}_k) > 0$. Then there exists a positive number ε such that

$$\limsup_{k\to\infty} kh(\mathbf{x}_k) = 3\varepsilon > 0 \tag{6.68}$$

Now consider a sequence of integers $\{N_n;\ \ n = 1,2,\cdots\}$ such that

$$N_{n+1} > 4N_n \tag{6.69}$$

Especially, since $\limsup_{k\to\infty} kh(\mathbf{x}_k) = 3\varepsilon$, each integer in the sequence can be selected to satisfy

$$2\varepsilon < N_n h(\mathbf{x}_{N_n}) < 4\varepsilon, \qquad n = 1,2,\cdots \tag{6.70}$$

Then

$$
\begin{aligned}
W_{N_n}\left(\mathbf{x}_{N_n}\right)+\varepsilon \le W_{N_n}\left(\mathbf{x}_{N_{n+1}}\right)+\varepsilon &= W\left(\mathbf{x}_{N_{n+1}}\right)+N_n h\left(\mathbf{x}_{N_{n+1}}\right)+\varepsilon \\
&= W\left(\mathbf{x}_{N_{n+1}}\right)+\left(\frac{N_n}{N_{n+1}}\right)N_{n+1}h\left(\mathbf{x}_{N_{n+1}}\right)+\varepsilon \\
&< W\left(\mathbf{x}_{N_{n+1}}\right)+\left(\frac{1}{4}\right)N_{n+1}h\left(\mathbf{x}_{N_{n+1}}\right)+\varepsilon \\
&< W\left(\mathbf{x}_{N_{n+1}}\right)+2\varepsilon \qquad \left(\text{because } N_{n+1}h\left(\mathbf{x}_{N_{n+1}}\right)<4\varepsilon\right) \\
&< W\left(\mathbf{x}_{N_{n+1}}\right)+N_{n+1}h\left(\mathbf{x}_{N_{n+1}}\right) \qquad \left(\text{because } 2\varepsilon < N_{n+1}h\left(\mathbf{x}_{N_{n+1}}\right)\right) \\
&= W_{N_{n+1}}\left(\mathbf{x}_{N_{n+1}}\right)
\end{aligned}
\tag{6.71}
$$

$W_{N_1}\left(\mathbf{x}_{N_1}\right)$ is bounded, so

$$
\lim_{n\to\infty} W_{N_n}\left(\mathbf{x}_{N_n}\right) \le \lim_{n\to\infty} W_{N_1}\left(\mathbf{x}_{N_1}\right)+(n-1)\varepsilon = \infty \tag{6.72}
$$

This contradicts (6.64).

6.4.9 The Final Step

Let $\mathbf{x}_\infty$ be $\lim_{k\to\infty}\mathbf{x}_k$. Because of (6.65),

$$
h\left(\mathbf{x}_\infty\right)=0 \tag{6.73}
$$

Since $h\left(\mathbf{x}_k\right)=\int_{\partial V}\left[g^-\left(\boldsymbol{X},\mathbf{x}_k\right)\right]^2 da$,

$$
g\left(\boldsymbol{X},\mathbf{x}_\infty\right)\ge 0,\ \ \forall \boldsymbol{X}\in\partial V \tag{6.74}
$$

(6.74) states that $\mathbf{x}_\infty$ satisfies the constraint of the hard constraint minimization problem, (6.55). μ_0 is the minimum energy of the problem (6.55). So

$$
\mu_0 \le W\left(\mathbf{x}_\infty\right) \tag{6.75}
$$

On the other hand, by (6.61) and (6.64),

$$
W\left(\mathbf{x}_k\right) \le W_k\left(\mathbf{x}_k\right) \le \mu_0 \tag{6.76}
$$

Therefore

$$
W\left(\mathbf{x}_\infty\right) \le \mu_0 \tag{6.77}
$$

Thus $W\left(\mathbf{x}_\infty\right)=\mu_0$, i.e., $\mathbf{x}_\infty$ is a solution of the constraint problem (6.55).

6.5 Numerical Example of Convergence

Earlier in this chapter, the proposed algorithm was examined by analytical methods. The most important result that was obtained there is that the contact forces used in the algorithm are continuous and consequently that the solution of the nonlinear system actually exists. However, what ultimately matters is whether the algorithm converges to a solution. While it is very difficult to guarantee convergence, it is possible to test the algorithm with an example that conventional methods fail to handle. In this section, the proposed algorithm is numerically compared with a conventional method.

6.5.1 A conventional point collocation algorithm

Conventional algorithms are characterized by point collocation for the purpose of integrating the contact force. The definition of point collocation (5.14) is repeated here:

$$\boldsymbol{F}_a^{cont} = \varepsilon_p g\left(\boldsymbol{X}_a\right) \left.\frac{\partial g}{\partial \boldsymbol{x}}\right|_{\boldsymbol{X}=\boldsymbol{X}_a} \int_{\partial v} N_a da \tag{6.78}$$

A point collocation algorithm can be easily created by modifying the proposed algorithm. As shown in (5.47)~(5.51), the proposed algorithm integrates the contact force over the intersection region

$$\int_\Gamma f\, da = 2A_{tri} \int_\Gamma f\, d\zeta_1 d\zeta_2 \tag{6.79}$$

The point collocation approximates this integration by the sum of samples at the three vertices of the triangle.

$$\int_\Gamma f\, da \approx 2A_{tri} \int_\Gamma f \left[\delta\left(\zeta_1\right)\delta\left(\zeta_2\right) + \delta\left(\zeta_1 - 1\right)\delta\left(\zeta_2\right) + \delta\left(\zeta_1\right)\delta\left(\zeta_2 - 1\right)\right] d\zeta_1 d\zeta_2 \tag{6.80}$$

6.5.2 The test case

Figure 25 illustrates the test case. In the initial configuration, two objects are intersecting. **Figure 26** shows the two objects apart. The pointy part of the object on the right is penetrating into the concave edge of the object on the left. This is a typical case that causes contact chatter.

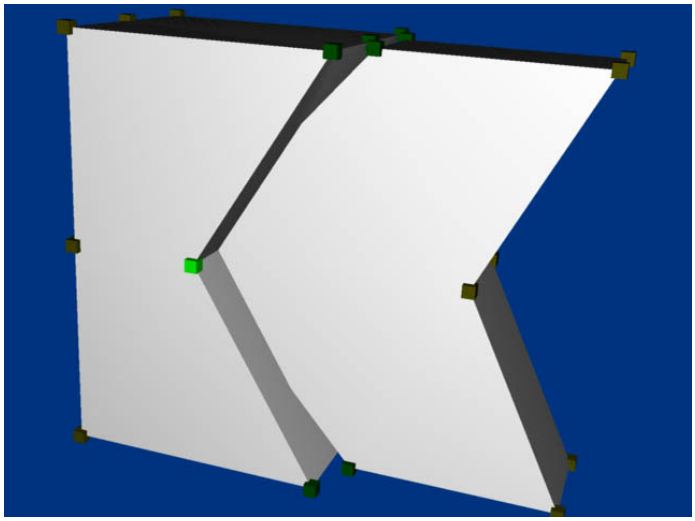

Figure 25: Two intersecting objects

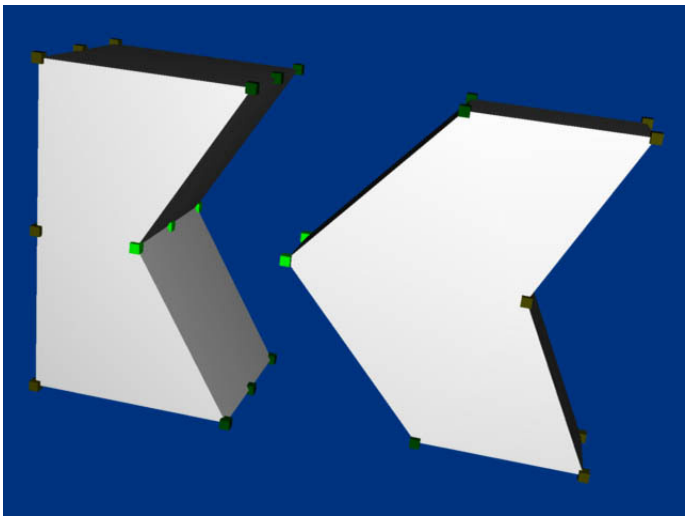

Figure 26: The same objects separated to show the geometry.

The material depths at nodal points are assigned such a way that two objects approximate cross-sections of two intersecting objects. **Figure 27** shows color coded depth fields. The right sides of the object on the left and the left sides of the object on the right have zero depth values so that they approximate contacting boundary surfaces. **Figure 28** reveals individual elements of the two finite element meshes.

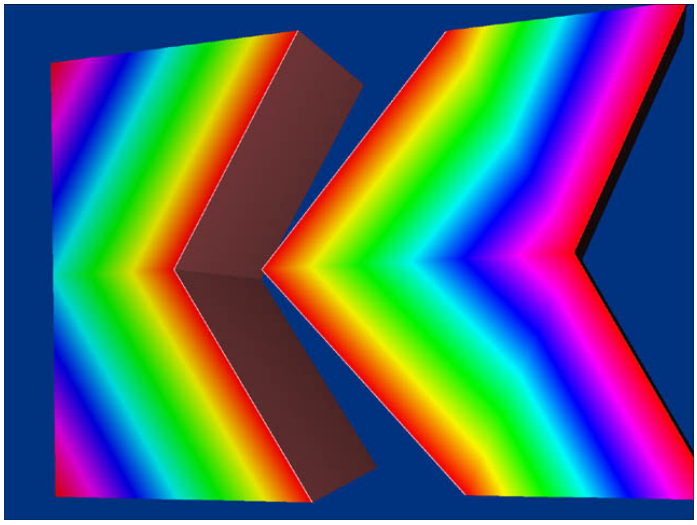

Figure 27: Material depth of two objects. The gradient of depth is discontinuous on the border of the upper and lower parts.

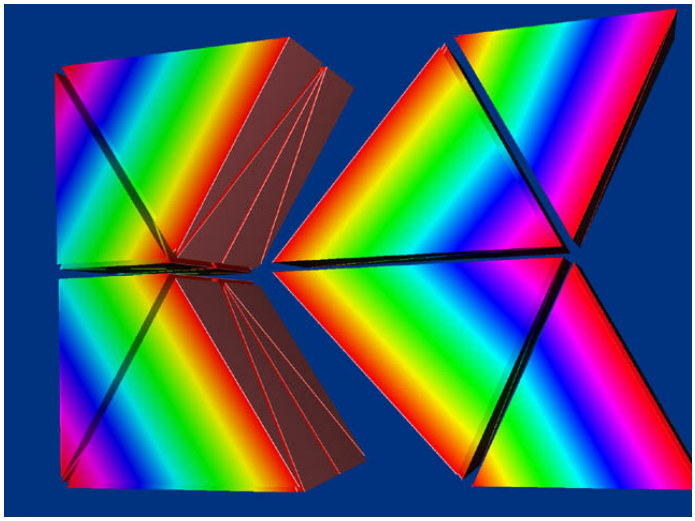

Figure 28: Tetrahedral meshing for two objects.

In **Figure 27** and **Figure 28**, the discontinuous borders of depth gradient are visible between the upper and lower parts of the objects. The directions of contact forces are the same as the gradient directions. As the position of sampling moves from the upper part to the lower part, the contact forces change abruptly (see **Figure 29**).

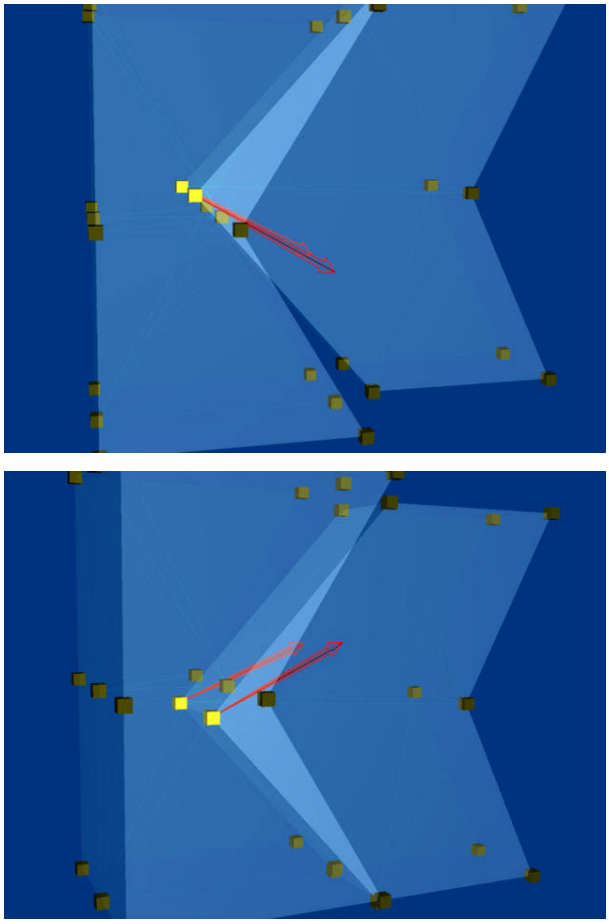

Figure 29: Contact penalty forces calculated by point collocation. The contact forces (red arrows) jumps as the sampling points (yellow cubes) cross the discontinuous border between upper and lower parts of the object.

Now if the new algorithm is used, the contact forces change continuously (see **Figure 30**).

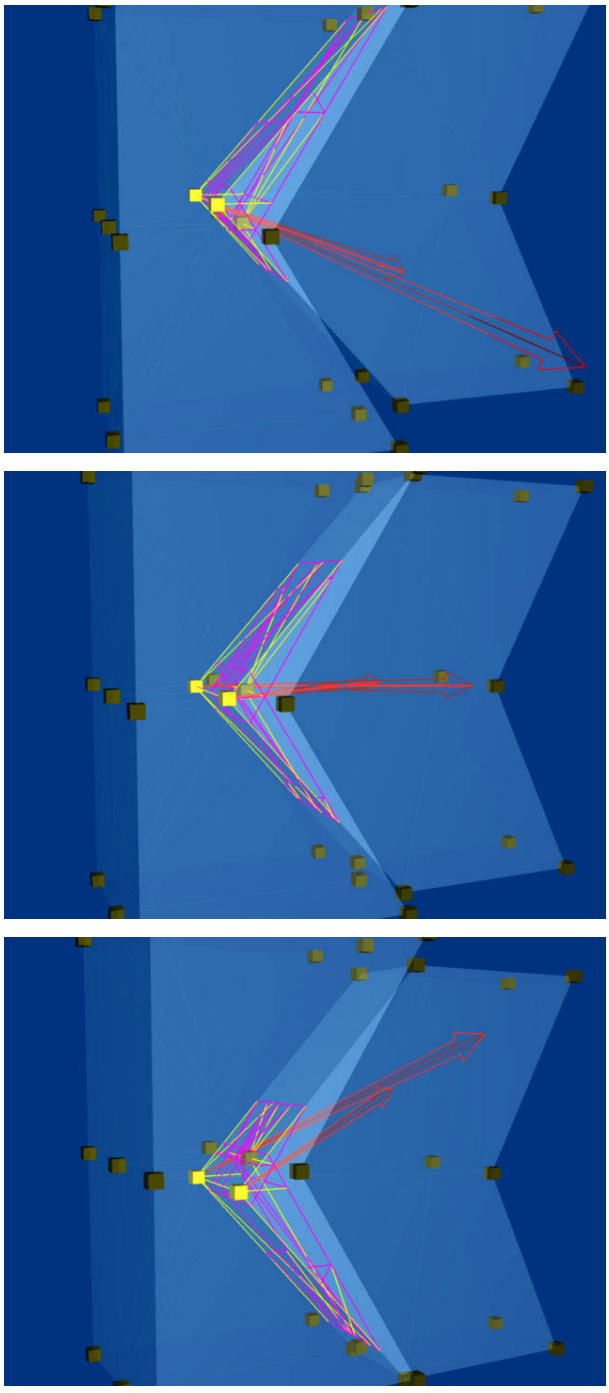

Figure 30: Contact penalty forces calculated by area integration. Contact forces are integrated over the intersecting area (rendered as wireframes). The contact forces change continuously as the object on the left moves down.

