# Multiresolution Green's Function Methods for Interactive Simulation of Large-scale Elastostatic Objects and Other Physical Systems in Equilibrium

by

Douglas Leonard James

B.Sc., Applied Mathematics, University of Western Ontario, 1995

M.Sc., Applied Mathematics, University of British Columbia, 1997

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

**Doctor of Philosophy**

in

THE FACULTY OF GRADUATE STUDIES

Department of Mathematics

Institute of Applied Mathematics

**The University of British Columbia**

September 2001

# Abstract

This thesis presents a framework for low-latency interactive simulation of linear elastostatic models and other systems associated with linear elliptic partial differention equations. This approach makes it feasible to interactively simulate large-scale physical models.

Linearity is exploited by formulating the boundary value problem (BVP) solution in terms of Green's functions (GFs) which may be precomputed to provide speed and cheap lookup operations. Runtime BVPs are solved using a collection of Capacitance Matrix Algorithms (CMAs) based on the Sherman-Morrison-Woodbury formula. Temporal coherence is exploited by caching and reusing, as well as sequentially updating, previous capacitance matrix inverses.

Multiresolution enhancements make it practical to simulate and store very large models. Efficient compressed representations of precomputed GFs are obtained using second-generation wavelets defined on surfaces. Fast inverse wavelet transforms allow fast summation methods to be used to accelerate runtime BVP solution. Wavelet GF compression factors are directly related to interactive simulation speedup, and examples are provided with hundredfold improvements at modest error levels. Furthermore, hierarchical constraints are defined using hierarchical basis functions, and related hierarchical GFs are then used to construct an hierarchical CMA. This direct solution approach is suitable for hard real time simulation since it provides a mechanism for gracefully degrading to coarser resolution approximations, and the wavelet representations allow for runtime adaptive multiresolution rendering.

These GF CMAs are well-suited to interactive haptic applications since GFs allow random access to solution components and the capacitance matrix is the contact compliance used for high-fidelity force feedback rendering. Examples are provided for distributed and point-like interactions.

Precomputed multizone kinematic GF models are also considered, with examples provided for character animation in computer graphics.

Finally, we briefly discuss the generation of multiresolution GF models using either numerical precomputation methods or reality-based robotic measurement.

# Contents

# List of Tables

# List of Figures

# Acknowledgements

I would like to thank my advisor Dr. Dinesh Pai for his vision and commitment to interactive simulation, and for the mentoring, understanding and generosity he has shown me. This research would not have happened without his tireless encouragement, and any other research would probably have been less enjoyable and involved far fewer squishy toy models. I would also like to thank Dr. Anthony Peirce for originally sharing his interest in fast numerical methods for singular integral equations during my Master's degree. This inspiration eventually led me to research fast Green's function transforms in this thesis. Many thanks to my long time friend and now wife, Karen, for her endless support and understanding over the years, as well as for her genuine interest and helpful criticism. For the family members back home, thank you for your constant encouragement, even though it was not always clear what I was doing and why it had to be so far away. Finally, I would like to acknowledge the many IS (Interactive Simulation) and ACME people over the years (Jochen, Chris, Kees, John, Sam, Josh, Paul, Derek, Som, Mike, et al.) that in one way or another enriched my experience at UBC.

DOUGLAS LEONARD JAMES

*The University of British Columbia*
*September 2001*

This thesis is dedicated to my beautiful wife Karen May James. Thank you for your loving support and for filling these past four years with such wonderful memories. This work would not have been nearly as enjoyable without you.

# Chapter 1

# Introduction

Interactive multi-modal simulation of deformable objects, in which a user may manipulate flexible objects and receive immediate sensory feedback via human-computer interfaces, is a major challenge for computer graphics and virtual environments. Deformation is essential in computer animation for plausibly modeling the behavior of the human body, animals, and soft objects such as furniture upholstery, but interactive applications, such as computer games, have very limited computing budgets for 3D physical continuum simulation. Current virtual prototyping and assembly applications also require deformable and also flexible kinematic models suitable for interactive simulations such as assembly path planning. Deformable models have a long history (see §1.1) and, one might say, are well understood within the graphics, scientific and engineering communities. The challenge addressed by this thesis is the design of deformable models that are both sufficiently realistic to capture the relevant physics, and sufficiently fast for interactive simulation on ubiquitous computing hardware.

The difficulty is that traditional scientific computing algorithms are capable of generating solutions to relatively complex 3D deformation problems on time-scales of minutes or seconds, but interactivity requires solution times on the time-scales associated with human sensory perception. For example, an interactive elastic object simulation should be capable of providing solutions for visual display at sufficient frame rates (30 Hz), as well as haptic force feedback at kinesthetically convincing rates (1000 Hz), and perhaps even contact force responses for sound generation. Failure to achieve sufficiently fast solution rates introduces latency and results in an unconvincing simulation. Given such severe time constraints, it makes sense to do as much work ahead of time as possible so that solution costs during a simulation are smaller.

In recent years, *linear elastostatic Green's function models* (LEGFMs) have been shown to strike an attractive trade-off between realism and speed. The models are physically-based and are accurate for a class of relatively stiff deformable materials which tend to reach equilibrium quickly during continuous contact. The linearity of the model allows for the

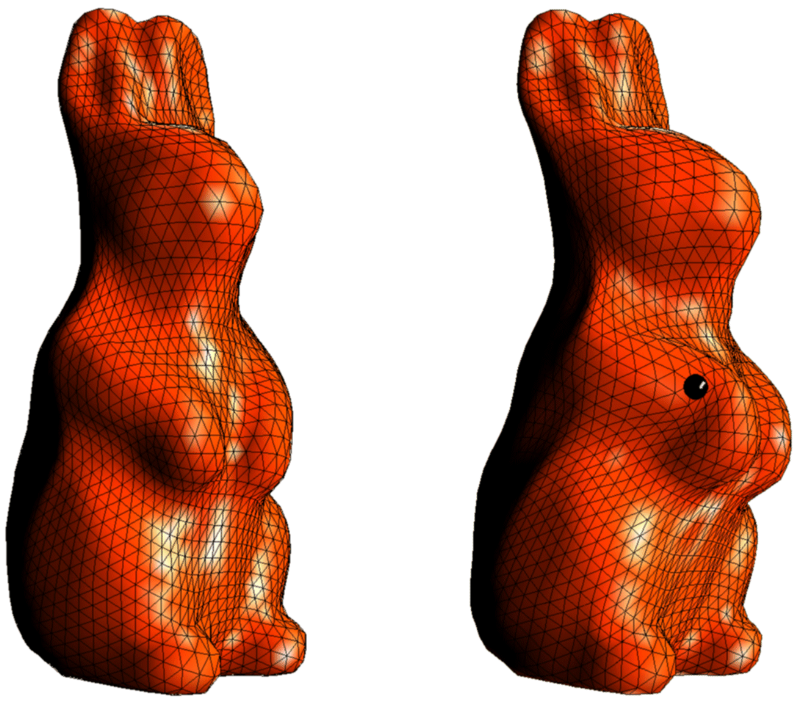


Figure 1.1: *Example of a Complex Elastic Model:* An elastic rabbit model with 2562 vertices, 5120 faces and 5 levels of subdivision connectivity ($L = 4$), capable of being rendered at 30 FPS with 1 kHz force feedback on a PC in our Java-based haptic simulation. The associated dense square Green's function (GF) submatrix contained 41 million floats (166 MB) but was compressed down to 655 thousand floats (2.6 MB) in this animation ($\varepsilon = .2$). The depicted deformation resulted from force interactions defined at a constraint resolution that was two levels coarser (L=2) than the visible mesh; for these coarse level constraints, the GF matrix block may be further compressed by a factor of approximately $16 = 4^2$. Even further compression is possible with file formats for storage and transmission of models. (Reparameterized rabbit model generated from mesh courtesy of Cyberware [Cyb].)

use of very fast solution algorithms based on linear superposition which support real-time rendering and stable force feedback. The use of these techniques in interactive simulation was advanced by [BC96, CDA99, JP99a] who demonstrated real time interaction with deformable models in applications such as force feedback surgical simulation and computer animation. A natural connection also exists between LEGFMs and active measurement techniques for the direct acquisition of deformable objects [PvdDJ$^+$01]. In this thesis we show that linear superposition techniques can lead to truly low-cost interactive simulation of small-strain deformations of extremely large models. Nevertheless, we also consider this a starting point for the development of a complete set of hybrid solution techniques with the ultimate goal of developing efficient simulation tools for difficult nonlinear phenomena associated with stiff models subject to constraints.

The key to the fast simulation technique is the use of precomputed *Green's functions* (GFs). Intuitively, Green's functions form a basis for representing all possible deformations of an object in a particular geometric configuration of boundary constraint types, e.g., essential (Dirichlet), natural (Neumann), or mixed (Robin). The benefit of linearity is that the response to any set of boundary values can be quickly reconstructed by a linear combination of precomputed GFs. In this way, these solution techniques can be used to obtain the solution for any set of applied constraints by using the GFs in combination with a collection of matrix updating methods (related to the Sherman-Morrison-Woodbury formula) which we refer to collectively as Capacitance Matrix Algorithms, or simply CMAs.