6.5.3 Comparison of residual force convergence

How does this difference in continuity affect the performance of algorithms? Below, the convergence characteristics of two algorithms are examined. The simulations are performed as dynamic analyses. For each time step of the dynamic analysis, the Newton-Raphson method is used to solve the motion equation. To examine the effect of penalty factors, 4 different values (ε_p=10, 100, 1000, and 10000) are used. **Figure 31**, **Figure 32**, **Figure 33**, and **Figure 34** plot the convergence graph for the 4 values. The horizontal axis represents the total number of Newton iterations from the start of the simulation. The vertical axis represents the magnitude of the residual force in a logarithmic scale. The magnitude of the residual is computed as the L_2 norm of the residual forces.

For every choice of penalty factor, the contact chatter (oscillation) is evident for the point collocation algorithm and convergence is not achieved. On the other hand, the area integration algorithm, quickly converges to low residual states. As the penalty factor increases, the problem becomes numerically challenging. But even with an extremely high penalty factor (ε_p=10000), the area integration algorithm achieves convergence. The rates of the convergence are overall linear reflecting the exponential dissipation of kinetic energy.

Figure 35 shows all the convergence graphs in one place. Note that the area integration with ε_p=10000 even outperforms the point collocation with ε_p=10.

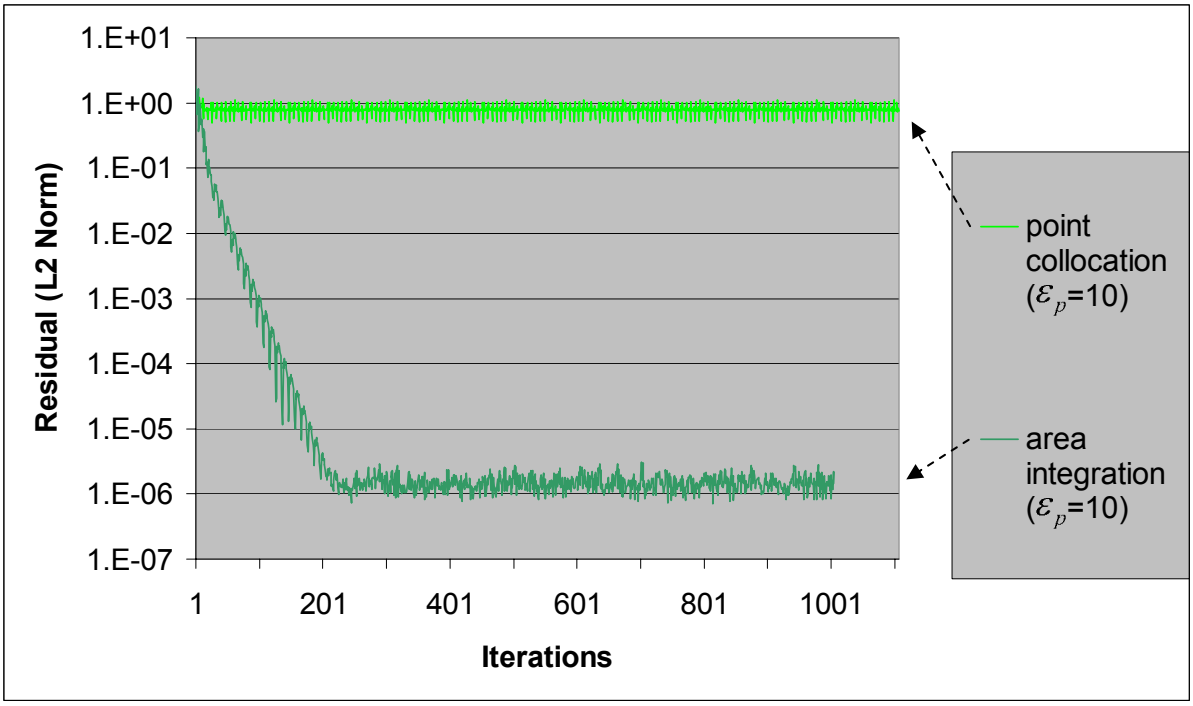

Figure 31: Comparison of convergence 1. The penalty factor is set as ε_p=10. The residual for the point collocation algorithm stays just below 1.00E+00, while the residual for the area integration algorithm goes down to a very low level until it hits the limit due to the floating point truncation error. The area integration algorithm converges to a visually stable geometry in just a few iterations, but the algorithm was kept running to examine further convergence characteristics.

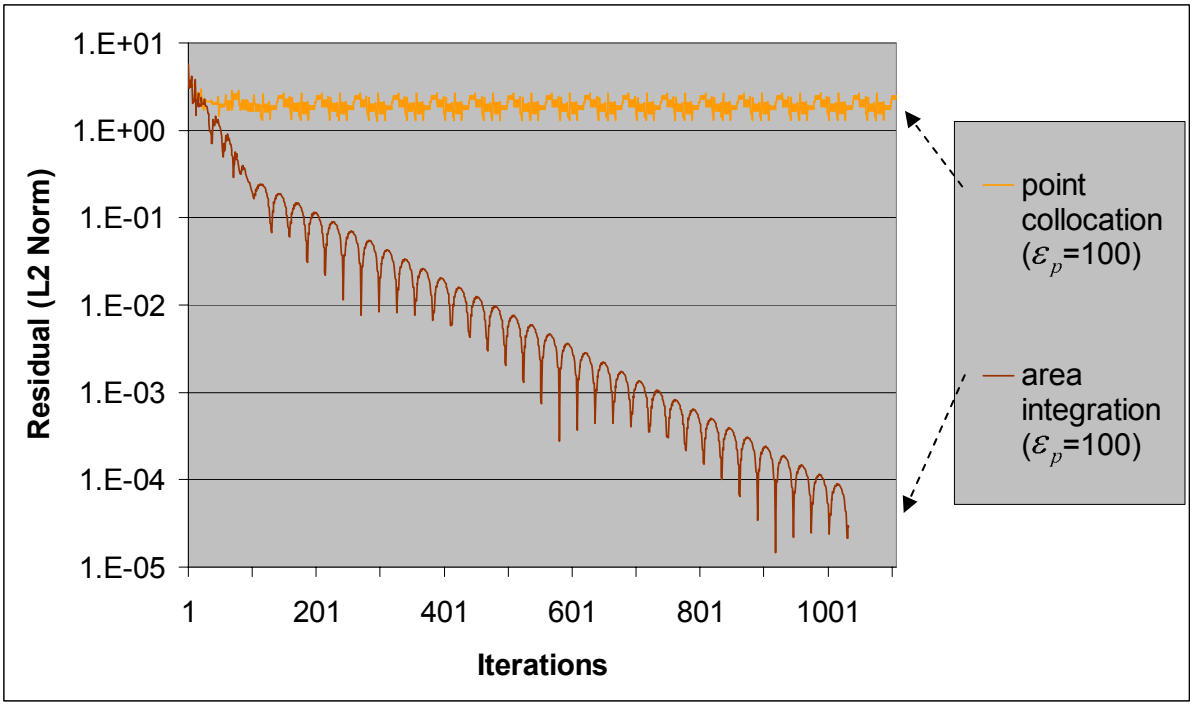

Figure 32: Comparison of convergence 2. The penalty factor is set as ε_p=100. The residual for the point collocation algorithm stays above 1.00E+00, while the residual for the area integration algorithm goes down to a very low level. After about 100 iterations, the area integration algorithm does not make a visible change of geometry.

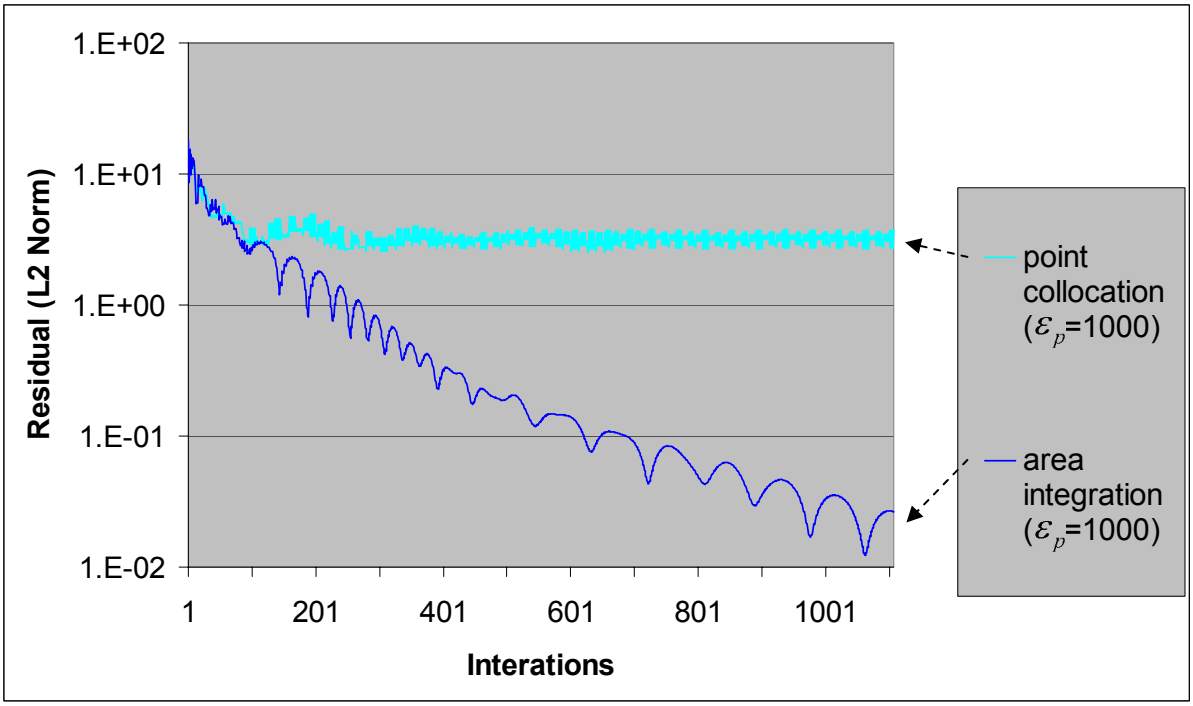

Figure 33: Comparison of convergence 3. The penalty factor is set as ε_p=1000. The residual for the point collocation algorithm stays between 1.00E+00 and 1.00E+01, while the residual for the area integration algorithm goes down to a low level. After about 200 iterations, the area integration algorithm does not make a visible change to the geometry.

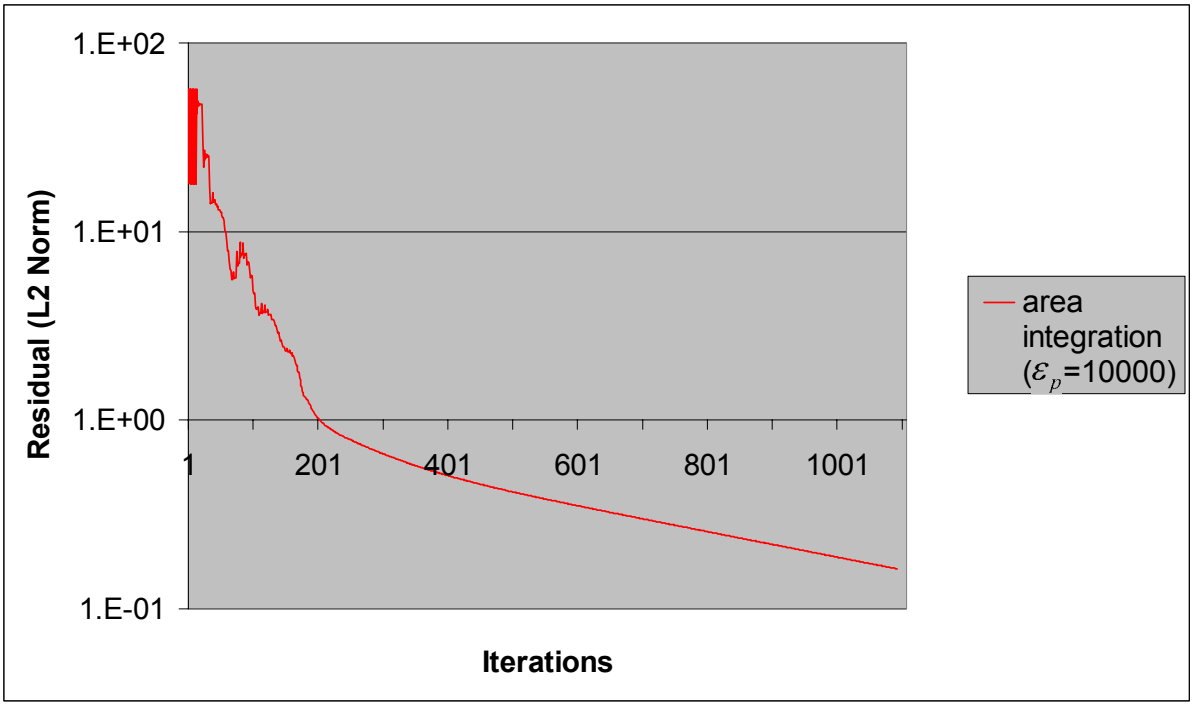

Figure 34: Comparison of convergence 4. The penalty factor is set as ε_p=10000. Because of the very high penalty value, the point collocation algorithm fails, while the residual for the area integration algorithm goes down to a low level. After about 250 iterations, the algorithm does not make a visible change to the geometry.