Furthermore, the GF-based CMA matrix solvers are not limited to just LEGFMs, and can in fact be used to simulate numerous other continuous physical systems in equilibrium. Interesting examples are the visualization of solutions to linear elliptic partial differential equations (PDEs), such as those used to model electrostatic fields, equilibrium diffusion, and transport phenomena. The unifying property is that any physical (or non-physical) equilibrium[1] system for which there exists a linear matrix relationship between specified and unspecified boundary values can be simulated; this is indeed a very large class of systems given that it contains all models described by discrete BVPs arising from discretizations of linear elliptic PDEs. An interesting point is that LEGFMs are small strain approximations of finite strain elasticity, however other physical systems are accurately modeled by linear elliptic PDEs, e.g., electrostatics. The reader should keep in mind that, although this thesis is largely explained in terms of deformable object simulation, parallel relationships exist between the physical quantities in other applications.

Recently we have shown that GF formulations play an important role in the direct acquisition of deformable models by active measurement [PvdDJ$^+$01]. Using the simulation methods here, it is possible for the quasi-GF measurements to be immediately used for interactive visualization. Multiresolution techniques developed here, such as hierarchical GFs, are also very useful in the context of measurement and estimation. The GF's input-output boundary description also provides an important starting point for the definition of inverse problems for determining internal material properties of deformable objects. Once the possibly quite complicated internal material distribution is estimated, a more complete simulation can then be undertaken, e.g., including nonlinear strain.

However, the CMAs achieve their impressive visualization speed ($O(sn)$ time per solution, where $n$ is the simulated model's geometric complexity and $s$ is the number of nonzero (or modified) boundary conditions) at the expense of storing $O(n^2)$ elements of

[1]We restrict our discussion to time-independent systems because, even though Green's function methods can be used for time-dependent systems, e.g., parabolic or even hyberbolic PDEs, the storage costs are exacerbated by the extra time dimension.

the large dense Green's function matrices which can be accessed in constant time. This clearly doesn't scale well to large models; for example the GF matrix stored as floats for a vertex-based model with 100 vertices requires only 360KB, however one with 10000 vertices such as dragon in Figure §3.5 (p. 56) would require 3.6GB! For these problems bus and cache considerations are significant as well. In addition, the capacitance matrix algorithm presented in [JP99a] requires an $O(s^3)$ factoring of the dense capacitance matrix, which scales very poorly as the number of run-time constraints increase.

In this thesis we present a family of algorithms for simulating deformable models and related systems that make Green's function techniques practical for very large models. The multiresolution enhancements introduced do much more than simply compress Green's functions to minimize storage. As a rule, these approaches are compatible with and improve the performance and real time feasibility of numerical operations required for the direct solution of boundary value problems. The algorithms exploit the fact that there exist several distinct, yet related, spatial scales corresponding to

- geometric detail,
- elastic displacement fields,
- elastic traction fields, and
- numerical discretization.

We develop multiresolution summation techniques to quickly synthesize deformations, and hierarchical capacitance matrix algorithms to deal with constraint changes. Wavelet GF representations are also useful for simulating multiresolution geometry for graphical and haptic rendering. Numerous other optimizations and applications are also presented, and are summarized in §1.2.

## 1.1 Related Work

This thesis borrows from and builds on several fields of research.

### 1.1.1 Traditional Numerical Methods for Linear Elastostatics

Static linear elasticity is an old and well-understood topic [Lov27]. Boundary integral equation (BIE) formulations of Navier's equation and other potential theories also have very established roots [Kel29, JS77]. Numerical methods for approximating the solutions of linear elasticity, such as the Finite Element Method (FEM) [Zie77] and, more recently, the Boundary Element Method (BEM) [BTW84], are well-known and commonly used. Efficient iterative methods exist for the solution of large scale problems, with notable examples being multigrid [Hac85] for domain discretizations and preconditioned fast multipole methods [GR87, YNK01] for boundary integral equations. Such methods are useful for

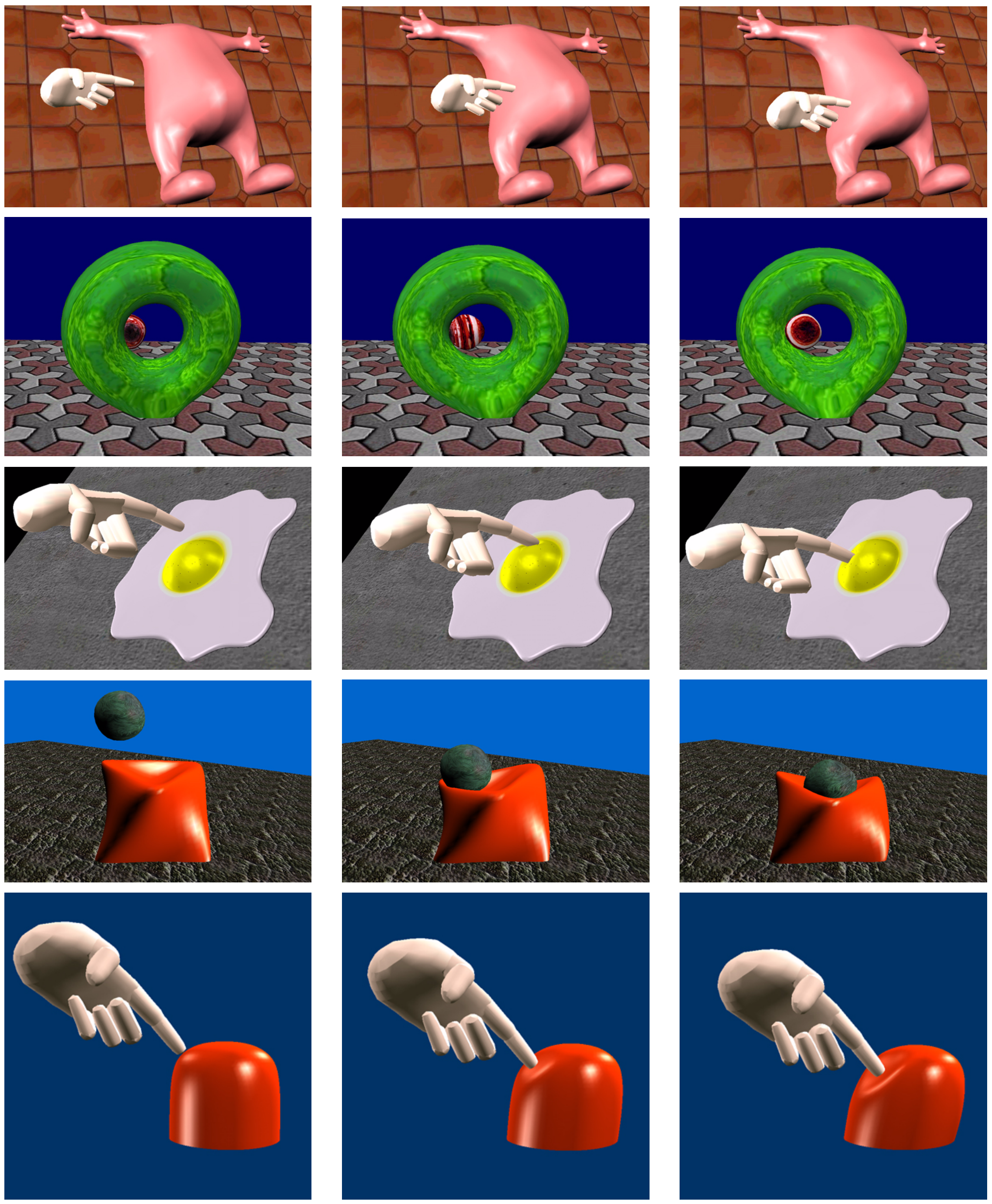


Figure 1.2: *Early examples* created using ARTDEFO the Java-based system described in [JP99a]. Each of these constant element BEM models has less than 300 nodes, and only a small fraction of these are ever contacted at the same time. The basic capacitance solver is adequate to handle these interactions and produce interactive frame rates on a PC. To efficiently interact with more complex models as well as allow more contacted nodes, the optimizations presented in this thesis are very useful.

precomputing Green's functions for simulations, but these (fast) solution methods aim to reduce the total solution time and were never intended for interactive simulation applications where low latency is more important than total solution time or asymptotic large-scale cost complexities. In this respect, a linear-time solution algorithm, such as multigrid or the fast multipole method, is not optimal for force feedback. On the other hand, while the capacitance matrix algorithm incurs significant precomputation costs, it is capable of computing force responses, e.g., in constant time for point-like contact, and this is more desireable.

### 1.1.2 Deformable Objects for Computer Graphics and Haptics

Substantial work has appeared on physical deformable object simulation and animation in computer graphics and related fields [TPBF87, BW92, GM97, CDA99, ZC00, DDBC01], although not all is appropriate for interactive or force feedback applications. Important interactive applications of elastic simulation include computer animation and interactive games, surgical simulation, computer aided design, interactive path planning, and virtual assembly for increasingly complicated manufacturing processes[2]. Several broad surveys are available for graphics [GM97] and surgical simulation [Del98], and we refer readers there for additional details.

From the point of view of this work, there are two clearly effective classes of interactive simulation methods for complex 3D physical elastic objects: methods for simulating stiff quasistatic systems under small strain, and explicit time-stepping schemes for simulating soft dynamic materials. This is primarily due to the fact that these approaches avoid the construction and solution of large systems of equations at each step of a simulation. The fast (quasi-)linear elastostatic models which can in a sense "precompute out stiffness" are complementary to explicit (lumped mass) time-stepping schemes most effective for soft, dynamic, highly deformable elastic models, e.g., [ZC00, PDA01, DDBC01]. Analogous to applications of rigid body simulation, equilibrium systems are useful for simulating stiff materials and in place of dynamic deformation that is too fast to be meaningfully resolved. Ever since the beginning of computer graphics, soft deformable objects have been simulated, and this is partly because they are much easier to compute than those involving stiff systems of equations [Wil89]. For force feedback interaction with stiff geometrically complex deformable models, in which model complexity and interaction fidelity are competing factors, fast approximation methods, such as those presented here, are very useful. Perhaps one of the most attractive features of LEGFM simulations is that it is possible to simulate contact force feedback at extremely high-rates without ever simulating the entire model; this is the reason that LEGFM simulations can achieve a much higher force feedback fidelity

[2]"Someone once said that a Boeing 747 is not really an airplane, but five million parts flying in close formation."(from [CM92])

than time-stepped simulations. We also expect that interactive simulations of deformable objects will involve more hybrid numerical models in the future.

In recent years, the importance of implicit[3] integration methods for numerically stiff simulations of constrained deformable dynamical systems has been revived in graphics [TF88, HE01]. Implicit methods have been very successfully applied to cloth simulation for off-line computer animation [BW98], and also for interactive simulation at constant frame rates using approximate cloth models of modest complexity [DSB99, KCC$^+$00]. Implicit-explicit (IMEX) time-stepping schemes developed for PDEs [ARW95] have also been used to address stiffness in the presence of nonlinearities and contact [KROM99, EEH00]. However, with the exception of simplified cloth models, implicit integration methods have remained uncommon for interactive simulation of 3D volumetric elastic models due to the need to solve a large linear system of (possibly changing) equations at each time step, although this might change with vector supercomputing hardware [Tay91].

There has been a natural recent trend toward explicit temporal integration of lumped mass nonlinear FEM systems, with examples using parallel computation [SBD$^+$00], adaptivity in space and perhaps time [ZC00, DDBC01, WDGT01] and also adaptive use of linear and nonlinear elements [PDA01]. These approaches are very useful for modeling soft materials in surgical simulation where a wide range of complex biological materials undergoing very large strain must be simulated and modified during virtual surgical procedures. Despite the appropriateness of this approach for human tissue modeling, it has several limitations which can be overcome in the case of linear elastostatic models: (1) only several hundred interior nodes can be integrated at a given time[4] (without special hardware), and while adaptivity does help, this ultimately limits model complexity; (2) deformations requiring that finely detailed discretizations be resolved will be slow due to CFL time-step restrictions; (3) while excellent for soft materials, stiff objects with detailed discretizations are difficult to time-step with explicit methods.

Multirate integration approaches are useful for introducing haptic interactions with dynamic models [cT00, DDBC01]. An interesting example is the work of Astley and Hayward [AH98] who precompute multilevel Norton equivalents for the stiffness matrix of a linear viscoelastic FEM model so that haptic interactions are possible by employing an explicit multirate integration scheme wherein a model associated with the contact region is

[3]A suitable background text on issues related to time-stepping methods [AP98] might also be consulted by the unfamiliar reader.

[4]Szekély et al. [SBD$^+$00] analyzed the time-stepping cost of their "optimized explicit finite-element algorithms" for nonlinear Cauchy-Green strain with a neo-Hookean strain energy functional. They arrived at a cost of 650 flops per element per time-step, or about 1000 flops including collision detection. Based on their restricted time-step of 100 $\mu$s they estimate that for a 2000 element model the sustained computational power required is 20 GFlops, hence their parallel computing approach. As shown in the Results chapter, this model is approximately one thousand times more costly than the corresponding LEGFM approximation.

integrated at a higher rate than the remaining coarser model. Local buffer models have also been introduced for deformable object simulations [Bal00, dBL00].

Modal analysis for linear elastodynamics [PW89] can also be used for interactive [Sta96] and force feedback applications [Bas01] by precomputing modal data. Related dimensional reduction time-stepping methods also exist for nonlinear solid dynamics, e.g., see [KLM01]. Sound models based on modal synthesis can also be used to render contact sounds [vdDP98, PvdDJ$^+$01]. The deformation method is costly as more modes are used, but as with the superposition of GF global deformation bases, the superposition of modal deformation bases can also be accelerated using fast-summation methods; for simple geometries, e.g., 1D, the FFT is commonly used. Unfortunately, while ideal for stiff free-vibrating dynamic models, resolving constraints associated with continuous contact interactions are problematic.

This thesis is most closely related to, and builds on, research on precomputed Green's function based models for real time simulation and force feedback interaction [BC96, HK98, CDA99, JP99a, JP01]. Of notable mention is the work on hepatic surgery simulation by the group at INRIA [BC96, CDA99, BN96]. In [BC96], a precomputation phase is used to condense [Zie77] a linear system of FEM equations to produce a dense boundary-only system of equations which are solved to obtain a set of surface Green's functions, while an iterative method is used in [CDA99]. During a laproscopic force feedback simulation they solve a small system of equations to determine the correct superposition of GFs, e.g., to apply displacement constraints at vertices of a triangle, which may be identified as a special case of the CMA. The approach taken by Hirota et al. [HK98] also involves precomputing what are GFs of a FEM model in order to impose a point contact constraint with force feedback rendering. Our work on the ARTDEFO simulator [JP99a] derived capacitance matrix updating equations in terms of GFs directly from the BEM matrix equations using the Sherman-Morrison-Woodbury formulae, and provided several examples of distributed contact constraints [JP99b]. The generality of this GF updating formula and implications for force feedback haptics were described in [JP01]. Despite their effectiveness, all of these approaches suffer from a potentially long precomputation period, and poorly scaling memory requirements which limit the complexity of models that can be constructed and/or simulated. Additionally, the basic CMA approach will ultimately degrade interactive performance for large contact areas. All of these short-comings are addressed by this thesis. Nevertheless, even with these limitations, the precomputed FEM model of [BC96] is of practical use, e.g., in brain surgery simulation [HL98] and Forschungszentrum Karlsruhe's commercial KISMET endoscopic surgery simulators [KcM99].

### 1.1.3 Subdivision and Wavelets

We make extensive use of multiresolution modeling related to Loop subdivision surfaces [Loo87] and their displaced variants [LMH00]. Such geometric subdivision schemes have proven to be highly successful geometric modeling tools [Sub00, War95, DKT98, CPD$^+$96, ZSS96, ZSS97]. For the BEM, notable benefits of subdivision meshing are high-quality surface triangulations and control over mesh uniformity. We also exploit the semi-regular subdivision mesh structure to define interpolating multiscale bases, such as surface adapted hierarchical basis functions [Yse86].

We are mostly concerned with the efficient representation of Green's functions defined on subdivision surfaces. Natural tools here are subdivision wavelets [LDW97], since geometric subdivision has deep connections with wavelets and the construction of multiresolution analyses (MRA) on general manifolds. We make extensive use of such wavelets based on the lifting scheme [Swe98, SS95a, SS96], since this provides a way to efficiently represent functions on arbitrary surfaces [SS95a, SS95b] as well as a means to construct fast $O(n)$ wavelet transforms for use in GF fast summation calculations.

Efficient GF representation is also related to the much larger area of multiresolution and progressive geometric representation [KSS00], and the efficient represention of functions on surfaces [KL97]. The issues governing choice of wavelets varies because of the very different quantities being represented. In particular, 3D geometry and quantities defined on it constitute a very broad class of functions, whereas displacement fields for 3D boundary Green's functions of elliptic PDEs are a very particular class of functions which are typically extremely smooth (or nearly constant) over very large portions of the surface with the exception of fast variations near constraints. For example, a surface region described by a GF may be locally undergoing translation (constant displacement field) while the underlying geometry is highly complex. For such reasons, we have observed that very simple wavelets, such as lifted linear wavelets, have performance comparable to often better performing smoother bases, such as lifted butterfly wavelets.

Our multiresolution elastostatic surface splines also have connections with variational and physically-based subdivision schemes [DLG90, WW98, WW99, WW00]. A relevant point is that the multiresolution framework presented here can be used to accurately solve boundary value problems arising from inhomogeneous linear partial differential equations by accurately modeling the nonlocal (dense) influences of (changing) boundary conditions. This contrasts with the local rules obtained for subdivision modeling, e.g., of linear Stokes flow [WW99], which do not solve arbitrary boundary value problems per se but rather provide a tool for convenient variational modeling of plausible physical solutions.