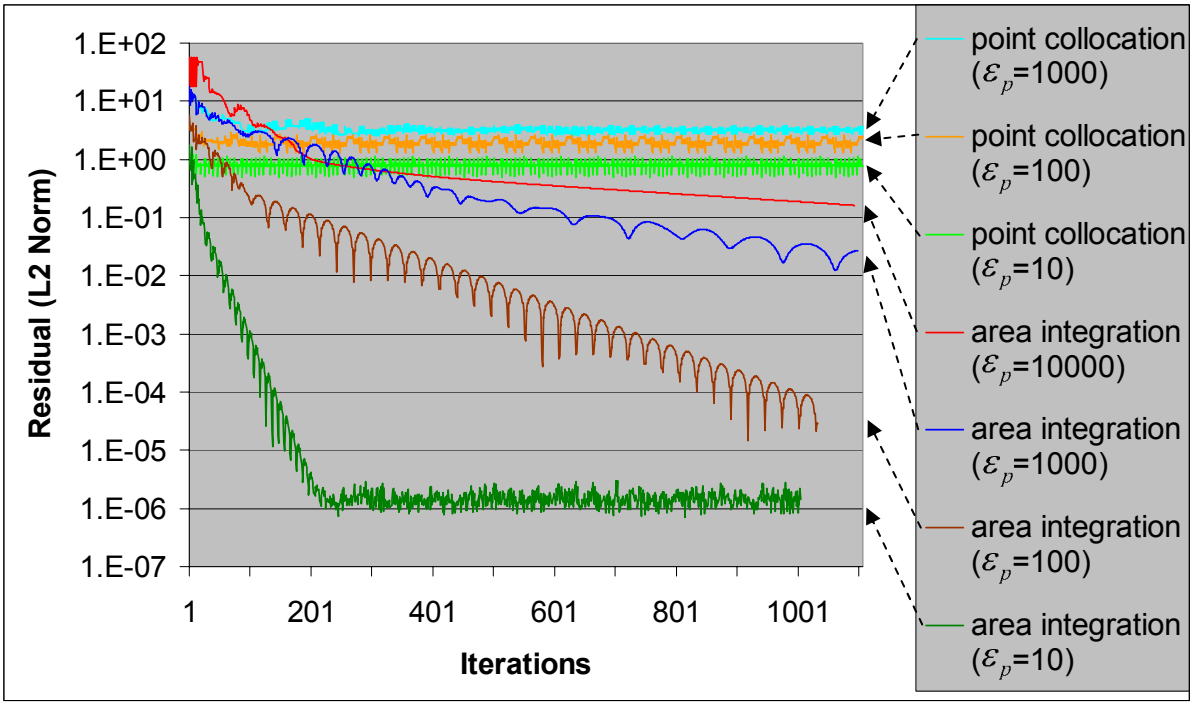

Figure 35: All convergence graphs in one place. The area integration converges in all cases, while the point collocation always fails to do. The point collocation algorithm becomes very unstable with ε_p=10000, so the convergence data is not available. Note that the area integration with ε_p=10000 even outperforms the point collocation with ε_p=10. It is also evident that the larger the penalty factor is, the slower the convergence is.

6.5.4 Penalty-penetration relationship

Table 1: Relationship between penalty factor and maximum penetration.

Penalty Factor (ε_p)	Point Collocation	Area Integration
0	0.441	0.441
10	0.345	0.329
100	0.231	0.212
1000	0.055	0.051
10000	—	0.014

As seen in the previous section, a large penalty factor slows down the convergence. But with a small penalty factor, the penetration between objects may be too large. **Table 1** shows the

relationship between the penalty factor and the final maximum penetration. There is no definite rule to decide a penalty factor. It should be determined based on the maximum penetration allowed in a particular application and desirable performance.

7 APPLICATION TO THE HUMAN BODY

7.1 The goal of the simulation

The purpose of this chapter is to demonstrate that the proposed algorithm is suitable and robust in simulating the deformation of anatomical models. Ultimately, my algorithm should be a part of a biomechanical simulation system that can accurately predict the mechanical deformations of human and animal bodies. However, accurate simulation, in the sense that one can confidently control the numerical error compared to real subjects, is quite difficult at this point. The difficulty is not just due to the problems in computational methods, but also (and more importantly) due to the premature state of the art of biomechanics; there are great difficulties in building mathematical models of real biological tissues. First of all, as described in section 3.6, the material models (constitutive laws) of tissues are not yet established. Even if it could be assumed that a simple constitutive law can be applied, there is no equipment or method that can measure heterogeneous in vivo material properties of tissues. Thus the aim of this section is not to present an example of accurate simulation but to demonstrate the advantage of the proposed algorithm as a computational method in a setting reasonably close to what future data collection procedures will provide. In other words, the simulation results in this section are meant to give a glimpse of what my computational method can offer once the advancement of biomechanics will enable the construction of accurate models of biological systems.

To make the simulation setting "reasonably close the future simulation," four conditions were set for the model construction:

1. The model's geometry is based on real anatomical data.

2. The model encompasses mechanically diverse tissues ranging from hard bones, ligaments, and tendons to softer muscles and connective tissues. In other words, the model must contain a broad range of material properties.
3. Complex contacts, including self contact, simultaneously occur in the model.
4. The materials have qualitative properties that are known to be typical for biological tissues such as nonlinearity and anisotropy.

In typical biomechanical analyses, very specific phenomena are of interest. In such cases, only specific tissues are modeled. For example, if one is interested in the cartilage of knee joints, skin and fat around the knee joint are not necessarily important. Since I am interested in more general applications, it is desirable to simulate all kinds of tissues at once.

As mentioned earlier, the goal of my algorithm is to handle contacts, especially complex self contacts. Therefore, it is best tested on the skin folds around flexing joints. The wrinkles around joints are particularly important in realistic image synthesis. Among many joints, the knee was chosen for the following reasons:

1. Since there are great biomedical and biomechanical interests on knees and significant effort has been applied to segment the tissues around knees, there are more 3D models available.
2. Since knees are larger than most of other joints, higher segmentation resolution can be achieved.

In the rest of this section, the modeling of a right human leg and a simulation of the knee flexion are described.

7.2 Leg and Knee Anatomy

The skeleton of the human leg includes the femur (thigh bone), the tibia (shin bone), the fibula, and the patella (knee cap). The largest muscle group in the human leg is the quadriceps femoris, the extender of the knee joint. My modeling effort concentrates on the quadriceps and other connective tissues directly related to its extension function. The quadriceps is made of 4 muscles; the rectus femoris, the vastus lateralis, the vastus

intermedius, and the vastus medialis. The rectus femoris originates from the spine and the hipbone and is connected to the patella via the quadriceps femoris tendon. The patella is then connected to the tibia via the patella ligament. The other 3 muscles originate from a broad area of the femur surface and are inserted (connected) to the quadriceps femoris tendon, the patella, and the patella ligament. The function of the quadriceps is the extension of the knee joint.

7.3 Geometric Modeling

7.3.1 Visible Human Male Data and Surface Extraction

The visible human data set consists of CT, MRI, and color images of male and female cadavers. The leg model described in this section was created based on color images of the male data set. The color images are color photographs of axial cross sections taken every 1mm interval. Slicing of the frozen cadaver, photographing, and image scanning were all performed at the University of Colorado Health Sciences Center. There are a total of 1871 cross-sections. Each color image of the male data set has 2048 × 1216 resolution with 24-bit color depth. To construct a volumetric image, the 3D relationships between slices must be determined. There are no marks on the images to automate this registration procedure. So each image was manually shifted in the image plane until the resulting volumetric image looked anatomically correct. The segmentation, classification of pixels to organs, is another step that requires human intervention. Each color image is painstakingly segmented by anatomists. The resulting mask image volume was processed by an isosurface extraction algorithm, and the boundary of each anatomical part was generated as triangular meshes. The registration, segmentation, and surface extraction were also performed at the Health Sciences Center. **Figure 36** shows all the extracted surfaces. There are approximately 35,000 vertices and 70,000 triangles.

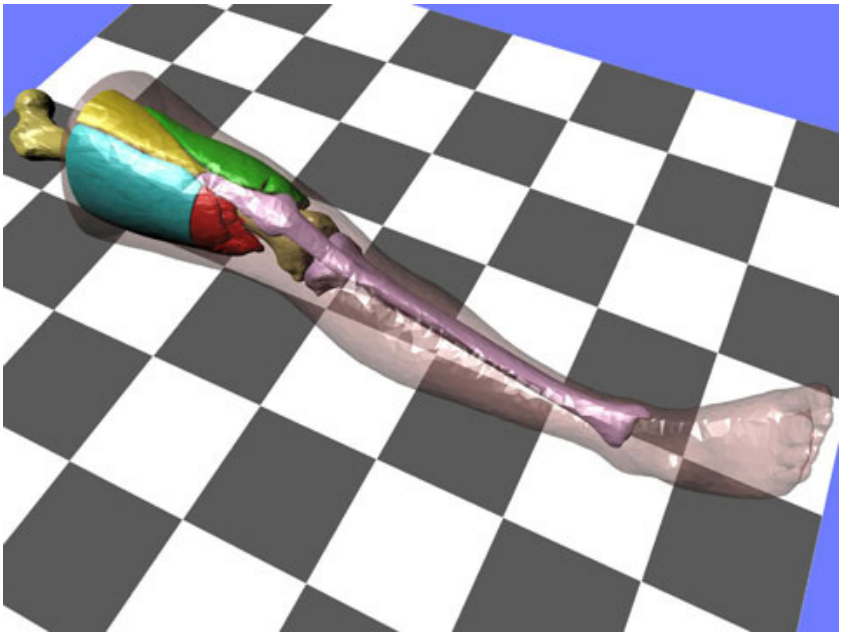

Figure 36: Surface models of organs extracted from the right leg of the Visible Human Male dataset. Organs include the skin, the femur, the tibia, the patella, and the quadriceps. The model contains approximately 35,000 vertices and 70,000 triangles.

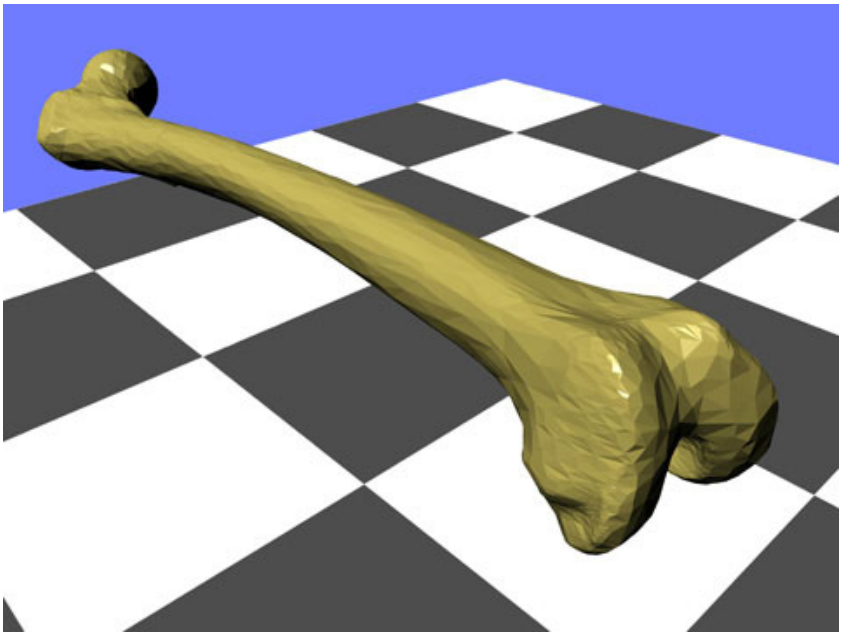

Figure 37: The extracted surface model of femur.

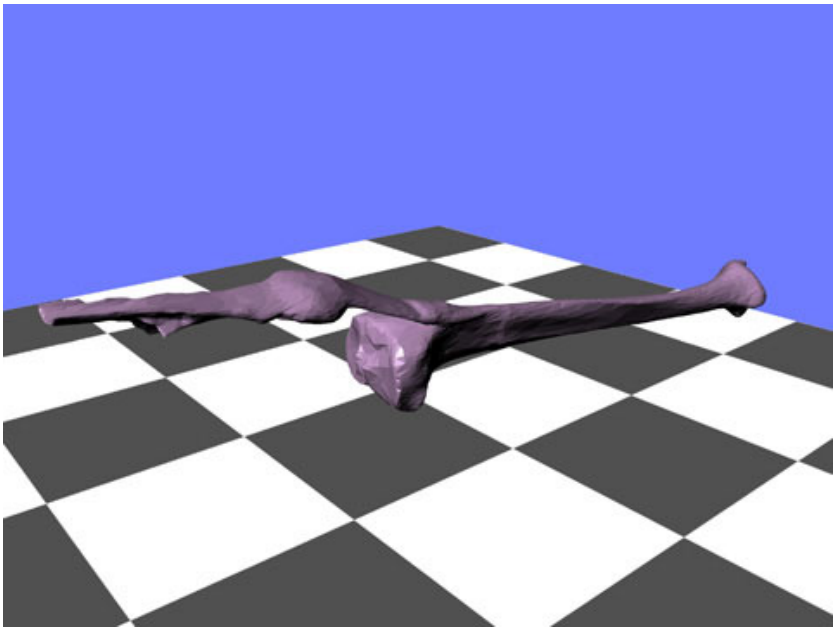

Figure 38: Extracted surface of the tibia, the patella, the patella ligament, and tendons that connect the patella and quadriceps.

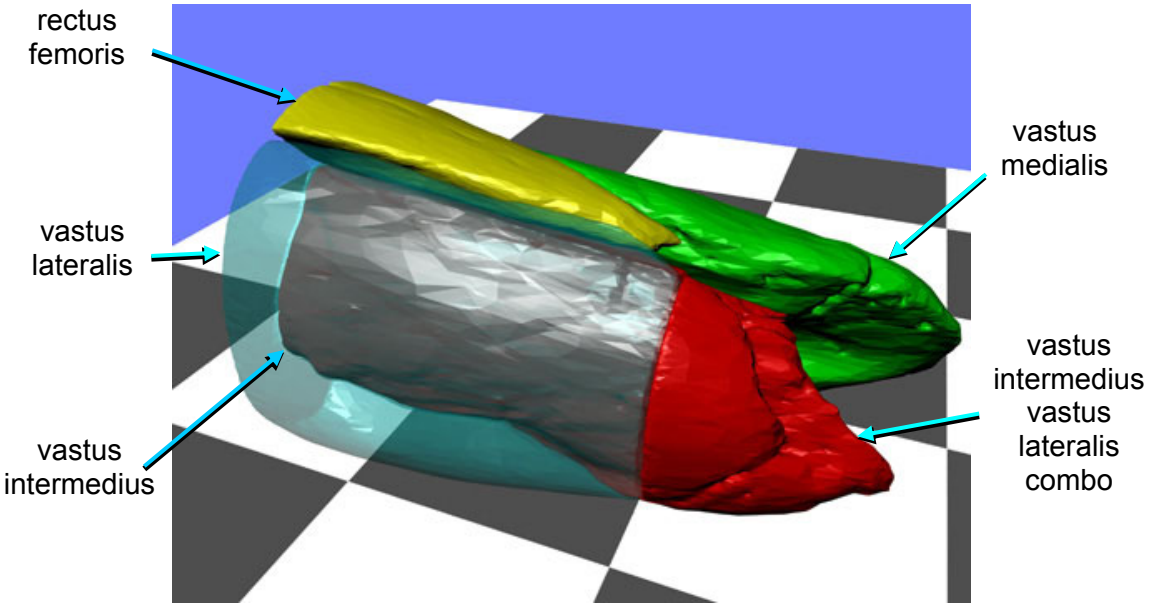

Figure 39: Extracted surfaces of the quadriceps, which includes rectus femoris (yellow), the vastus lateralis (transparent blue), the vastus intermedius (red), and the vastus medialis (green). On the patella side, the vastus lateralis and intermedius are segmented as a "combo," and are included in the vastus intermedius in the extraction.

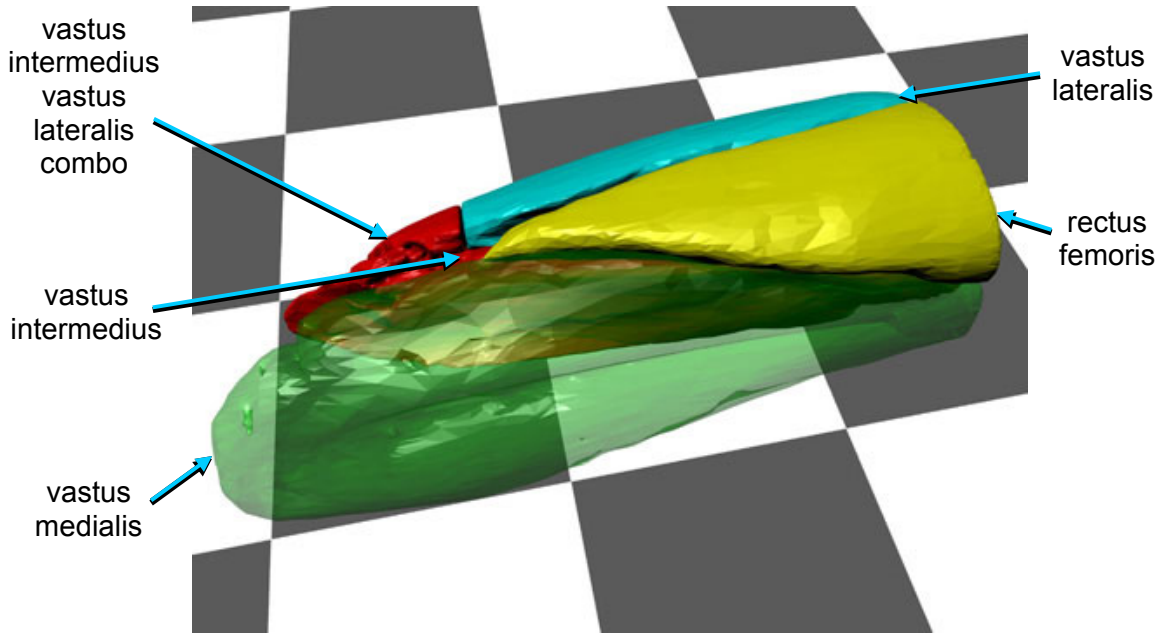

Figure 40: The quadriceps. This is the same as the model in Figure 39. The coloring is as follows: the vastus lateralis: blue, the vastus intermedius + the vastus lateralis- intermedius combo: red, the rectus femoris: yellow, and the vastus medialis: transparent green.

7.3.2 Simplification and Parts Assembly

The extracted surfaces, however, have many problems. First of all, there are too many triangles around fine, high curvature, structures. Various mesh simplification algorithms, as well as a custom resampling program, were applied to remove those fine details.

Muscles and bones must be properly connected to build a mechanically sound anatomical model. But figuring out this mechanical connectivity is a big issue. The extracted surface models are based on the image segmentation, which is merely meant to locate parts that have distinct anatomical names. But an anatomical name does not necessarily correspond to a geometrically distinct part. Often a broad collection of muscle or tendon tissues share a single name. Also, two mechanically separate tissues, such as two tendons, may be right next to each other. On the color images, tendons (and all other collagen rich connective tissues) are simply white pixels. Finding a border between those connective tissues is often impossible. It is very difficult to extract the necessary information from only Visible Human Data. Therefore, the missing information must be supplied by a manual modeling process. I used Maya's [Anderson 1999] Polygonal Modeling operations not only for stitching together

all parts but also for creating missing tendons. At this point, I also merged the vastus lateralis, intermedius, and medialis together. This is due to the observation that, since those three muscles have very similar functions, the boundaries between them are not sliding, so there is no need to treat them as mechanically separate entities.

The tendon between the patella and the rectus femoris is quite distinct, so they were connected with relatively little effort. The vastus medialis's attachment to the patella is also quite obvious; therefore, the stitching was not difficult. However, the tendon between the vastus lateralis-intermedius bundle and the patella is completely missing in the extracted surface data. Therefore, a tendon was created based on illustrations and descriptions in anatomy books.

Another issue is modeling fatty connective tissues under the skin. Those connective tissues are everywhere in the human body, and surrounding numerous muscles and tendons. Roughly speaking, if I subtract bones, muscles, tendons, and ligaments from the leg skin geometry, the connective tissue attached to the skin will be obtained. Maya's solid modeling function (difference operation in this case) can be used for this purpose. But since there are very small gaps between muscles and between muscles and bones, such an operation would create very fine structures, which would be an unnecessary burden for the later processes. To avoid it, I made a "shrink wrap" program that computes a surrounding volume out of several objects. The shrink wrap program was applied to all parts of the knee except for the skin surface. After that the surrounding "hull" was subtracted from the skin surface to generate the skin-fat layer.

7.3.3 FEM Mesh Generation

Afterwards, a tetrahedral mesh was generated from the triangle mesh boundaries using SolidMesh [Marcum 1995]. The mesh contains about 10,000 nodes, 10,000 triangular elements, and 40,000 tetrahedral elements.

7.3.4 Material depth assignment

The material depth of each internal nodal point is computed by finding the minimum of the distances to all triangles on the boundaries except for triangles adjacent to the nodal point.

7.4 Mechanical Properties

7.4.1 Material Assignment

Various material parameters are assigned to the tetrahedral elements in order to approximate the mechanical properties of different parts (**Table 2**). The Mooney-Rivlin and Veronda models were both applied. To approximate the gradual property transition from muscles to tendons, two intermediate materials (green and magenta) are used between the muscle material (yellow) and the tendon material (blue).

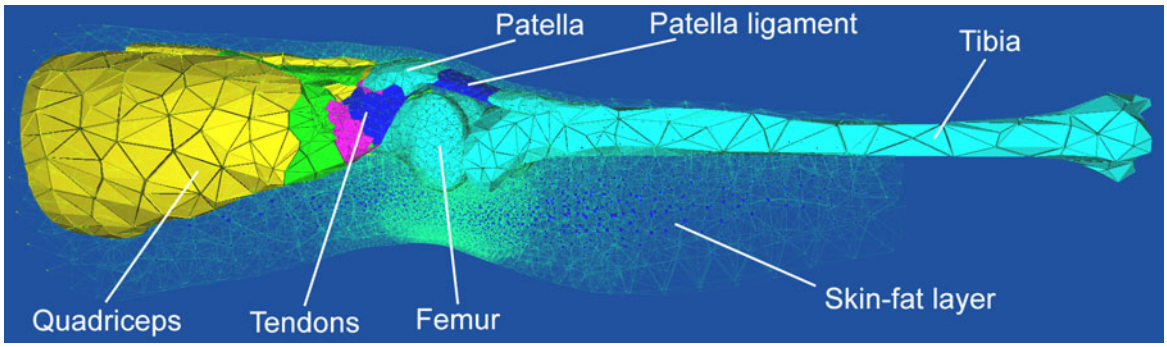

Figure 41: The finite element mesh. Colors indicate different material properties.

Table 2: Material Assignment

Color	*Material model*	*Material Properties*
Cyan	**Rigid**	-
Blue	**Veronda**	α=1400.0, β=8.0, a=1000.0
Magenta	**Veronda**	α=350.0, β=8.0, a=500.0
Green	**Veronda**	α=14.0, β=8.0, a=200.0
Yellow	**Veronda**	α=7.0, β=8.0, a=100.0
Transparent	**Mooney-Rivlin**	C_1=5.0, C_2=1.0, a=50.0

7.4.2 Interface Property

All the boundaries are treated as frictionless interfaces except for the inner part of the tibia, which is attached to the skin-fat layer. This assumption is justified because the friction between lubricated organ surfaces is known to be small [Malcolm 1976, Moro-oka 1999]. Friction is further discussed in Section 10.1.5. In the actual body, the quadriceps has a broad attachment to the femur. However, in the simulation model, they meet on a sliding interface as well since the objective of the simulation is to test the algorithm by using examples with a large amount of sliding contact.

7.5 Simulation

7.5.1 Boundary Conditions

The femur is fixed in space. The cross section of the thigh is constrained so that it can only move on the cutting plane. The tibia is known to translate slightly as it rotates, but the amount of translation is not large for a limited amount of rotation. Therefore, in this simulation, the tibia is rotated around an axis in the knee joint, which is specified by an anatomy expert. These positional constraints constitute a displacement boundary condition. The tibia's total 150-degree rotation was divided into 50 three-degree intervals and the algorithm was applied to each interval to generate deformations. In this simulation, external forces such as gravity were not applied.

7.5.2 The Result

Figure 42 shows snapshots of the simulation sequence. The left column shows the skin deformation, and the right column is the same sequence with wire frame rendering revealing the internal structure. Anatomically expected movements are automatically generated by the simulation. The patella's movement is particularly noteworthy. **Figure 43** shows the movement of the patella. Even though the movement of the patella is not prescribed, it slides over the head of the femur (between the lateral and medial condyles) as it is pulled by the rotating tibia via the patella ligament. **Figure 44** shows the result at the maximum flexion. The

left image shows the skin geometry. The right image shows the cut-away view. The colors encode the material depth value. The folded skin-fat layer inside the knee joint is highly compressed, yet the amount of penetration is not visible. **Figure 45** shows the complex wrinkling pattern of the skin. The curvature of the skin surface is extremely high where the folding occurs. **Figure 46** shows cut-away views of the same part. Again the penetration is visually undetectable. Finally, **Figure 47** shows the bent knee in the context of the entire visible human data.

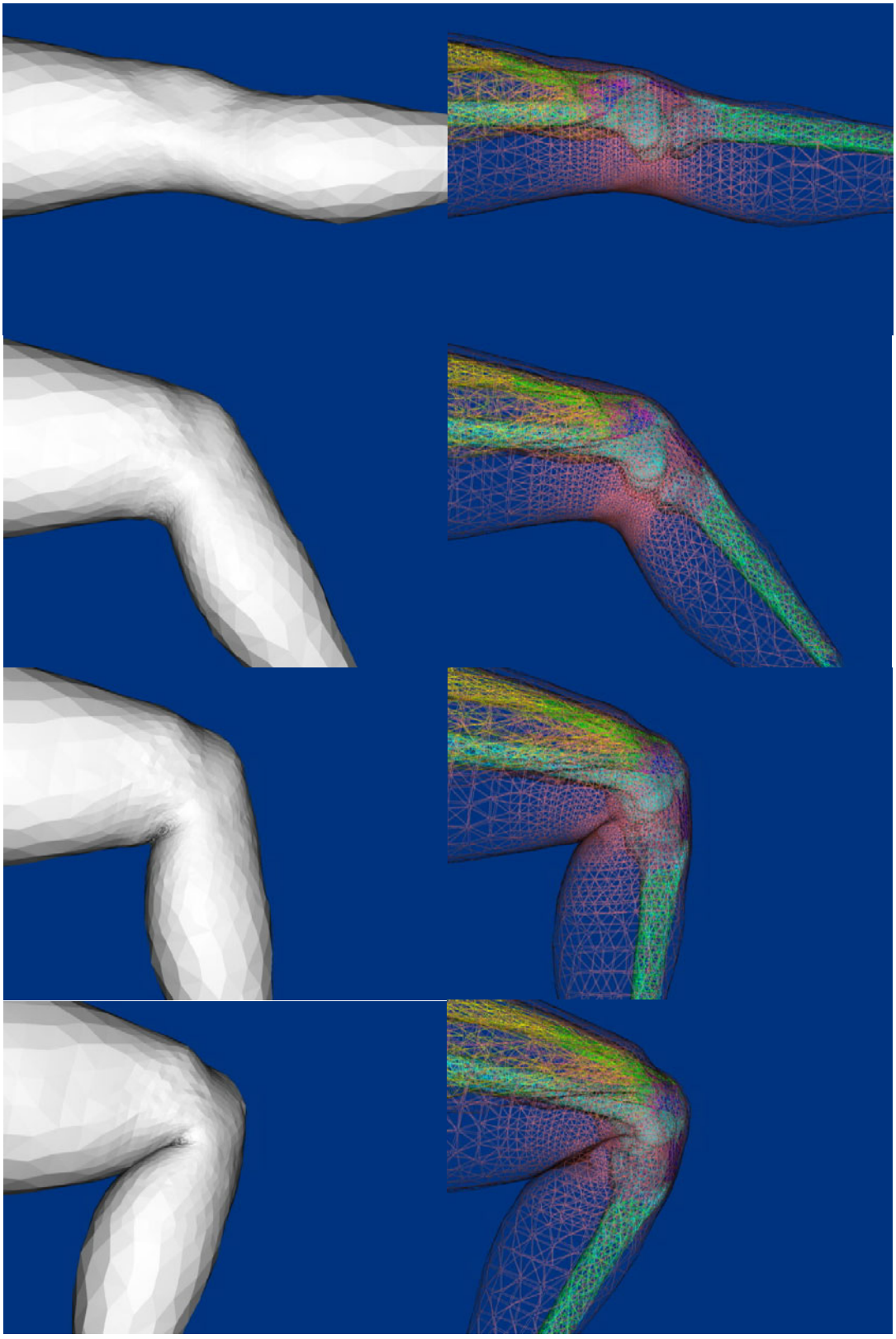

Figure 42: Simulation sequence of knee flexion.

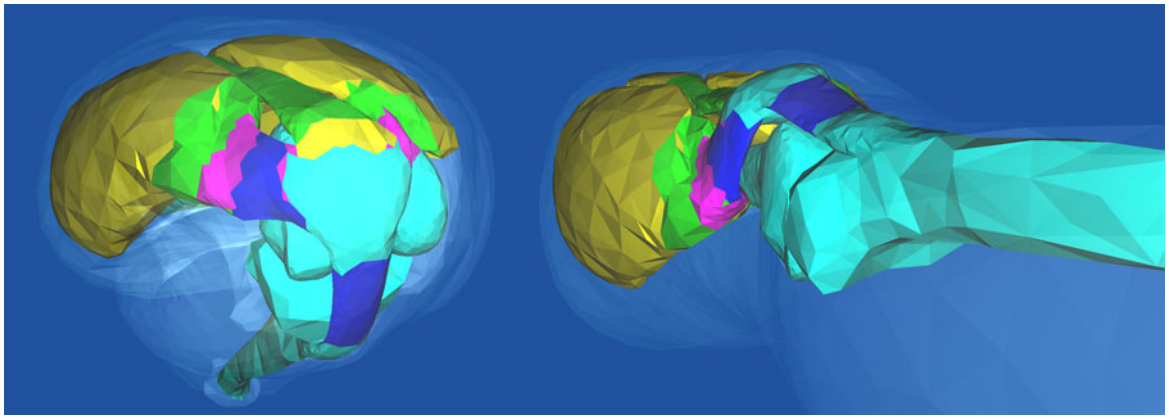

Figure 43: Bent knee (left) and stretched (initial) position (right). The patella automatically slides over the femur as a result of the simulation.