### 1.1.4 Fast Summation and Integral Transforms

Our work on wavelet Green's functions is strongly related to multiresolution discretization techniques [BCR91b, ABCR93] for sparsely representing integral operators and obtaining fast summations for fast matrix multiplication. However, unlike the cases from classical potential theory where the integral operator's kernel is explicitly known [JS77] and can be exploited [GR87, HN89, YNK01], or for *wavelet radiosity* in which the form factors may be extracted relatively easily [GSCH93], here the integral operator's discrete matrix elements are defined implicitly as Green's functions which are obtained by solving a class of boundary value problems. Nevertheless, it is known that such (Green's function) integral operators which describe the solutions to boundary value problems arising from elliptic partial differential equations may be efficiently represented in wavelet bases [Bey92]. However, unlike these multiresolution discretization approaches that are commonly used with iterative solution methods, we are not primarily interested in performing fast matrix-vector multiplies with the *full* Green's function matrix; we also require a number of specially optimized matrix-vector multiplication and matrix element extraction operations for use with capacitance matrix algorithm constraint solver.

### 1.1.5 Capacitance Matrix Algorithms

The main focus of this work is on capacitance algorithms used for updating matrices in the solution of linear algebraic systems. This topic has a long and interesting history [OW79, PW80, Dry83, CS85, Yip86, KT87, Che87, Hag89, GMW91, Rie92, LV98, AGH01] dating back to the pioneering work by Sherman, Morrison, Woodbury [She53, SM50, Woo50, PFTV87] and a few others. The range of application of these methods is very broad, and hundreds of papers have been published on updating matrices, especially for optimization and least squares [Hag89]. There is also a fundamental relation between capacitance methods and domain decomposition, imbedding and Schur complements, e.g., see [PV94, PV95] and references contained therein, and this is discussed later in the context of multizone models with precomputed GFs. Recently, a generalized capacitance matrix algorithm [LV98] has appeared for updating matrices with general dimensions and rank.

### 1.1.6 Capacitance Matrix Updating and Downdating

Methods for updating and downdating[5] the capacitance matrix inverse, in which rows and columns are respectively added or deleted, are very important tools for exploiting temporal coherence with the CMAs (discussed in §2.3). Matrix factorizations may also be efficiently updated [GL96, GMW91], in addition to updating the explicit capacitance matrix inverse.

[5] I will refer to updating and downdating collectively as updating.

This is not a new idea, as the updating of capacitance inverses and their factorizations has been a useful tool for a few decades, e.g., in linear programming and quasi-Newton recurrences [BGS70, BG66, Hag89]. Efficiently updating QR factorizations has been an important problem, e.g., for least squares (see [GL96] and references therein). It is possible to update LU factorizations [GL96, GGMS74, Ste79]; however for various reasons, e.g., pivoting requirements, this can be problematic. While there are nice properties of QR factorizations for the capacitance problem, e.g., stability, the generally larger cost of updating matrix factorizations compared to inverses leads us to explicitly consider inverse updating methods.

### 1.1.7 Static Reanalysis and Contact Mechanics

The engineering community uses updating methods for the analysis of elastic structures undergoing changes, e.g., during a design process, and it is generally referred to as *static reanalysis* (see [Aro76, KT87, BH93] for comprehensive reviews). These numerous direct solution techniques are very similar to the general matrix updating schemes, and it turns out that many are related to the Sherman-Morrison-Woodbury formula. It is also possible to extend reanalysis to handle nonlinear material changes [AGH01].

The formal description of contact between deformable/rigid objects using unilateral inequalities (precluding tensile contact tractions and material interpenetration) and further equations approximating friction and other contact physics, is the subject of contact mechanics [Joh85, BC00]. In the BE contact community, Sherman–Morrison–Woodbury related methods are commonly used to modify contact constraints between contacting elastic bodies [EO89, MAR93]; this provided an insight which eventually led us to use CMAs for solving contact problems interactively, e.g., in [JP99a]. Coupling elastic models together is also related to domain decomposition and multizone models, and this is discussed later in §5.1 in the context of precomputed GF models. While the methods described in this thesis are appropriate and intended for solving quasistatic contact problems, we focus on developing fast methods for computing the structural response of elastic models, and do not explicitly address the solution of contact problems. These latter issues are already addressed by the literature (although not in an interactive context), and we have made use of this information in our software for interactive simulation of deformation resulting from contact.

## 1.2 Thesis Organization and Contributions

The organization and contributions of each chapter of this thesis are as follows:

- Chapter 2 introduces the Capacitance Matrix Algorithm (CMA) for interactive simulation of Green's function (GF) based physical models (§2.1-2.2). We use a generic formulation and notation which separates discretization details from simulation issues. This formulation highlights the role of the capacitance matrix in the BVP solution process and also simplifies discussion of later material. The final section (§2.3) provides extensive detail on efficient methods for sequential updating and caching of capacitance matrix inverses to exploit temporal coherence in BVPs.

- Chapter 3 introduces several new multiresolution enhancements for the CMA. A summary of multiresolution analysis on manifolds is given for second-generation biorthogonal vertex-based wavelets with fast lifted wavelet transforms (§3.1). These are used to construct sparse *wavelet GF* approximations which yield substantial compression (§3.2). This also allows the definition of a fast summation CMA which significantly improves simulation speeds for large models (§3.3). A multiresolution constraint formalism based on hierarchical basis functions is also introduced (§3.4) to improve the efficiency of simulating detailed models, especially in the presence of numerous (updated) constraints. GFs corresponding to hierarchical constraints, or *hierarchical GFs* (§3.5), are used to define an hierarchical CMA (§3.6) also supporting fast summation. Approaches for multiresolution and detailed rendering are also discussed (§3.7).

- Chapter 4 outlines the special properties of the CMA for haptic interaction. Section 4.1 illustrates the special role of the capacitance matrix as a surface compliance, and its use for force feedback of distributed contact. The important special case of point-like contact for force feedback rendering is considered in §4.2. We introduce *pressure masks* for smooth force rendering and consistent scale-specific definition of boundary conditions for the otherwise ill-defined case of point contact.

- Chapter 5 presents several advanced modeling techniques for extending the GF simulation framework to include nonlinear strain and/or material properties. Aspects of multizone GF models for simulation are discussed (§5.1), and extensions for multizone elastostatic kinematic models with large strain are derived.

- Chapter 6 addresses the generation of GFs prior to simulation. Numerical precomputation is discussed in §6.1 with an emphasis on boundary integral methods; examples are provided for direct BEM matrix solver, as well as a preconditioned iterative method employing fast integral transforms based on wavelets. As an alternative to numerical precomputation, the acquisition of multiresolution GF models using reality based modeling techniques in introduced in §6.2.

- Chapter 7 presents various experimental results. Timings of key CMA operations are given in §7.2 for typical BVP solution and capacitance matrix inversion costs, and for sequential capacitance matrix inverse updating in §7.3. Benefits related to multiresolution enhancements are described in §7.4 for (hierarchical) GF compression and consequent fast summation speedup. The remainder of the chapter discusses force feedback simulation examples (§7.5), precomputation times (§7.6), and wavelet compression of reality based models (§7.7).

- Chapter 8 presents our conclusions and suggestions for future work.

Several appendices provide tutorials on boundary integral formulations of Navier's equation (§A) and the boundary element method (BEM) (§B), as well as a justification of our pressure mask approach for point contact (§C), and an overview of the system system developed during this thesis project (§D).

# Chapter 2

# Interactive Simulation of Green's Function Models using Matrix Updating Techniques

## 2.1 Linear Elastostatic Boundary Model Preliminaries

Linear elastostatic objects are generalized three dimensional linear springs, and as such they are useful modeling primitives for physically-based simulations. In this section, background material for a generic discrete Green's function (GF) description for a variety of precomputed linear elastostatic models is provided. While this chapter can stand on its own to a certain degree, it is not an introduction to this topic, and the reader unfamiliar with such models might also consult a suitable background reference before continuing [Bar92, Har85, Zie77, BTW84, JP99a]. Conceptually, GFs form a basis for describing all possible deformations of a linear elastostatic model subject to a certain class of constraints. This is useful because it (1) provides a common language to describe all discrete models, (2) subsumes extraneous discretization details by relating only physical quantities, and (3) clarifies the generality of the force feedback and multiresolution algorithms described later.

Another benefit of using GFs is that they provide an efficient means for exclusively simulating only boundary data (displacements and forces). This is useful when rendering of interior data is not required or in cases where it may not even be available, such as for reality-based models obtained via boundary measurement [PvdDJ$^+$01]. While it is possible to simulate various internal volumetric quantities (§2.1.5), simulating only boundary data involves less computation. This is sufficient since in interactive computer graphics we are primarily concerned with haptic interactions that impose surface constraints and provide feedback via visible surface deformation and contact forces.

### 2.1.1 Geometry and Material Properties

Given that the fast solution method is based on linear systems principles, essentially any linear elastostatic model with physical geometric and material properties is admissible. We shall consider models in three dimensions, although many arguments also apply to lower dimensions. Suitable models would of course include bounded volumetric objects with various internal material properties, as well as special subclasses such as thin plates and shells. Since only a boundary or interface description is utilized for specifying user interactions, other exotic geometries may also be easily considered such as semi-infinite domains, exterior elastic domains, or simply any set of parametrized surface patches with a linear response. Similarly, numerous representations of the surface and associated displacement shape functions are also possible, e.g., polyhedra, NURBS and subdivision surfaces [Sub00]. The undeformed boundary is denoted by $\Gamma$.