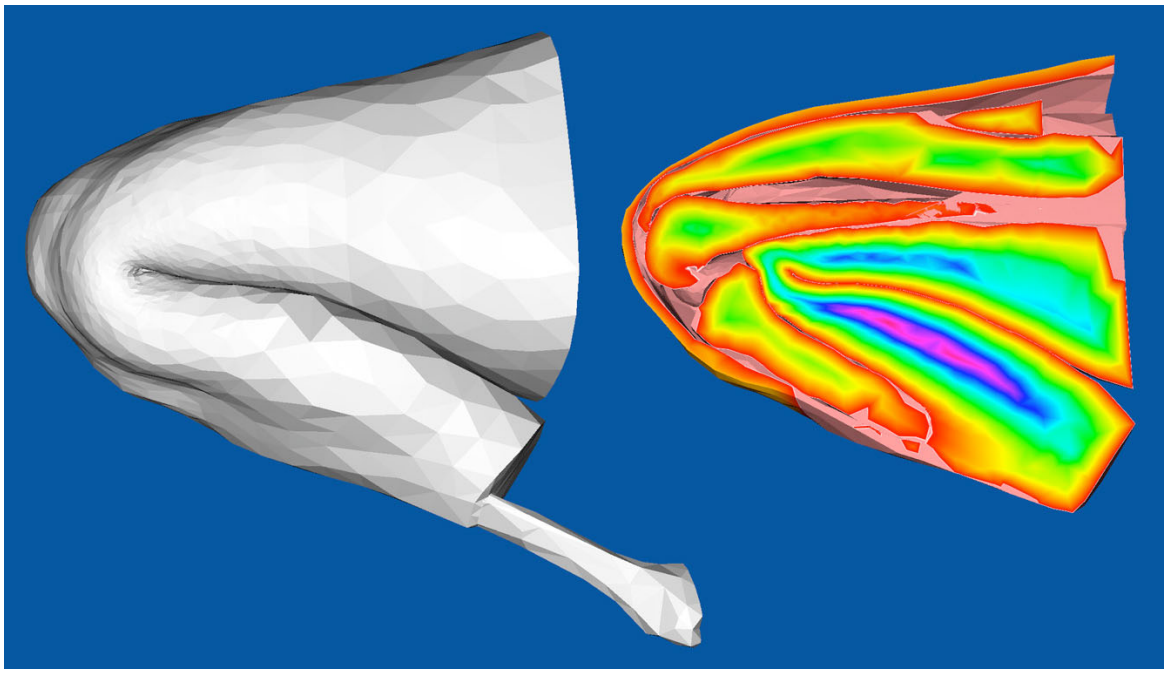

Figure 44: Skin surface of highly flexed knee (left), cut-away view of the same flexed knee (right). Only parts of the tibia and femur are visible in the cut-away, since they are partly in front of or behind the cutting plane. Note the natural-looking sliding contact between skin areas, skin and bones/muscles, patella and femur. The colors encode the material depth value; red is the shallowest and cyan is the deepest.

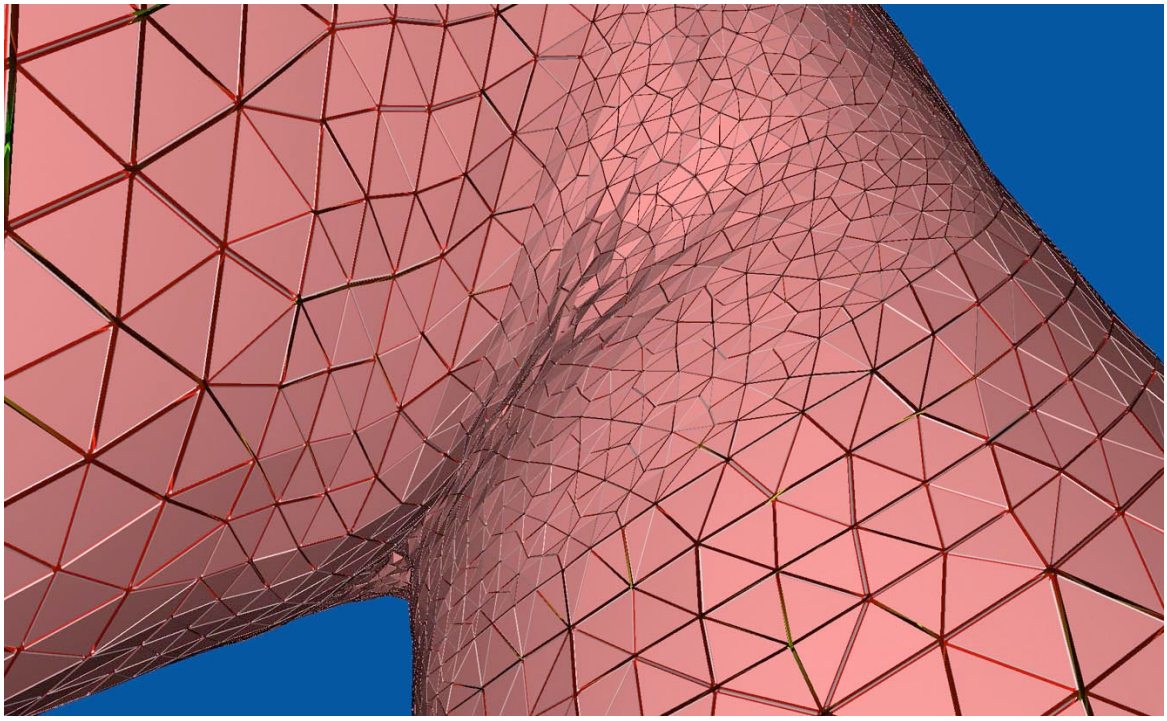

Figure 45: Close-up of knee, illustrating pattern of skin folding.

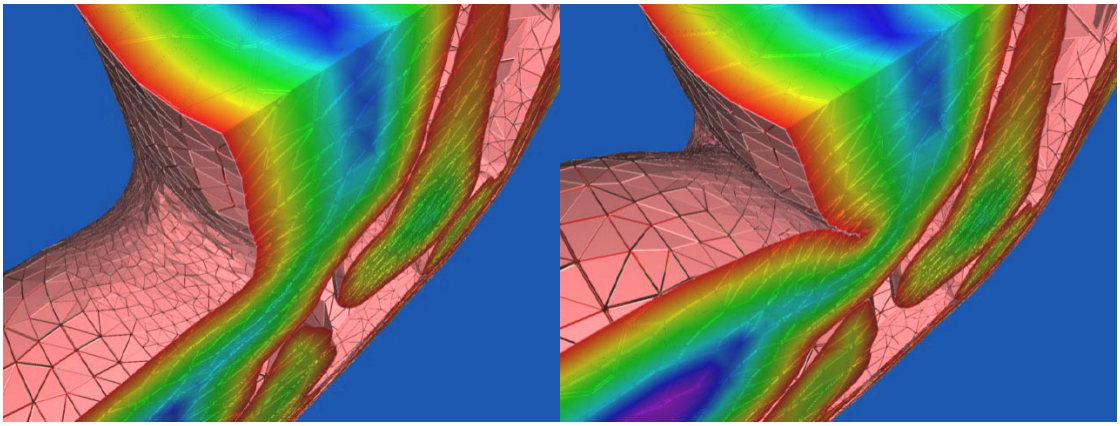

Figure 46: The complex self-contact of folding skin was handled without visible penetration.

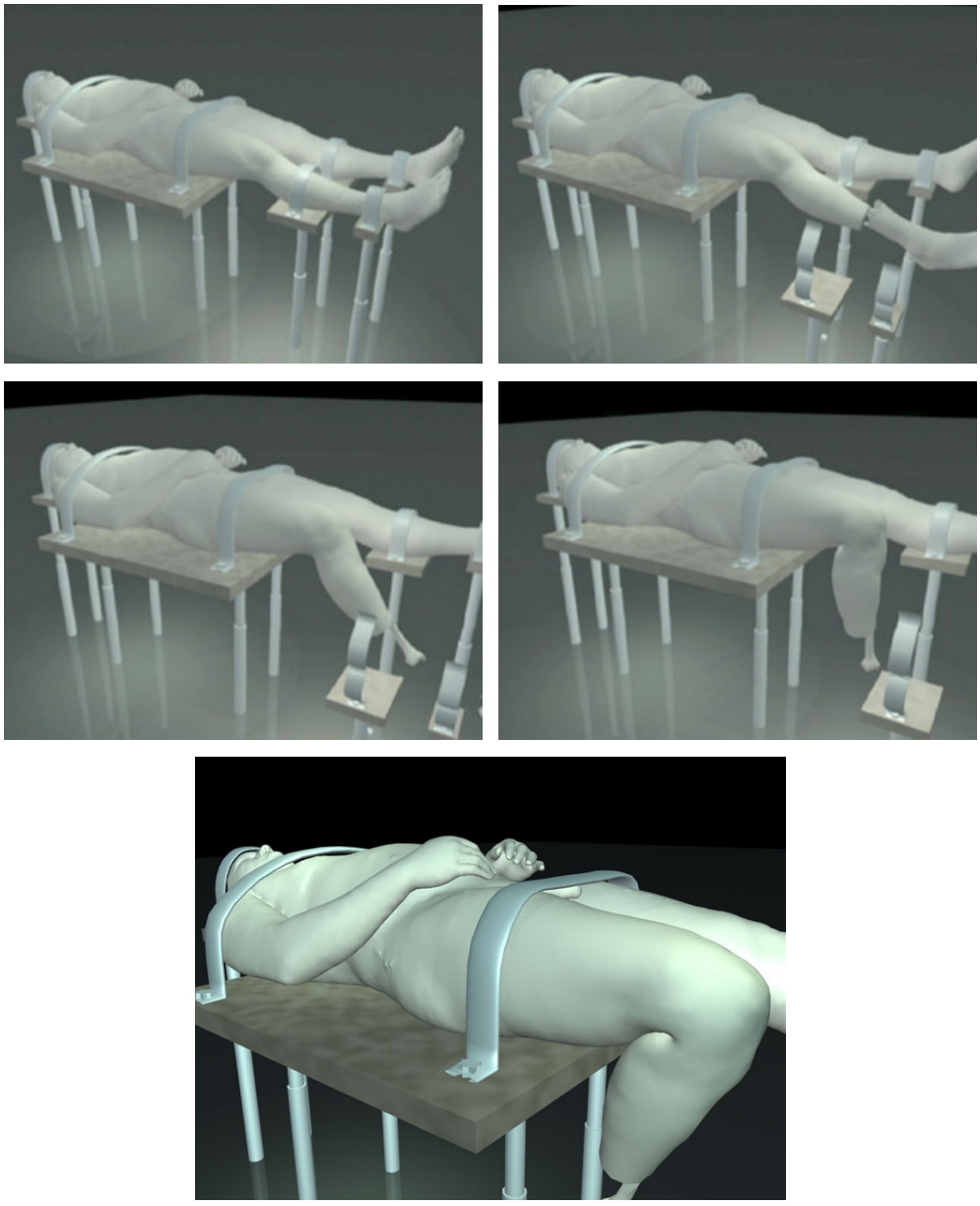

Figure 47: Visible human data with bent knee

7.5.3 CPU Time Statistics

The complete simulation took 376 minutes on a single 300MHz R12000 CPU of an SGI Onyx system. **Figure 48** shows ratios of various processes in the algorithm. Most of the time (63%) was consumed by the force and stiffness matrix computations. 22% of the time was spent on collision detection, out of which the bounding volume tree construction took less than 1%. The rest of the time, 15%, was spent on the linear system solution.

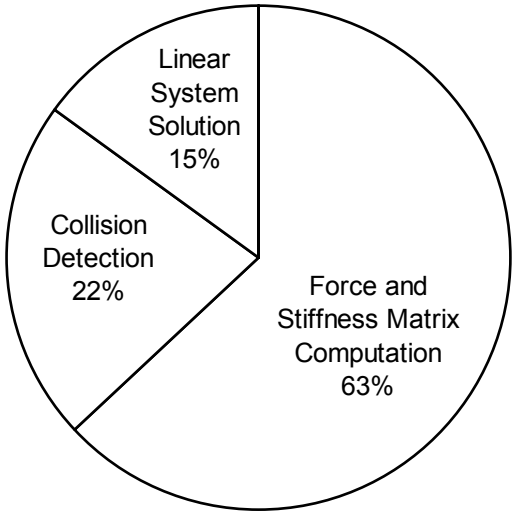

Figure 48: CPU time ratios of three parts of the algorithm

8 OTHER EXAMPLES

This chapter shows two other examples. One is an earlier attempt to use my old algorithm with point collocation to create a simple animation. The other example shows my algorithm's ability to analyze complex dynamic problems.

8.1 Early Example

The point collocation based algorithm used for comparison in section 6.5 was implemented in 1999. The algorithm was applied to create simple animations.

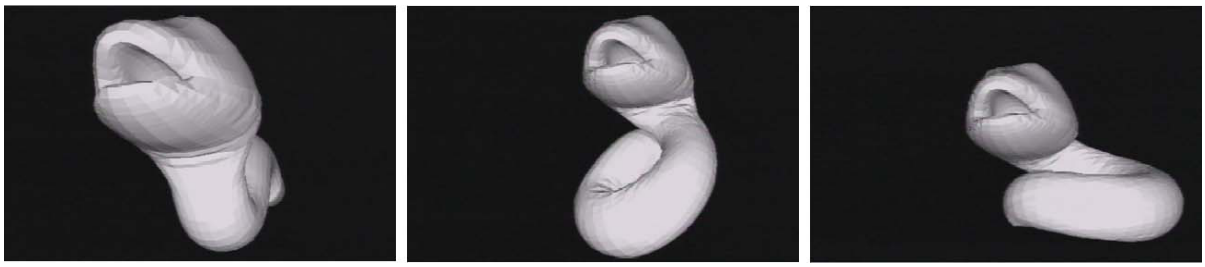

Figure 49: Coiling snake.

Figure 49 shows an animation sequence of a coiling snake. The nodal points located on the tail of the snake were fixed in space. The nodal points on the head were moved towards the tail. The simulation was done as a static analysis. This algorithm generated the coiling movements of the snake while avoiding self-penetration of the body.