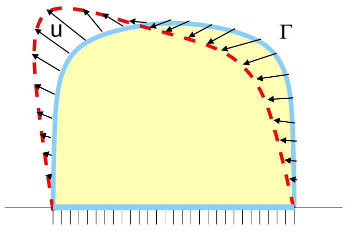


Figure 2.1: *Illustration of discrete nodal displacements* $\mathsf{u}$ defined at vertices on the undeformed boundary $\Gamma$ (solid blue line), that result in a deformation of the surface (to dashed red line). Although harder to illustrate, a similar definition exists for the traction vector, $\mathsf{p}$.

### 2.1.2 Nodal Displacements and Tractions

The change in shape of the surface is described by the surface *displacement field* $\mathbf{u}(\mathbf{x})$, $\mathbf{x} \in \Gamma$, and the surface force distribution is called the *traction field* $\mathbf{p}(\mathbf{x})$, $\mathbf{x} \in \Gamma$. We will assume that each surface field is parametrized by $n$ nodal variables (see Figure 2.1), so that the discrete displacement and traction vectors are

$$\mathsf{u} = [\mathsf{u}_1, \dots, \mathsf{u}_n]^{\mathsf{T}} \tag{2.1}$$

$$\mathsf{p} = [\mathsf{p}_1, \dots, \mathsf{p}_n]^{\mathsf{T}}, \tag{2.2}$$

respectively, where each nodal value is a vector in $\mathbb{R}^3$. This description admits a very large class of surface displacement and traction distributions, suitable for considering those associated with elasticity discretizations.

Each discrete vector field $\mathsf{u}$ and $\mathsf{p}$ have continuous counterparts, whose components belong to a continuous scalar space on the model's boundary, e.g.,

$$\mathcal{L} = \operatorname{span}\{\phi_j(\mathbf{x}), \quad j = 1 \dots n, \quad \mathbf{x} \in \Gamma\}, \tag{2.3}$$

where $\phi_j(\mathbf{x})$ is a scalar basis function associated with the $j^{th}$ node. The continuous traction field may then be defined as a 3-vector function with components in $\mathcal{L}$,

$$\mathbf{p}(\mathbf{x}) = \sum_{j=1}^{n} \phi_j(\mathbf{x})\mathsf{p}_j, \tag{2.4}$$

The force on any surface area is equal to the integral of $\mathbf{p}(\mathbf{x})$ on that area. We can then define the nodal force associated with any nodal traction as

$$\mathsf{f}_j = a_j \mathsf{p}_j \qquad \text{where} \qquad a_j = \int_\Gamma \phi_j(\mathbf{x}) d\Gamma_{\mathbf{x}} \tag{2.5}$$

defines the area associated with the $j^{th}$ node. A similar space exists for the continuous displacement field components, and is in general different from the traction field, and perhaps more smooth.

Our implementation uses linear boundary element models, for which the nodes are vertices of a closed triangle mesh model using Loop subdivision [Loo87]. Such surfaces are convenient for obtaining multiresolution models for rendering as well as smoothly parameterized surfaces suitable for BEM discretization and deformation depiction. We describe both the traction field and the polyhedral displacement field using continuous piecewise linear basis functions: $\phi_j(\mathbf{x})$ represents a "hat function" located at the $j^{th}$ vertex normalized so that[1] $\phi_j(\mathbf{x}_i) = \delta_{ij}$. Given our implementation, we shall often refer to node and vertex interchangeably. The displacement and traction fields both have convenient vertex-based descriptions

$$\mathsf{u}_j = \mathbf{u}(\mathbf{x}_j), \qquad \mathsf{p}_j = \mathbf{p}(\mathbf{x}_j), \tag{2.6}$$

where $\mathbf{x}_j$ is the $j^{th}$ vertex.

### 2.1.3 Discrete Boundary Value Problem (BVP)

At each step of the simulation, a discrete BVP must be solved which relates specified and unspecified nodal values, e.g., to determine deformation and force feedback forces. Without loss of generality, it shall be assumed that either position or traction constraints are specified at each boundary node, although this can be extended to allow mixed conditions, e.g., normal displacement and tangential tractions. Let nodes with prescribed displacement or traction constraints be specified by the mutually exclusive index sets $\Lambda_u$ and $\Lambda_p$, respectively, so that $\Lambda_u \cap \Lambda_p = \emptyset$ and $\Lambda_u \cup \Lambda_p = \{1, 2, ..., n\}$. In order to guarantee an equilibrium

[1] $\delta_{ij}$ is the Kronecker delta function,

$$\delta_{ij} = \begin{cases} 1, & i = j, \\ 0, & i \neq j \end{cases}$$

constraint configuration we will require that there is at least one displacement constraint, i.e., $\Lambda_u \neq \emptyset$. We shall refer to the $(\Lambda_u, \Lambda_p)$ pair as the system constraint or *BVP type*.

Typical boundary conditions for a force feedback loop consist of specifying some (compactly supported) displacement constraints in the area of contact, with free boundary conditions (zero traction) and other (often zero displacement) support constraints outside the contact zone. The solution to (2.8) yields the rendered contact forces and surface deformation.

Denote the unspecified and complementary specified nodal variables by

$$\mathsf{v}_j = \begin{cases} \mathsf{p}_j & : \quad j \in \Lambda_u \\ \mathsf{u}_j & : \quad j \in \Lambda_p \end{cases} \quad \text{and} \quad \bar{\mathsf{v}}_j = \begin{cases} \mathsf{u}_j & : \quad j \in \Lambda_u \\ \mathsf{p}_j & : \quad j \in \Lambda_p \end{cases}, \tag{2.7}$$

respectively. By linearity of the discrete elastic model, there formally exists a linear relationship between all nodal boundary variables

$$\boxed{0 = \mathsf{A}\mathsf{v} + \bar{\mathsf{A}}\bar{\mathsf{v}} = \mathsf{A}\mathsf{v} - \mathsf{z}} \tag{2.8}$$

where the BVP system matrix $\mathsf{A}$ and its complementary matrix $\bar{\mathsf{A}}$ are, in general, dense block $n$-by-$n$ matrices [Har85] with 3-by-3 blocks.

Body force terms associated with other phenomena, e.g., gravity, have been omitted for simplicity, but can be included since they only add an extra contribution to the $\mathsf{z}$ term. More generally (2.8) may be written

$$0 = \mathsf{A}\mathsf{v} + \bar{\mathsf{A}}\bar{\mathsf{v}} - \mathsf{b} = \mathsf{A}\mathsf{v} - \mathsf{z} \tag{2.9}$$

where

$$\mathsf{z} = \mathsf{b} - \bar{\mathsf{A}}\bar{\mathsf{v}}. \tag{2.10}$$

For later purposes, it is useful to parametrically describe contributions to $\mathsf{b}$ as[2]

$$\mathsf{b} = \mathsf{B}\beta = \sum_j \mathsf{B}_{:j}\beta_j \tag{2.11}$$

where $\beta$ are some scalar parameters. For example, gravitational body force contributions can be parameterized in terms of gravitational acceleration, $\mathbf{g} \in \mathbb{R}^3$.

A fundamental relationship between BVP system matrices $(\mathsf{A}, \bar{\mathsf{A}})$ arising from different BVP types $(\Lambda_u, \Lambda_p)$ is that they are related by exchanges of corresponding block columns, e.g., $(\mathsf{A}_{:\mathrm{j}}, \bar{\mathsf{A}}_{:\mathrm{j}})$. Therefore changes to a small number of nodes in the BVP type will result in low-rank changes to the BVP system matrices (see §2.2.3).

While the boundary-only system matrices in (2.8) could be constructed explicitly, e.g., via condensation [Zie77] or using a boundary integral formulation (see next section), it need not be in practice. Equation 2.8 is primarily a common starting point for later definition of GFs and derivation of the CMA, while *GFs may be computed by any convenient method*.

[2]Notation: Throughout we use a *colon subscript* to refer to all components of a matrix quantity, e.g., $\mathsf{B}_{:j}$ represents the $j^{th}$ (block) column vector of the matrix $\mathsf{B}$.

### 2.1.4 Example: Boundary Element Method (BEM) Models

Closed-form definitions of $(\mathsf{A}, \bar{\mathsf{A}})$ are possible for boundary element models ([BTW84, JP99a], Appendix B). BEM discretizations are possible for models with homogeneous and isotropic material properties. The surface nodal quantities are related by the dense linear block matrix system

$$\begin{aligned} 0 &= \mathsf{Hu} - \mathsf{Gp} & (2.12)\\ 0 &= \sum_{j=1}^{n} \mathsf{h}_{ij}\mathsf{u}_j - \sum_{j=1}^{n} \mathsf{g}_{ij}\mathsf{p}_j, \quad i = 1 \ldots n, & (2.13)\end{aligned}$$

where $\mathsf{G}$ and $\mathsf{H}$ are $n$-by-$n$ block matrices, with each matrix element, $\mathsf{g}_{ij}$ or $\mathsf{h}_{ij}$, a 3-by-3 influence matrix with known formulae [BTW84]. In this case, the $j^{th}$ block columns of $\mathsf{A}$ and $\bar{\mathsf{A}}$ may be identified as column exchanged variants of $\mathsf{G}$ and $\mathsf{H}$:

$$\mathsf{A}_{:j} = \begin{cases} -\mathsf{G}_{:j} & : \quad j \in \Lambda_u \\ \mathsf{H}_{:j} & : \quad j \in \Lambda_p \end{cases} \quad \text{and} \quad \bar{\mathsf{A}}_{:j} = \begin{cases} \mathsf{H}_{:j} & : \quad j \in \Lambda_u \\ -\mathsf{G}_{:j} & : \quad j \in \Lambda_p \end{cases} . \tag{2.14}$$

### 2.1.5 Fast BVP Solution with Green's Functions (GFs)

Green's functions (GFs) of a single BVP type provide an economical means for solving (2.8) for that BVP, and when combined with the CMA (§2.2) will also be useful for solving other BVP types. From (2.8), the general solution of a BVP type $(\Lambda_u, \Lambda_p)$ may be expressed in terms of discrete Green's functions (GFs)[3] as

$$\boxed{\mathsf{v} = \Xi\bar{\mathsf{v}} = \sum_{j=1}^{n} \xi_j \bar{\mathsf{v}}_j = \sum_{j\in\Lambda_u} \xi_j \bar{\mathsf{u}}_j + \sum_{j\in\Lambda_p} \xi_j \bar{\mathsf{p}}_j,} \tag{2.15}$$

where the discrete GFs of the BVP system are the block column vectors

$$\xi_j = -\left(\mathsf{A}^{-1}\bar{\mathsf{A}}\right)_{:j} \tag{2.16}$$

and

$$\boxed{\Xi = -\mathsf{A}^{-1}\bar{\mathsf{A}} = [\xi_1 \xi_2 \cdots \xi_n]\,.} \tag{2.17}$$

Equation (2.15) may be taken as the definition of the discrete GFs, since it is clear that the $j^{th}$ GF simply describes the linear response of the system to the $j^{th}$ node's specified

[3]Note on GF terminology: we are concerned with discrete numerical approximations of continuous GFs, however for convenience these GF vectors will simply be referred to as GFs.

boundary value, $\bar{\mathsf{v}}_j$ (see Figure 2.2). This equation may be interpreted as the discrete manifestation of a continuous Green's function integral equation[4]. Once the GFs have been computed for one BVP type, that BVP may then be solved easily using (2.15). An attractive feature for interactive applications is that the entire solution can be obtained in $18ns$ flops[5] if only $s$ boundary values (BV) are nonzero (or have changed since the last time step); moreover, individual components of the solution may also be computed independently at proportionately smaller costs, as shown below.

Parameterized body force contributions may also be included in (2.15) to yield the summation

$$\mathsf{v} = \Xi\bar{\mathsf{v}} + \left(\mathsf{A}^{-1}\mathsf{B}\right)\beta, \tag{2.19}$$

for which the matrix $\left(\mathsf{A}^{-1}\mathsf{B}\right)$ could be precomputed.

Temporal coherence may also be exploited by considering the effect of individual changes in components of $\bar{\mathsf{v}}$ on the solution $\mathsf{v}$. For example, given a sparse set of changes to the constraints, $\delta\bar{\mathsf{v}}$, if follows from (2.15) that the new solution can be incremented efficiently,

$$\bar{\mathsf{v}}^{new} = \bar{\mathsf{v}}^{old} + \delta\bar{\mathsf{v}} \tag{2.20}$$

$$\mathsf{v}^{new} = \mathsf{v}^{old} + \Xi\,\delta\bar{\mathsf{v}}. \tag{2.21}$$

By only summing contributions to constraints which have changed significantly, temporal coherence can be exploited to reduce BVP solve times and obtain faster frame rates.

Further leveraging linear superposition, each GF system response may be enhanced with additional information in order to simulate other precomputable quantities. In this way, volumetric stress, strain and displacement data may also be simulated at preselected locations. For example, a GF could be augmented with additional rows containing quantities to be superposed,

$$\xi_j^{modified} = \begin{bmatrix} \xi_j \\ \mathsf{u}_j^{volume} \\ \vdots \\ \sigma_j \end{bmatrix} \tag{2.22}$$

[4]The continuous representation may be written, in an obvious notation, as

$$v(\mathbf{x}) = \int_{\Gamma_u} \Xi_u(\mathbf{x},\mathbf{y})\bar{u}(\mathbf{y})d\Gamma_{\mathbf{y}} + \int_{\Gamma_p} \Xi_p(\mathbf{x},\mathbf{y})\bar{p}(\mathbf{y})d\Gamma_{\mathbf{y}} \tag{2.18}$$

[5]Flops convention [GL96]: count both $+$ and $\times$. For example, the scalar saxpy operation $y := a * x + y$ involves 2 flops, so that the 3-by-3 matrix-vector multiply accumulate, $\mathsf{v}_i := \Xi_{ij}\bar{\mathsf{v}}_j + \mathsf{v}_i$, involves 9 saxpy operations, or 18 flops.

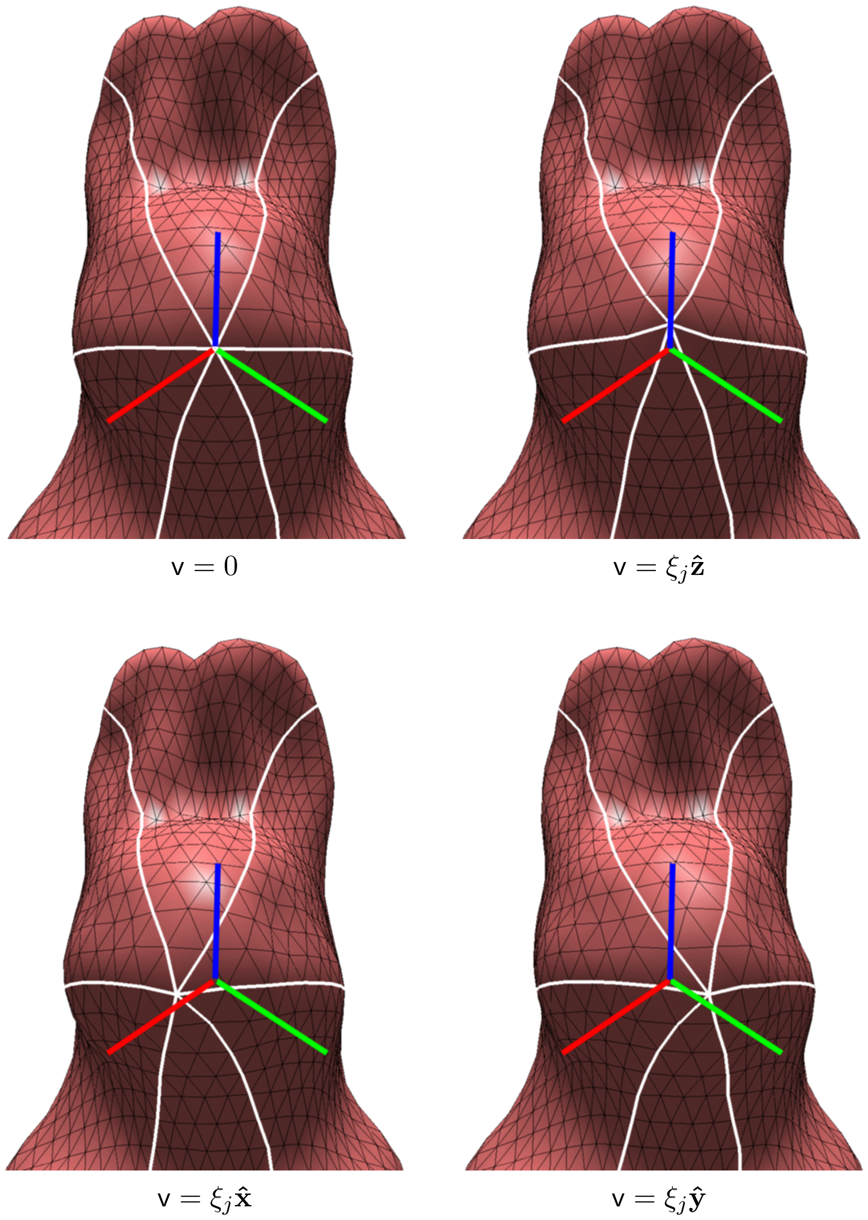

$\mathsf{v} = 0$
$\mathsf{v} = \xi_j \hat{\mathbf{z}}$
$\mathsf{v} = \xi_j \hat{\mathbf{x}}$
$\mathsf{v} = \xi_j \hat{\mathbf{y}}$

Figure 2.2: *Illustration of the $j^{th}$ Green's function block column*, $\xi_j = \Xi_{:j}$, representing the model's response due to the three XYZ components of the $j^{th}$ specified boundary value, $\bar{\mathsf{v}}_j$. Here the vertex belongs to the ("free") traction boundary, $j \in \Lambda_p$, and so $\xi_j$ is literally the three responses due to unit tractions applied in the (RGB color-coded) XYZ directions. White edges emanating from the (displaced) $j^{th}$ vertex help indicate the resulting deformation. Note that the vertex does not necessarily move in the direction of the XYZ tractions. Using linear superposition, the CMA can determine the combinations of these and other tractions required to move vertices to specified positions.

Applications could use this to monitor stresses and strains to determine, e.g., if fracture occurs or that a nonlinear correction should be computed. The multiresolution methods presented later can be extended to efficiently handle such volumetric data.