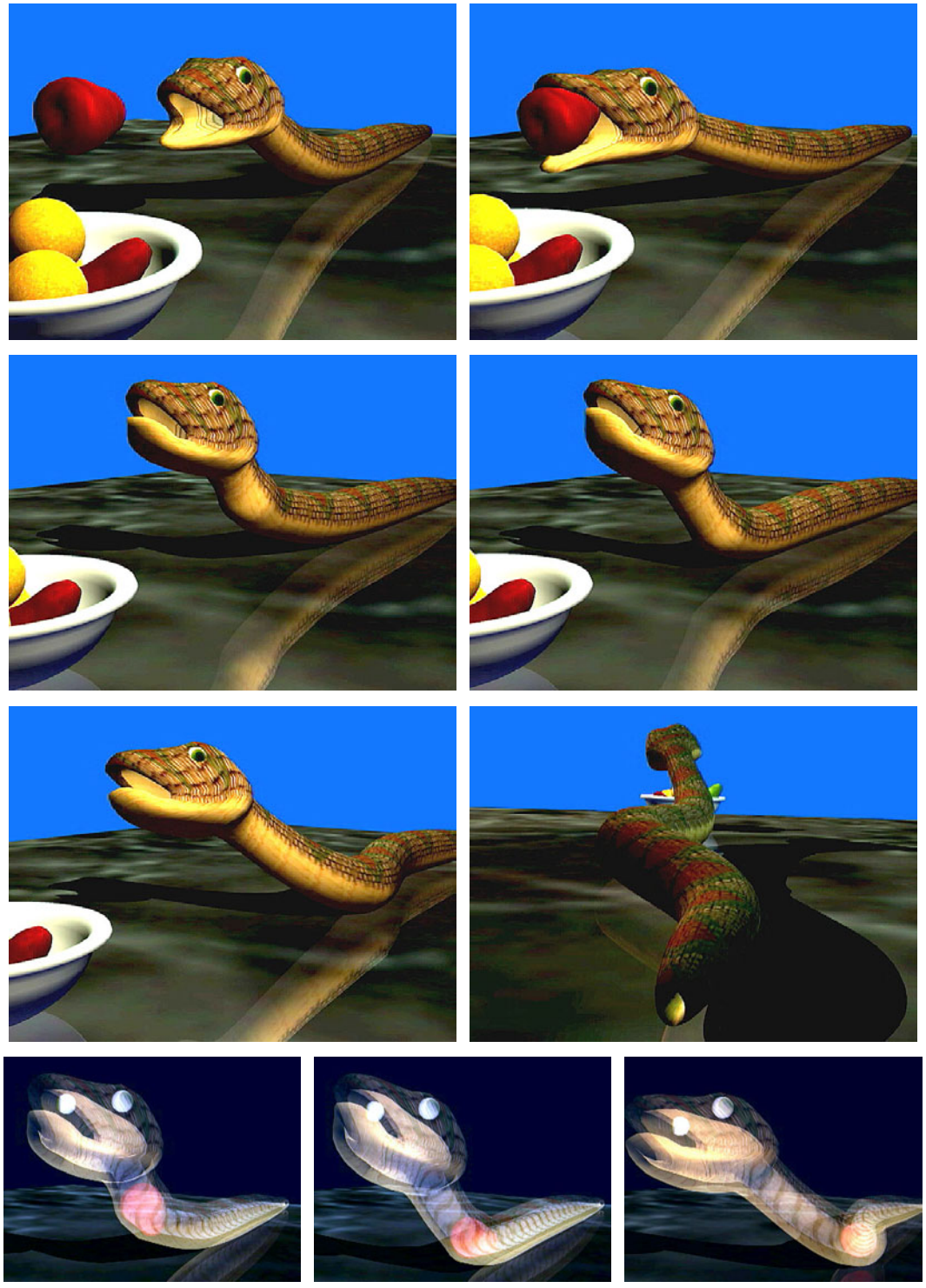

Figure 50: Snake and apple: A snake swallows an apple. The upper 3 rows show the sequence in opaque rendering. The images in the lowest row use transparent rendering to show the internal movement of the apple.

Figure 50 shows another animation sequence. The snake in the figure was modeled as a long tube. A few nodal points on the snake's head were fixed in space. The nodal points at which the snake's belly touched the ground were constrained so that they moved only on the ground plane. The apple was guided into the snake's mouth and traveled along a prescribed zigzag path toward the snake's tail. My algorithm simulated the snake's deformation caused by the apple moving inside of it. This simulation took approximately 8 hours on a single 300MHz R12000 CPU of an SGI Onyx system.

8.2 Dynamic Analysis

This section describes a simulation performed by the implicit dynamic FEM code equipped with my area integration contact algorithm. The upper-left image of **Figure 51** shows the initial state of the simulation. There is a 7×5 array of rods mounted on a 45° slanted foundation. The foundation rises from a flat ground plane. There is a curved bar running just below the rod array. The rods are deformable and made of a Mooney-Rivlin material. The arrangement of rods is slightly skewed rather than completely rectangular to provide a more interesting deformation. The foundation, ground, and curved bar are rigid.

At the beginning of the simulation, all the objects are stationary. Gravity then pulls the rods toward the ground causing them to gradually gain speed. Then the lower rods collide with the curved bar, bounce back, and collide again with the upper rods. The collisions are propagated upwards, generating simultaneous contact-impact patterns. A simple linear viscosity is applied to the deformation. Therefore, the kinetic energy of the rods gradually dissipates until they become stationary. The images of **Figure 51** show the sequence of deformation to the final stationary state. This simulation took approximately 3 hours on a single 300MHz R12000 CPU of an SGI Onyx system.

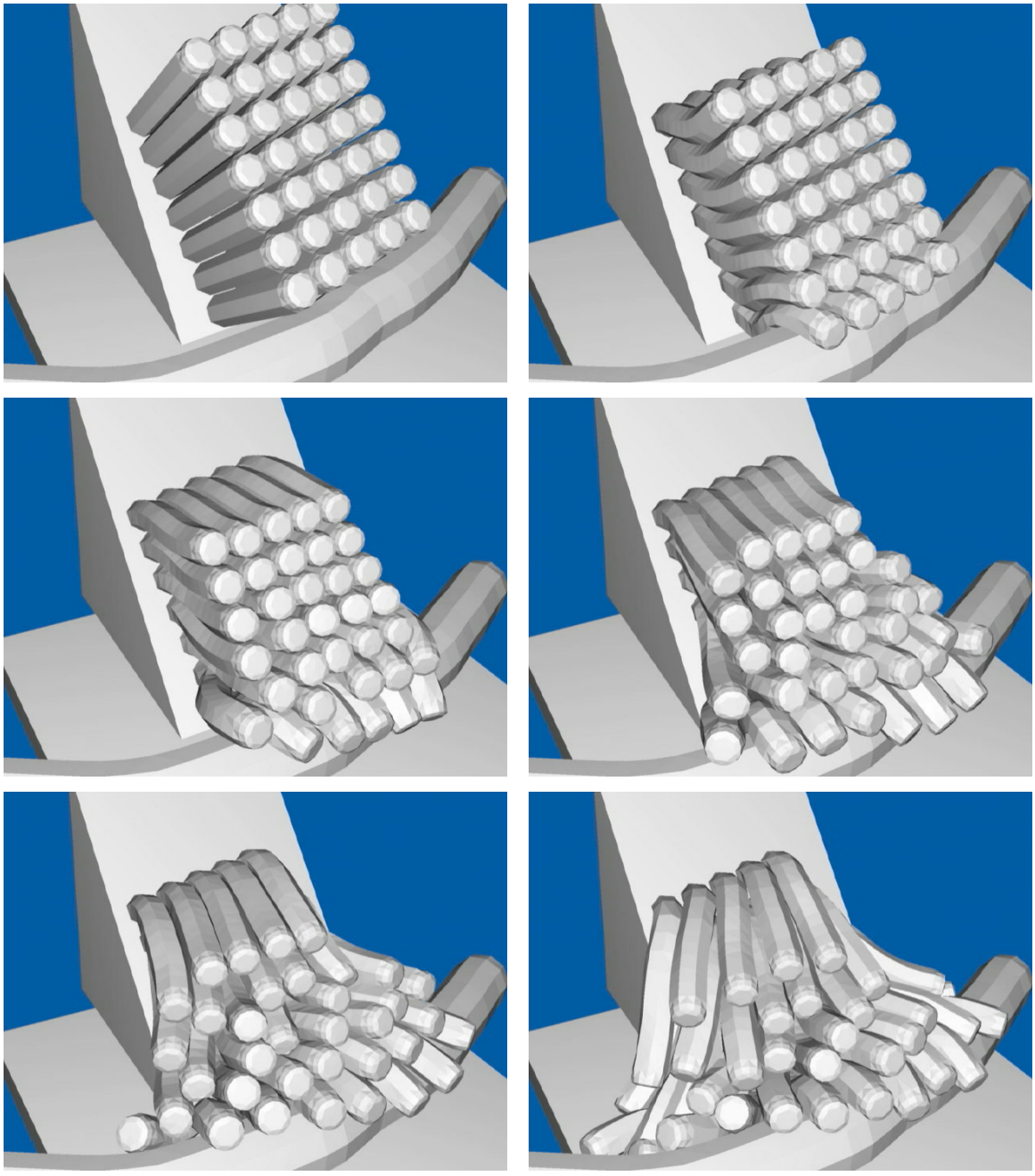

Figure 51: A stack of rods deformed by gravity

9 SUMMARY AND CONCLUSIONS

9.1 Contributions

9.1.1 New Algorithm

I developed a new contact force computation algorithm. The novelty of this algorithm is summarized as follows:

1) **Unambiguous contact force computation:** Contact forces are derived from material depth, the distance from the boundary in the reference (undeformed) configuration. Given a penetration free initial state, the definition of material depths is always unambiguous.
2) **Area-integrated smooth contact forces:** The field of material depth is approximated by a linear interpolation of depth values at finite element nodal points. This simplification enables efficient analytical integration of contact penalty forces over the contact area and results in contact forces that are continuous functions of deformation.

The continuity achieved by this algorithm reduces the oscillation and divergence problems often present in previous techniques.

9.1.2 Algorithm Analysis

I analyzed my algorithm using both mathematical and numerical methodologies. Three mathematical proofs were given:

1) **Continuity of contact forces:** The continuity of contact penalty forces was proven. The proof is subject to two major geometric assumptions. The first assumption is that all tetrahedral elements have positive volumes. Since zero or negative volumes (element reversal) are precluded by the iterative solver, this situation should not occur. The second assumption is that none of tetrahedral elements' sides are exactly coplanar with a triangular boundary element. Such cases are very rare and also can be avoided by perturbing nodal positions (see section 6.3).
2) **Existence of a solution:** Given the continuity result stated above, the existence of equilibrium point was proven via the existence of a minimum energy point (see section 6.2).
3) **Convergence of penalty method to exact constraint method:** My algorithm uses the penalty method to impose impenetrability constraints. A mathematical proof is given to show that, as the penalty factor approaches infinity, the solution of the penalty problem converges to a solution of the hard-constraint (i.e., constraints exactly satisfied) problem (see section 6.4).

Because of the discontinuity problem, conventional methods do not guarantee existence of a solution, reducing the chance of convergence. My algorithm, on the other hand, seeks a solution that is guaranteed to exist. Furthermore, the solution of my algorithm approaches a zero-penetration solution as the penalty factor grows.

My algorithm (and probably any other algorithm) does not guarantee the convergence of residual errors. Numerical analysis is more suitable to analyze the error convergence. I compared my algorithm with a traditional point-collocation-based method (section 6.5). Two methods were applied to an identical test case. The residual errors of my algorithm converged to low levels quickly while the errors of the conventional method hovered at very high levels. Furthermore, the stable convergence of my algorithm was demonstrated for high penalty factors (see section 6.5).

9.1.3 Anatomical Simulation

I demonstrated the performance of my method by simulating very large deformations on part of a human anatomical model. To my knowledge, this is the first demonstrated simulation of

large-scale motion of a complex model derived from the widely used Visible Human dataset and encompassing multiple tissue types including bone, muscle, tendons, and skin (see chapter 7).

9.2 Lessons Learned

In the course of this research, I had to deal with problems that I did not anticipate at the beginning. Here I explain those unexpected lessons I learned.

9.2.1 Solution of Nonlinear Systems

Nonlinear finite element methods involve the solution of large-scale nonlinear systems of equations. The treatment of contact problems further increases the nonlinearity of the systems. The development of a reliably converging solver was much harder task than expected. The first program I developed used the adaptive arc-length continuation algorithm, which was highly recommended in various textbooks, for example, [Le Tallec 1994]. However, the algorithm turned out to be unstable when solving very large deformations.

I finally succeeded in developing a stable solver that combines various numerical techniques such as Newton iteration, adaptive incremental loading, two-point predictor, line search (or variable damping factor), and quasi-viscosity. Yet, to handle very non-linear materials such as Veronda materials (see section 3.6.6) with large β values, the simulation must be carried out as a full dynamic simulation to avoid bifurcation and indefinite stiffness matrix problems.

9.2.2 Modeling from a 3D Scan

As shown in chapter 7, I created a polygonal solid model and a finite element mesh of the knee anatomy from the visible human data. This task turned out to be very time consuming. The difficulties are summarized as follows:

1) There are discrepancies between the visible human data and anatomy books.
2) Meshing programs demand high-quality triangle mesh solids.

3) Mechanical connectivities between organs are not always clear.

First of all, it was difficult to find one-to-one correspondence between visible human data and each part named in anatomical textbooks. Some tendons and ligaments are difficult to find due to their unclear borders in the color photographs. There is not enough information in the visible human data to make an anatomical model of the knee. Therefore, I had to create some parts of the model manually.

I used Maya's polygonal editor [Anderson 1999] for the modeling task. But this animation tool is not very convenient for creating geometry for simulations. Rendering and finite element analysis have different requirements for geometry. For example, CSG and polygon-merge operations often created degenerate mesh and small triangles which were not visible in rendered images. Meshing programs cannot deal with non-2-manifold objects or zero area triangles. In addition, the triangle mesh of each solid must be properly resampled to homogenize the sizes of triangles before it was passed to the tetrahedral meshing stage, for which I had to write my own program.

Some borders of two muscles are unclear even in the eyes of experts. It is also not easy to decide which borders are sliding interfaces or firmly attached ones. In such cases, I had to rely on educated estimation by trained anatomists.

The modeling task was not easy. It is interesting to project this experience to an actual animation production. As discussed in the introduction chapter, *automation* is one of the purposes of using "physically based" techniques for computer animation. The use of physics demands physically-sound models, which, ironically, may prove to require time-consuming manual modeling efforts.

9.3 Limitations

This section discusses some of the limitations of my algorithm, specifically problems with thin objects and high-pressure contact.

9.3.1 Thin object

In dynamic analysis, my algorithm detects contact based on the geometry at each time step. If an object is thinner than the distance it travels in one step, two objects may pass through each other. In other words, my algorithm cannot detect contacts that occur between two time steps.

9.3.2 High-Pressure Contact

As the pressure between two objects increases, more penetration is allowed. The amount of penetration may exceed a certain application-dependent limit. If this limit is exceeded (as mentioned in section 6.4), the penalty factor should be manually increased. In extreme cases, the penalty factor may reach the level where numerical stability is lost. In such cases, however, the forces between internal elements are also very large. Those forces cancel each other at nodal points resulting in a loss of floating point precision. Thus the stability problem may emerge whether the penalty method is used or not.