### 2.1.6 Precomputation of Green's Functions

Since the GFs for a single BVP type only depend on geometric and material properties of the deformable object, they may be precomputed for use in a simulation. Obtaining the GF deformation basis ahead of time is a key step that provides a dramatic speed-up for the simulation. While this is not necessarily a huge amount of work (see Table 7.7) (p. 118), the principal benefits for interactive simulations are reduced latency due to the availability of the GF elements via cheap look-up table operations, and elimination of redundant runtime computation. For example, using a haptic device to grab a vertex of the model and move it around simply renders a single GF, however an iterative method would recompute the solution each time.

Once a set of GFs for a LEGFM are precomputed, the overall stiffness can be varied at runtime by scaling BVP forces accordingly, however changes in compressibility and internal material distributions do require recomputation. In practice it is only necessary to compute the GF corresponding to nodes which may have changing or nonzero boundary values during the simulation.

Further details of the approaches taken for precomputation of GFs (using BEM models) as well as robotic measurement [PvdDJ$^+$01] are discussed in Chapter 6.

## 2.2 Fast Global Deformation using Capacitance Matrix Algorithms (CMAs)

This section presents an algorithm for using the precomputed GFs of a relevant *reference BVP* (RBVP) type to efficiently solve other BVP types by avoiding redundant computation and providing random access to solution components. This section on CMA lays the foundation for the framework of this thesis. With an improved notation and emphasis on haptics, this section also helps to unify and extend the approaches presented in [JP99a] exclusively for BEM models, and for FEM models in, e.g., [BC96], in a way that is applicable to all LEGFMs regardless of discretization. The key benefit of the formulation for haptic applications is that the capacitance matrix is the surface compliance of the contact zone. This is discussed further in §4.1.

### 2.2.1 Reference Boundary Value Problem (RBVP) Choice

A key step in the precomputation process is the identification of a RBVP type, denoted by $(\Lambda_u^0, \Lambda_p^0)$, that is similar to the BVP types arising during a simulation. For interactions with an exposed free boundary, a common choice is to have the uncontacted model attached to a rigid support (see Figure 2.3). The system matrices associated with the RBVP are denoted by[6] $\mathsf{A}_0$ and $\bar{\mathsf{A}}_0$, and the corresponding GFs will hereafter be simply represented by $\Xi$.

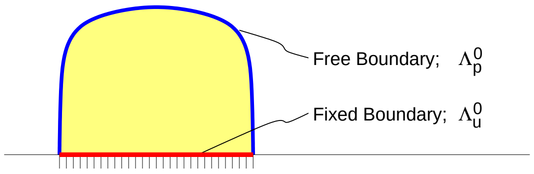


Figure 2.3: *Reference Boundary Value Problem (RBVP) Definition:* The RBVP associated with a model attached to a flat rigid support is shown with boundary regions having displacement ("fixed", $\Lambda_u^0$) or traction ("free", $\Lambda_p^0$) nodal constraints indicated. A typical simulation would then impose contacts on the free boundary via displacement constraints with the capacitance matrix algorithm.

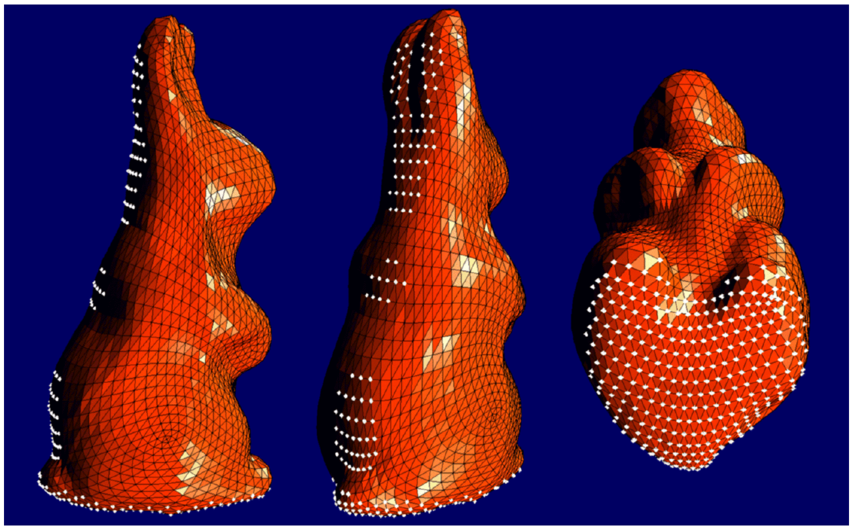


Figure 2.4: *Rabbit model Reference Boundary Value Problem (RBVP):* A RBVP for the rabbit model is illustrated with white dots attached to position constrained vertices in $\Lambda_u^0$. These (zero) displacement constraints were chosen to hold the rabbit model upright while users pushed on his belly in a force feedback simulation.

[6]It is convenient to use subscripts to identify $\mathsf{A}$ and $\bar{\mathsf{A}}$ matrices of different BVPs. Note that $\mathsf{A}_0$ and $\bar{\mathsf{A}}_0$ still represent square $n$-by-$n$ block matrices.

### 2.2.2 Sherman-Morrison-Woodbury Inverse Updating Formula

The Sherman-Morrison-Woodbury (SMW) formula is used to relate GFs of one BVP to those of another BVP, and is included here for reference.

Consider an $n$-by-$n$ block matrix $\mathsf{B}_0$, and a second block matrix related by a general rank $3s$ change

$$\mathsf{B} = \mathsf{B}_0 + \mathsf{U}\mathsf{V}^{\mathsf{T}}, \tag{2.23}$$

where both $\mathsf{U}$ and $\mathsf{V}$ are $n$–by–$s$ block matrices. The SMW formula [PFTV87, GL96] gives an expression for the rank $3s$ difference between $\mathsf{B}_0^{-1}$ and $\mathsf{B}^{-1}$, namely

$$\mathsf{B}^{-1} = \mathsf{B}_0^{-1} - \mathsf{B}_0^{-1}\mathsf{U}\mathsf{C}^{-1}\mathsf{V}^{\mathsf{T}}\mathsf{B}_0^{-1} \tag{2.24}$$

$$\mathsf{C} = \mathsf{I} + \mathsf{V}^{\mathsf{T}}\mathsf{B}_0^{-1}\mathsf{U} \tag{2.25}$$

where the $s$-by-$s$ block matrix $\mathsf{C}$ is called the *capacitance matrix*. The benefit of this is that only the smaller capacitance matrix must be inverted to obtain $\mathsf{B}^{-1}$ if $\mathsf{B}_0^{-1}$ is known. This formula is also useful for evaluating matrix-vector products with the new inverse without explicitly forming it, which is the spirit in which it will be used here.

### 2.2.3 Capacitance Matrix Algorithm (CMA) Formulae

Precomputed GFs speed-up the solution to the RBVP, but they can also dramatically reduce the amount of work required to solve a related BVP when used in conjunction with CMAs. If this were not so, precomputing Green's functions for a single BVP would have little practical use.

Suppose the constraint-type changes, e.g., displacement↔traction, with respect to the RBVP at $s$ nodes specified by the list of nodal indices

$$\mathsf{S} = \{\mathsf{S}_1, \mathsf{S}_2, \dots, \mathsf{S}_s\}. \tag{2.26}$$

As mentioned earlier, it follows from (2.7) and (2.8) that the new BVP system matrices $(\mathsf{A}, \bar{\mathsf{A}})$ are related to those of the RBVP $(\mathsf{A}_0, \bar{\mathsf{A}}_0)$ by $s$ block column swaps. This may be written as

$$\mathsf{A} = \mathsf{A}_0 + \left(\bar{\mathsf{A}}_0 - \mathsf{A}_0\right)\mathsf{E}\mathsf{E}^{\mathsf{T}} \tag{2.27}$$

$$\bar{\mathsf{A}} = \bar{\mathsf{A}}_0 + \left(\mathsf{A}_0 - \bar{\mathsf{A}}_0\right)\mathsf{E}\mathsf{E}^{\mathsf{T}} \tag{2.28}$$

where $\mathsf{E}$ is an $n$-by-$s$ block matrix

$$\mathsf{E} = \left[\mathsf{I}_{:\mathsf{S}_1}\mathsf{I}_{:\mathsf{S}_2}\cdots\mathsf{I}_{:\mathsf{S}_s}\right]. \tag{2.29}$$

containing columns of the $n$-by-$n$ identity block matrix, $\mathsf{I}$, specified by the list of updated nodal indices $\mathsf{S}$. Postmultiplication by $\mathsf{E}$ *extracts* columns specified by $\mathsf{S}$. Throughout, $\mathsf{E}$ is used to write sparse matrix operations using dense data, e.g., $\Xi$, and like the identity matrix, it should be noted that there is no cost involved in multiplication by $\mathsf{E}$ or its transpose.

An explicit formula for the GF matrix of the new BVP in terms of the old BVP's GF matrix $\Xi$ can be obtained using the Sherman-Morrison-Woodbury formula (§2.2.2) for $\mathsf{A}^{-1}$,

$$\mathsf{A}^{-1} = \mathsf{A}_0^{-1} - \mathsf{A}_0^{-1}\left(\bar{\mathsf{A}}_0 - \mathsf{A}_0\right)\mathsf{E}\mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}}\mathsf{A}_0^{-1}. \tag{2.30}$$

Substituting this in the expression for the new GFs and simplifying with the expression for the old GFs,

$$\Xi = -\mathsf{A}_0^{-1}\bar{\mathsf{A}}_0, \tag{2.31}$$

we obtain

$$\begin{aligned}
\Xi^{new} &= -\mathsf{A}^{-1}\bar{\mathsf{A}} && (2.32)\\
&= \left(\mathsf{I} + (\mathsf{E} + (\Xi\mathsf{E}))\mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}}\right)\left[\Xi(\mathsf{I} - \mathsf{E}\mathsf{E}^{\mathsf{T}}) - \mathsf{E}\mathsf{E}^{\mathsf{T}}\right] && (2.33)\\
&= \Xi + (\Xi + \mathsf{I})\mathsf{E}\mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}}(\Xi - \mathsf{I}) && (2.34)
\end{aligned}$$

It then follows immediately[7] that the BVP solution may be written in terms of the precomputed GFs. The resulting *capacitance matrix formulae* for $\mathsf{v}$ are

$$\boxed{\mathsf{v} = \underbrace{\mathsf{v}^{(0)}}_{n\times 1} + \underbrace{(\mathsf{E} + (\Xi\mathsf{E}))}_{n\times s}\underbrace{\mathsf{C}^{-1}}_{s\times s}\underbrace{\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)}}_{s\times 1}} \tag{2.35}$$

where $\mathsf{C}$ is the $s$-by-$s$ *capacitance matrix,* a negated submatrix of $\Xi$,

$$\boxed{\mathsf{C} = -\mathsf{E}^{\mathsf{T}}\Xi\mathsf{E},} \tag{2.36}$$

and $\mathsf{v}^{(0)}$ is the response of the RBVP system to $\mathsf{z}$, so that $\mathsf{A}_0\mathsf{v}^{(0)} = z$ and

$$\boxed{\mathsf{v}^{(0)} = \mathsf{A}_0^{-1}\mathsf{z} = -\mathsf{A}_0^{-1}\bar{\mathsf{A}}\bar{\mathsf{v}} = \left[\Xi\left(\mathsf{I} - \mathsf{E}\mathsf{E}^{\mathsf{T}}\right) - \mathsf{E}\mathsf{E}^{\mathsf{T}}\right]\bar{\mathsf{v}}.} \tag{2.37}$$

If body forces are included, it follows from (2.10) and (2.11) that (2.37) becomes

$$\mathsf{v}^{(0)} = \mathsf{A}_0^{-1}\mathsf{z} = -\mathsf{A}_0^{-1}\bar{\mathsf{A}}\bar{\mathsf{v}} + \left(\mathsf{A}_0^{-1}\mathsf{B}\right)\beta = \left[\Xi\left(\mathsf{I} - \mathsf{E}\mathsf{E}^{\mathsf{T}}\right) - \mathsf{E}\mathsf{E}^{\mathsf{T}}\right]\bar{\mathsf{v}} + \left(\mathsf{A}_0^{-1}\mathsf{B}\right)\beta. \tag{2.38}$$

### 2.2.4 A Capacitance Matrix Algorithm for Global Solution

With $\Xi$ precomputed, formulae (2.35)-(2.37) immediately suggest an algorithm given that only simple manipulations of $\Xi$ and inversion of the smaller capacitance submatrix is required. An algorithm for computing *all* components of $\mathsf{v}$ is as follows:

[7] alternately from [JP99a] with $\mathsf{A}_0^{-1}\delta\mathsf{A}_{\mathsf{S}} = (\mathsf{I} - \Xi)\mathsf{E}$.

- For each new BVP type (with a different $\mathsf{C}$ matrix) encountered, construct and temporarily store $\mathsf{C}^{-1}$ (or LU factors) for subsequent use.
- Construct $\mathsf{v}^{(0)}$.
- Extract $\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)}$ and apply the capacitance matrix inverse to it, $\mathsf{C}^{-1}(\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)})$.
- Add the $s$ column vectors $(\mathsf{E} + (\Xi\mathsf{E}))$ weighted by $\mathsf{C}^{-1}(\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)})$ to $\mathsf{v}^{(0)}$ for the final solution $\mathsf{v}$.

Each new capacitance matrix inversion/factorization involves $\mathcal{O}(s^3)$ work, after which each solve takes no more than $\mathcal{O}(ns)$ operations given $\mathcal{O}(s)$ nonzero boundary values. This is particularly attractive when $s \ll n$ is small, such as often occurs in practice with localized surface contacts.

An important feature of the CMA for interactive methods is that it is a direct matrix solver with a deterministic operation count. It is therefore possible to predict the runtime cost associated with each matrix solve and associated force feedback subcomputations (see §4.1), thus making CMAs predictable for real-time computations.

### 2.2.5 Numerical Stability of the CMA

The stability of solving a modified linear system (2.23) using the SMW formula is determined by the conditioning of the matrices involved, the relative scaling of matrix quantities, and the particular (nonunique[8]) choice of the $\mathsf{U}\mathsf{V}^{\mathsf{T}}$ update pair; Yip [Yip86] has shown that if $\mathsf{B}$ and $\mathsf{B}_0$ are well-conditioned, then there exists a choice of update pair such that the process is stable.

Since our formulation involves precomputed GF quantities, and so the capacitance matrices (submatrices of the GF matrix) are all explicitly known beforehand, understanding stability issues is slightly different. On the practical side, for the example models considered in this thesis, we have observed that the CMA, and in particular the capacitance matrix inversion and related updating techniques (in §2.3), have been numerically stable, and can be evaluated using 32-bit floating point operations provided that certain precautions are taken (see below). There are however cases where the CMA can fail, and we shall mention why this might occur.

In practice, the main precaution to be taken regards the appropriate relative scaling of quantities used in the capacitance matrix formulae (2.35)-(2.37). Since GFs are added and subtracted from columns of the identity matrix, to avoid problems the GF's should be of a similar magnitude. This means that the tractions and displacements described by the rows of the GF matrix, as well as the postmultiplied constraints, should be suitably normalized. Suitable nondimensionalization can be easily achieved by describing displacements

[8]Consider inserting an $s$-by-$s$ matrix factored as $\mathsf{X}\mathsf{X}^{-1}$.

as multiples of the object's length scale, and tractions in terms of the object's average shear modulus[9]. Without such precautions, the capacitance matrix inversion can easily lead to catastrophic cancellation errors and be numerically unstable. For example, failure to suitably rescale for an object of unit size with a large shear modulus, e.g., $10^8$, is asking for trouble in single precision arithmetic, and will also make precomputation more problematic.

This latter point leads to the second and more serious aspect of stability: the numerical conditioning of inverted matrices involved, i.e., $\mathsf{B}_0$ and $\mathsf{C} = \mathsf{I} + \mathsf{V}^{\mathsf{T}}\mathsf{B}_0^{-1}\mathsf{U}$. Because of our precomputation phase, in practice, all problems enter through the GF matrix. Put simply, the stability of the CMA, after suitable nondimensionalization, is limited by the conditioning of the GF matrix and its submatrices. First, the issue of the conditioning of $\mathsf{B}_0$ is really a moot point here, as $\mathsf{B}_0$ is a formality of our derivation, and has effectively been subsumed by the "black box" GF precomputation phase. In practice it is possible to precompute the GFs using a suitable stable discretization scheme, e.g., FEM or BEM, to a specifiable accuracy. Therefore the real stability concern is the capacitance matrix, whose expression (2.25) has been conveniently identified as a negated GF submatrix, thus avoiding further numerical concerns. Stably updating from one RBVP to all other valid BVP types requires that all possible capacitance matrices must also be well-conditioned; this is a much stronger condition than simply requiring that the GF matrix be well-conditioned. This also ensures that the GF summation (the last step of the CMA (§2.2.4)) is well behaved, so that GFs of one BVP type can be used to describe GFs of another BVP type.

The overall conditioning of the GF submatrices depends on several factors: geometry, material properties, underlying discretization, accuracy of precomputed GFs, and the RBVP choice. It is difficult to make any general statements regarding conditioning in this case. If necessary, GF conditioning properties could be quantified before simulation begins.

Finally, later in this thesis, methods are considered which introduce approximation errors into the GF matrix, e.g., by wavelet compression and/or reality-based GF estimation techniques. For a well-behaved GF matrix and sufficiently small errors there are no problems, however for GFs with bad conditioning and/or larger errors one is walking upon less stable ground. Nevertheless, for our presented wavelet GF examples, we have observed that the CMA has been stable long after the approximation's utility has vanished, e.g., for contact mechanics problems.

### 2.2.6 Selective Deformation Computation

A major benefit of the CMA with precomputed GFs is that it is possible to just evaluate selected components of the solution vector at runtime, with the total computing cost pro-

[9]Our BEM models are precomputed with unit shear modulus, $G$, with overall stiffness being modified at runtime by suitably scaling input/output tractions. On the other hand, Poisson's ratio can not be changed at runtime for 3D objects.

portional to the number of components desired. This random access to the BVP solution is a key enabling feature for haptics where, e.g., contact force