9.4 Discussion

9.4.1 Achievement with Respect to Initial Motivation

As mentioned earlier, the initial motivation was to simulate the deformation of breasts in ultrasound needle biopsy operations. Although the objective of the dissertation was later focused on the development of a new contact algorithm, it is interesting to speculate how the current implementation of my finite element simulation system would perform in terms of predicting breast deformations.

The breast is deformed by contacts with external objects such as the needle, the ultrasound probe, and the physicians' hands. In addition, the movement of the foundation (rib cage) and the change of the gravity direction with respect to the patient due to movement contribute to breast deformation. The skin surface may make self-contact as well. To limit the argument to the simulation aspect, the issue of tracking (the measurement of the movements of the external objects and the patient) is ignored here. External objects can be treated as rigid, so the contact problem can be simplified accordingly.

The deformations of breasts are quite large. Therefore, classic linear elasticity would be unrealistic. The nonlinear capability of my system is suitable for this application. On the other hand, exact constitutive laws for biological tissues are not fully understood. Breast tissue is not an exception in this sense. Of course, simpler models might suffice for this particular application.

Augmented reality (AR) image-guided breast biopsy also requires real-time simulation. Therefore, the speed of simulation is an important consideration. Significant simplification would be necessary to achieve interactive response. The next section discusses the possibilities of such a simplification.

9.4.2 Trading Accuracy for Speed

The most decisive factor for computation time is the complexity of the problem. Fast simulations can be achieved by simplifying (hence reducing the fidelity of) the simulation models.

The time complexity of FEM computation is roughly linear in the number of elements. The simplest way to reduce the computation time is to use fewer elements. The choice of materials is another chance for simplification. In this work, I used relatively complex constitutive laws such as Mooney-Rivlin and Veronda models. If I use a simpler model, namely the Neo-Hookean material, the stiffness matrix computation is greatly simplified reducing the cost by nearly an order of magnitude [Bonet 1997]. The movements of objects can be slowed down by increasing the inertial forces. Because of the sluggishness of the objects, larger time steps can be safely chosen [Richtmyer 1967]. Since such modification changes the objects' behavior, the simulation is no longer accurate. Nevertheless, such inaccuracy is justified in many interactive applications.

9.4.3 The Myth of Curved and Smooth Surfaces

In the computer graphics community, it is often believed that smoothly-curved surfaces are more realistic. On computer screens, discontinuous borders (namely triangle edges) can be easily detected by human visual systems and give people the impression that what they are looking at is not real. The continuous borders of high-order polynomial or rational surfaces

are often preferred. But does the smoothness of the surface help the physics simulation as well? Many mechanical parts have mathematically-defined smoothly curved surfaces. Some of the surfaces of internal organs (like the liver) certainly look smooth, so one might argue that smoothness is closely related to physical realism.

The question about smoothness includes two nearly independent issues:

- Should the initial geometry be smooth?
- Should the deformation be smooth?

The first question is about the smoothness of object boundaries in the reference configuration. This question is about the domain *where the PDE is integrated*. The domain of the PDE and the shape of the finite element mesh do not always have to match. The PDE can be integrated within curved boundaries by adjusting the weights and positions of quadrature points (Quadrature is numerical integration by weighted average of sampled values). However, such decoupling may make the meshing process more difficult.

The second question is about the smoothness of the shape functions (approximation bases). The smoothness requirement of shape functions are directly related to the order of the PDE. The linear bases for tetrahedral elements are sufficient for the PDEs used in this dissertation. Higher polynomial-order elements are often used in finite element analysis. However, the use of high-order polynomials is based purely on accuracy considerations. A higher-order polynomial implies more degrees of freedom, which reduces the discretization error. In fact, in usual finite element methods, the increased polynomial order is not used to achieve continuity across element boundaries. (Of course, the continuity must be increased as the order of the PDE increases. For example, shell elements have a higher continuity requirement. But 3D bodies require only second order PDEs.) Additional continuity constraints would reduce the degrees of freedom and increase the discretization error. Thus in this particular context, *the continuity decreases the realism of the simulation*, which might sound counterintuitive for graphics researchers.

By using shape functions with a broader basis (such as NURBS), the continuity of the deformation function can be achieved. However, a broader basis means more cross-talk between nodal points resulting in denser stiffness matrices. Consider a trilinear hexahedral

element. The size of the stiffness matrix is $2^3 \times 2^3$. If a tricubic B-Spline is used, the size jumps to $4^3 \times 4^3$, which is computationally impractical.

As mentioned earlier, smooth surface normals are sometimes useful for stabilizing contact forces [Puso 2001]. However, the area-integration smoothing technique developed in this dissertation is more preferable when self-contacts and high curvature surfaces are treated. To simulate contacts between two bodies with smooth boundaries, normal smoothing techniques may be more efficient. Ultimately, the differential equations decide if the surface should be smooth. If a wrinkling surface emerges as a solution of the equations, one would know that the surface is not supposed to be smooth.

The next chapter discusses the possible improvements to and enhancements of my system.

10 FUTURE WORK

This chapter discusses future research possibilities following this dissertation work. The first section discusses possible enhancements to and improvements of my contact handling algorithm. The second section looks at biomechanical simulations in terms of accuracy.

10.1 Improvement of the Contact Algorithm

This section explains enhancement of my contact algorithm in three directions: 1) increasing robustness and accuracy, 2) improving performance, and 3) simulating more complex phenomena. The first direction would include a robust treatment of thin objects and more stable contact force computation. The second direction includes faster collision detection and parallelization. The third direction is the simulation of frictional contact.

10.1.1 Thin Objects

It is rare that flat objects are simulated as zero thickness plates [Heinstein 1993 May]. However, it is certainly an advantage for an algorithm if it can handle objects with smaller thickness efficiently. This can probably be achieved by truncating the length of time steps to the nearest contact time estimated by a method similar to the one proposed by Heistein et al. [Heinstein 1993]. Instead of checking snapshots at discrete points in time, their algorithm examines geometry between two successive steps, and the approximate contact time is obtained as the intersection of trajectories. After the time step is properly truncated, my algorithm can perform the rest of the contact handling.

10.1.2 Augmented Lagrangian Method

The penalty method used in this dissertation seems to perform very well. Yet, a more sophisticated method can be devised by representing pressure on contact surfaces (contact force magnitude) by Lagrange multipliers. Lagrange multipliers would make the enforcement of the penetration limit more robust. The pressure field would be approximated by a linear interpolation of nodal pressure values and the contact force would be defined as a product of the interpolated pressure and the gap function gradient. By combining penalty methods and Lagrangian multiplier techniques, an augmented Lagrangian scheme can be devised. In this scheme, the penalty term is preserved from the current algorithm. The difference between the interpolated pressure and the magnitude of the penalty force is integrated and used to update the nodal pressures. The pressure update is repeated until the penetration becomes below the desired tolerance.

10.1.3 Optimizing Collision Detection on the Finite Element Mesh

The topological information in finite element meshes can be utilized to eliminate unnecessary checks in collision detection. In the current implementation, if a triangle is one of the tetrahedron's sides, the algorithm skips the detailed tetrahedron-triangle intersection check. However, my algorithm does not check if a triangle shares an edge or a vertex with the tetrahedron. Therefore, some of the tests in the intersection checks are redundant. In addition, there is great amount of duplicated computation among intersection checks for different triangle-tetrahedron pairs. A more sophisticated pruning mechanism would reduce such unnecessary checks.

10.1.4 Parallelization

Most of the computation in my algorithm is local to each finite element. For example, force and stiffness matrix computation can be performed for a small collection of elements making it the easiest part of the algorithm to parallelize. The final assembly of forces and matrices can be done by a parallel summation algorithm.

It is also possible to parallelize the collision-detection technique employed in my algorithm. The traversal of the bounding volume tree and all subsequence checks can be concurrently done for each triangle element. The bottom-up update of the bounding volume tree can also be performed locally. The occasional tree rebuilding task is probably the hardest part for parallelization, but partial parallelization is probably sufficient because the tree building occurs infrequently. Brown et al. has shown that their parallelized collision detection method scales up to a few thousand processors in very large scale contact-impact simulations [Brown 2000].

10.1.5 Friction

My algorithm is limited to frictionless contact problems. The surface traction force is assumed to have no tangential components. This limitation is justified because the friction between lubricated organ surfaces is known to be small [Malcolm 1976, Moro-oka 1999]. This is not the case for friction between (non-lubricated) skin surfaces. My algorithm should be extended to frictional forces to increase the simulation fidelity. I expect that the extension to be relatively straightforward. Frictional forces are functions of normal forces and relative velocities on the contact surfaces. They can be integrated over the contact areas just as normal contact forces are integrated. Furthermore, continuous normal forces (which were realized by my algorithm) are helpful in performing more accurate simulations of frictional contacts [Puso 2001].

10.2 More Accurate Simulation

The contact algorithm discussed in this dissertation is merely a component of a larger biomechanical simulation system. The sole purpose of my finite element simulation program was to demonstrate the usefulness of my contact handling algorithm. Therefore, this dissertation did not discuss the numerical errors contained in the computed results. However, the accurate simulation of biological systems is the final goal of virtually every researcher who is interested in analyzing and predicting the deformations of human bodies or animal

anatomies. This section considers various sources of simulation error and discusses some of the opportunities for improving the accuracy of the simulations.

10.2.1 More Accurate Model

The first step in performing an accurate simulation is to build a high-fidelity mathematical model of the subject. The leg model used in chapter 7 can be improved in many respects: material, friction, residual stress, and fluid-solid interactions.

My leg model uses compressible elastic materials with simple linear viscosity. The volume-preserving property is approximated by a logarithm term that goes to infinity as the material's volume approaches zero. It is certainly more realistic than classic linear elastic models, which allow zero or negative volume. However, biological tissues contain 60-70% water which is nearly incompressible. Therefore, an accurate material model should exhibit incompressibility. To make things even more complicated, the fluids contained in tissues can migrate from one place to another. While the fluids are incompressible, the underlying protein networks are compressible. The use of *biphasic* constitutive laws is a possible way to simulate both incompressible and compressible *phases* [Mow 1980, Donzelli 1995].

In vivo measurement of material properties will continue to be a challenging problem. One research direction tries to solve the inverse problem: given the deformation and stress of an anatomical subject, one should be able to infer the material properties of the internal organs. If the anatomical structure includes sliding contacts, then it should also be possible to estimate frictional properties on the interfaces. Additional information from diffusion MRI would help find fiber orientations and anisotropic mechanical properties.

Tissues are usually subject to certain stresses even without external forces. If skin is cut, the wound opens up due to its surface tension. A longitudinal dissection of a blood vessel "unrolls" the vessel which also indicates the existence of *residual stress*. This must also be measured and incorporated into the simulation model.

Other mechanical effects that should be considered include atmospheric pressure, which squeezes the body through the skin, as well as fluid-solid interaction such as the one between blood and blood vessels. Chemical and biophysical phenomena also contribute to internal stresses. Muscle contraction is the most dramatic example for this. Such stresses should be

included as parts of the external body forces to simulate "active" aspects of biological tissues.

10.2.2 More Accurate Computation

All physical phenomena explained above are described in the form of partial differential equations (PDEs). Discretization (FEM in this dissertation) converts the PDEs into algebraic equations, and a nonlinear equation solver finds the final answer (the objects' deformations). By plugging the solution back into the original PDEs, one would find that the solution does *not* satisfy the PDEs; the PDEs would have non-zero values (residual errors measured as volumetric forces). The residual error does not go away even if the nonlinear solver finds an exact solution for the algebraic equations since the algebraic equations are fundamentally different from the original PDE. The existence of a residual error (force) implies that the solution contains an error (measured as position), too. This positional error in the solution is introduced by discretization; hence it is called the *discretization error*.

The discretization error in the FEM is affected by the shape and size of each element. The polynomial orders of shape functions also change the error. It is usually hard to estimate the discretization error before computation (a priori error estimation). Most methods estimate the error from residual error (a posteriori error estimation) [Babuska 1984]. The distribution of the estimated error is used to identify the location where the finite element mesh should be adjusted. There are at least three aspects that can be adjusted. If the element sizes are adjusted, the method is called *h-adaptive* ("h" is usually the choice of symbol for the step size i.e., element size. With an h-adaptive method, the FEM analysis can be started with a coarse mesh, and, according to the a posteriori error distribution, the mesh can be locally refined until a sufficient accuracy is achieved.) If polynomial order is adjusted, it is called *p-adaptive*. And finally, if the position of an element is moved (relocated), it is called *r-adaptive*. In my knee bending simulation, the mesh around the knee joint was highly refined because it was obvious that a complex deformation would emerge in the region. Adaptive methods would achieve something similar but in an automatic and optimum way. An interesting aspect of the adaptive method is that it potentially relaxes the convergence criteria for the equation solver. Given a finite element mesh, the error estimator can tell how far the

residual error can be reduced. Once the nonlinear equation solver reduces the error to the same level, the iteration can be terminated, resulting in more efficient computation.

APPENDIX A. VECTOR, MATRIX AND TENSOR NOTATION

A.1 Nomenclature

Non-boldface symbols are used to name scalar objects (e.g. a, C, λ, Ψ), whereas boldface symbols are used to name non-scalar objects such as vectors, matrices, and tensors (e.g. $\boldsymbol{v}, \boldsymbol{V}, \boldsymbol{M}, \boldsymbol{\mathcal{C}}$).

In this dissertation, only a Cartesian space is used and a single basis defines a coordinate system. All tensors in this dissertation are linear mappings. Therefore, the components of those objects[4] are uniquely determined. It is sometimes convenient to specify a component of non-scalar object by using an indicial form. In such cases, non-boldface symbols with index subscripts denote the components. For example, given a vector $\boldsymbol{V}$ and a matrix (or a second order tensor) $\boldsymbol{M}$, their components are denoted as V_i and M_{ij} respectively. A vector is considered as a matrix with a single column:

$$\boldsymbol{V} = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \end{bmatrix} \tag{11.1}$$

For a matrix, the first (left) subscript specifies the row number, and the second the column number

$$\boldsymbol{M} = \begin{bmatrix} M_{11} & M_{12} & \\ M_{21} & M_{22} & \\ & & \ddots \end{bmatrix} \tag{11.2}$$

[4] Numbers, variables, vectors, matrices, and tensors are all mathematical *objects*.

A non-scalar quantity is sometimes given as an expression rather than a symbol. In such cases, since direct indexing is not possible, components must be extracted by subscripting outside a parenthesis. For example, given two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, the components of $\boldsymbol{A}+\boldsymbol{B}^{-1}$ are denoted as $\left(\boldsymbol{A}+\boldsymbol{B}^{-1}\right)_{ij}$.

A.2 Operators for Vectors and Matrices

A.2.1 Ordinary multiplications

Succession of two symbols implies ordinary multiplication. For example, given a scalar a and, a vector $\boldsymbol{v}$, and two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$,

$$a\boldsymbol{v}, a\boldsymbol{A}, \boldsymbol{A}\boldsymbol{v}, \boldsymbol{A}\boldsymbol{B}$$

are among possible ordinary multiplications. Componentwise, they are defined as

$$\left(a\boldsymbol{v}\right)_i = av_i \tag{11.3}$$

$$\left(a\boldsymbol{A}\right)_{ij} = aA_{ij} \tag{11.4}$$

$$\left(\boldsymbol{A}\boldsymbol{v}\right)_i = \sum_j A_{ij}v_j \tag{11.5}$$

$$\left(\boldsymbol{A}\boldsymbol{B}\right)_{ij} = \sum_k A_{ik}B_{kj} \tag{11.6}$$

A second-order tensor $\boldsymbol{S}$ is a matrix that maps a vector $\boldsymbol{u}$ to a vector $\boldsymbol{v}$ as

$$\boldsymbol{v} = \boldsymbol{S}\boldsymbol{u} \tag{11.7}$$

A.2.2 Scalar products

A *scalar product* '$\cdot$' between two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is a scalar value:

$$\boldsymbol{u}\cdot\boldsymbol{v} = \sum_i u_i v_i \tag{11.8}$$

The scalar product does exist between two matrices. It is discussed in the *double contraction* section below.

A.2.3 Cross products

The only *cross product* '$\times$' appears in this dissertation is defined between three vectors. Given two three-vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, the cross product is defined as

$$\boldsymbol{u} \times \boldsymbol{v} = \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix} \tag{11.9}$$

A.2.4 Tensor products

Given two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, the *tensor product* is defined as

$$\boldsymbol{u} \otimes \boldsymbol{v} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & \\ u_2 v_1 & u_2 v_2 & \\ & & \ddots \end{bmatrix} \tag{11.10}$$

Or more precisely

$$\left(\boldsymbol{u} \otimes \boldsymbol{v}\right)_{ij} = u_i v_j \tag{11.11}$$

A.2.5 Double contractions

Given two matrices (or second-order tensors) $\boldsymbol{A}$ and $\boldsymbol{B}$, their *double contraction* is

$$\boldsymbol{A} : \boldsymbol{B} = \sum_{ij} A_{ij} B_{ij} \tag{11.12}$$

A.3 Higher Order Tensors

As seen in (11.7), the second order tensor was a linear mapping from a vector to a vector. Similarly a third order tensor $\boldsymbol{\mathcal{A}}$ is defined as a linear mapping from a vector $\boldsymbol{u}$ to a second order tensor $\boldsymbol{S}$

$$\boldsymbol{S} = \boldsymbol{\mathcal{A}}\,\boldsymbol{u} \tag{11.13}$$

Component wise, this is defined as

$$S_{ij} = \sum_{k} \mathcal{A}_{ijk} u_k \tag{11.14}$$

The double contraction of $\boldsymbol{\mathcal{A}}$ and a second order tensor $\boldsymbol{T}$ is defined as

$$\left(\boldsymbol{\mathcal{A}} : \boldsymbol{T}\right)_i = \sum_{jk} \mathcal{A}_{ijk} : T_{jk} \tag{11.15}$$

A fourth order tensor $\boldsymbol{\mathcal{C}}$ is defined as a linear mapping from a vector to a third order tensor $\boldsymbol{\mathcal{A}}$

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{C}}\boldsymbol{u} \tag{11.16}$$

or component wise

$$\mathcal{A}_{ijk} = \sum_{l} \mathcal{C}_{ijkl}\, u_l \tag{11.17}$$

The double contraction of $\boldsymbol{\mathcal{C}}$ and a second order tensor $\boldsymbol{T}$ is defined as

$$\left(\boldsymbol{\mathcal{C}} : \boldsymbol{T}\right)_{ij} = \sum_{kl} \mathcal{A}_{ijkl} : T_{kl} \tag{11.18}$$

BIBLIOGRAPHY

ANDERSON, P., et al (1999), Using Maya. Alias|Wavefront.

BABUSKA, I., CHANDRA, J., and FLAHERTY, J. E. (1984), Adaptive computational methods for partial differential equations. SIAM, Philadelphia.

BARAFF, D., WITKIN, A. (1992), Dynamic simulation of non-penetrating flexible bodies. Proceedings of SIGGRAPH `92, Computer Graphics, 26, 2, pp. 303-308.

BARAFF D., WITKIN A. (1998), Large steps in cloth simulation. Proceedings of SIGGRAPH `98, Computer Graphics Proceedings, Annual Conference Series, pp. 43-54.

BARTH, T.J., SETHIAN, J.A. (1998), Numerical schemes for the Hamilton-Jacobi and level set equations on triangulated domains. J. Computational Physics 145(1), pp. 1-40.

BECHMANN, D. (1994), Space deformation models survey. Computer & Graphics, 18(4), pp. 571-586.

BELYTSCHKO, T., NEAL, M.O. (1991), Contact Impact by the pinball algorithm with penalty and Lagrangian Methods. International Journal for Numerical Methods in Engineering, vol. 31, 547-572. Jon Wiley & Sons, Ltd.

BELYTSCHKO, T., YEH I.-S. (1992), The splitting pinball method for general contact. In R.Glowinksi (ed.), Computing Methods in Applied Science and Engineering. Nova Science Publishers, New York, NY.

BENSON, D.J., HALLQUIST, J.O. (1990) A single surface contact algorithm for the post-buckling analysis of shell structures. Computer Methods in Applied Mechanics and Engineering 78:141-163, NORTH-HOLLAND.

BONET, J., WOOD, R.D. (1997), Nonlinear continuum mechanics for finite element analysis. Cambridge University Press.

BREGLER, C. (chair) (2000), Hollow men and invisible people - Layers of a digital actor. SIGGRAPH 2000 Technical Sketch.

BRIDSON, R., FEDKIW, R., ANDERSON, J. (2002), Robust treatment of collisions, contact and friction for cloth animation. Proceedings of SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series.

BROWN, K., ATTAWAY, S., PLIMPTON, S., HENDRICKSON, B. (2000), Parallel strategies for crash and impact simulations. Computer methods in applied mechanics and engineering 184, pp. 375-390.

CARSTENSEN, C., SCHERF, O., WRIGGERS, P. (1999), Adaptive finite elements for elastic bodies in contact. SIAM J. Scientific Computing 20(5), pp. 1605-1626.

CIARLET., P.G. (1987), Mathematical Elasticity. North-Holland.

CLOUGH, R.W. (1960), The finite element in plane stress analysis. Proc. 2nd ASCE Conf. on Electronic Computation, Pittsburgh, Pa.

DEROSE, T., KASS, M., TRUONG, T. (1998), Subdivision surfaces in character animation. Proceedings of SIGGRAPH 98, Computer Graphics Proceedings, Annual Conference Series, pp. 85-94.

DONZELLI, P.S. (1995), A mixed-penalty contact finite element formulation for biphasic soft tissues, PhD Thesis, Dept. of Mech. Eng., Aeronautical Eng. and Mechanics, RPI, Troy, NY.

DONZELLI, P.S., SPILKER, R.L. (1998), A Contact Finite Element Formulation for Biological Soft Hydrated Tissues. Computer Methods in Applied Mechanics and Engineering, 153, pp. 63-79.

FUNG, Y.C. (1993), Biomechanics. Springer-Verlag.

GALERKIN, B.G. (1915), Series solution of some problems of elastic equilibrium of rods and plates, Vestnik. Inzh. Tech. 19, pp. 897-908.

GEORGE, P.L. (1996), Automatic Mesh generation and finite element computation. In CIARLET, P.G., LIONS, J. L. (eds.), Handbook of numerical analysis. Vol IV, North-Holland.

GOURRET, J.-P., THALMANN, N.M., THALMANN, D. (1989), Simulation of object and human skin deformations in a grasping task. Proc. of SIGGRAPH 89, Computer graphics, 23, 4, pp. 21-30.

HEINSTEIN, M.W., ATTAWAY, S.W., SWEGLE, J.W., MELLO, F.J. (May 1993), A general-purpose contact detection algorithm for nonlinear structural analysis codes. SANDIA REPORT, SAND92—2141 • UC—705. SANDIA NATIONAL LABORATORIES TECHNICAL LIBRARY, Albuquerque, New Mexico.

HEINSTEIN, M.W., ATTAWAY, S.W., SWEGLE, J.W., MELLO, F.J. (July 1993), A general contact detection algorithm for finite element analysis. In Aliabadi, M. H., Brebbia, C. A. (eds.), Contact Mechanics. Computational Mechanics Publications, Southampton, UK.

HIROKAWA, S., TSURUNO, R. (2000), Three-dimensional deformation and stress distribution in an analytical / computational model of the anterior cruciate ligament. Journal of Biomechanics 33, 1069-1077.

HIROTA, G., FISHER, S., STATE, A., FUCHS, H., LEE, C. (2001), Simulation of Deforming Elastic Solids in Contact. SIGGRAPH Technical Sketch. In SIGGRAPH 2001 Conference Abstracts and Applications, p. 259.

KIKUCHI, N., ODEN, J.T. (1988), Contact Problems in Elasticity: A study of variational inequalities and finite element methods. SIAM Studies in Applied and Numerical Mathematics.

KLISCH, S.M., LOTZ, J.C. (1999), Application of a fiber-reinforced continuum theory to multiple deformations of the annulus fibrous. Journal of Biomechanics 32 1027-1036.

KOCH, R. M., GROSS, M. H., CARLS, F. R. (1996), von BÜREN, D.F., FANKHAUSER, G., PARISH, Y.I.H., Simulating facial surgery using finite element methods. proceedings of SIGGRAPH '96, Computer Graphics Proceedings, Annual Conference Series, pp. 421-428.

LE TALLEC, P. (1994), Numerical methods for solids. In CIARLET, P.G., LIONS, J. L. (eds.), Handbook of numerical analysis. North-Holland.

LEWIS, J.P., CORDNER, M., FONG, N. (2000), Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. Proceedings of SIGGRAPH 2000, Computer graphics proceedings, Annual conference series, pp. 165-172.

LIN J.I. (1998), DYNA3D: A nonlinear, explicit, three-dimensional finite element code for solid and structural mechanics, User manual, Methods development group, Lawrence Livermore National Laboratory.

MALCOLM, L.L. (1976), An experimental investigation of the frictional and deformational response of articular cartilage interfaces to statistic and dynamic loading. PhD thesis, Univ. of California San Diego.

MARCUM, D.L., and WEATHERILL, N.P., (1995) Unstructured grid generation using iterative point insertion and local reconnection. AIAA Journal, 33(9), pp. 1619-1625.

MILLER, K., CHINZEI, K., ORSSENGO, G., BEDNARZ, P. (2000), Mechanical properties of brain tissue in-vivo: experiment and computer simulation. Journal of Biomechanics 33, pp. 1369-1376.

MOONEY, M. (1940), A theory of large elastic deformation. Journal of Applied Physics. 11, pp. 582-592.

MORO-OKA, T., MIURA, H., HIGAKI, H., ARIMURA, S., MAWATARI, T., MURAKAMI, T., IWAMOTO, Y. (1999), A new friction tester of the flexor tendon. Journal of biomechanics, 32, pp 1131-1134.

MOW, V.C., KUEI, S.C., LAI, W.M., ARMSTRONG, C.G. (1980), Biphasic creep and stress relaxation of articular cartilage: theory and experiment. AMSE Journal of Biomechanical Engineering 102, pp 73-84.

O'BRIEN J.F., HODGINS J.K. (1999), Graphical modeling and animation of brittle fracture. Proceedings of SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series.

PADMANABHAN, V. LAURSEN, T.A. (1998), A new contact surface smoothing procedure for the implicit finite element analysis of frictional contact. Proceedings of the First International Symposium on Impact and Friction of Solids, Structures and Intelligent Machines, SERIES ON STABILITY, VIBRATION AND CONTROL OF SYSTEMS, Series B, Volume 14, pp. 297-300.

PAPADOPOULOS, P., and TAYLOR, R.L. (1993), A simple algorithm for three-dimensional finite element analysis of contact problems. Computers & Structures 46(6), pp. 1107-1118.

PIOLETTI, D.P., RAKOTOMANANA, L.R., BENVENUTI, J.F., LEYVRAZ P.F. (1998), Viscoelastic constitutive law in large deformations: application to human knee ligaments and tendons. Journal of Biomechanics 31, pp. 753-757.

PUSO, M. A., LAURSEN, T.A. (2001), A 3D Contact Smoothing Method Using Gregory Patches. International Journal for Numerical Methods in Engineering (under review).

REID, J.D., HARGRAVE, M.W., PAULSON, S.L. (2001), LS-DYNA: A computer modeling success story. Public Roads, 64, 4 (January/February), Federal Highway Administration.

RICHTMYER, R.D., MORTON, L.W. (1967), Difference methods for initial-value problems. Interscience, New York.

ROBLE, D. (2001), A talk in the SIGGRAPH panel session entitled “Newton's nightmare: reality meets faux physics”

SCHROEDER, W., MARTIN, K., LORENSEN, B. (1997), The visualization toolkit: an object-oriented approach to 3-d graphics. Prentice-Hall.

SEAGER, M. (1988), A SLAP for the masses. Lawrence Livermore Nat. Laboratory Technical Report, UCRL-100267.

SEDERBERG, T.W., and PARRY, S.R. (1986), Free-form deformation of solid geometric models. Proceedings of SIGGRAPH 86, Computer Graphics, 20, 4, pp. 151-160.

SETHIAN, J.A. (1999), Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science. Cambridge Univ. Press.

SMONDYREV, A., BERKOWITZ, M.L. (1999), Structure of DPPC/cholesterol bilayer at low and high cholesterol concentrations: molecular dynamics simulations. Biophysical Journal, 77, pp. 2075-2089.

SPITZER, V., ACKERMAN, M.J., SCHERZINGER, A.L., WHITLOCK, D. (1996), The visible human male: a technical report. J. Am. Med. Inform. Assoc. 3(2), Mar-Apr, pp. 118-130.

STATE, A, LIVINGSTON, M.A., HIROTA, G., GARRETT, W.F., WHITTON, M. C., FUCHS, H., PISANO, E.D. (1996) Technologies for Augmented-Reality Systems: realizing Ultrasound-Guided Needle Biopsies. Proc. of SIGGRAPH '96, New Orleans, LA, In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, pp. 439-446.

TERZOPOULOS, D., PLATT, J., BARR, A., FLEISCHER, K. (1987), Elastically deformable models. Proc. of SIGGRAPH'87, Computer Graphics, 21, 4, pp. 205-214.

TURNER, M.J., CLOUGH, R.W., MARTIN, W.C., TOPP, L.J. (1956), Stiffness and deflection analysis of complex structures, J. Aeron. Sci., 23, pp. 805–824.

ÜN, K., SPILKER, R.L. (2001), Finite element simulation of biphasic soft tissue contact with application to the shoulder joint. Proceedings of 23rd Conference of IEEE EMBS 2001 Istanbul, Turkey.

VERONDA, D.R., WESTMANN, R.A. (1970), Mechanical characterization of skin-finite deformation. Journal of Biomechanics 3, pp. 111-124.

WILHELMS, J., and VAN GELDER, A. (1997), Anatomically based modeling. Proceedings of SIGGRAPH 97, Computer Graphics Proceedings, Annual Conference Series, pp. 173-180.

ZHUANG, Y. (2000), Real-time simulation of physically-realistic global deformations. PhD thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley.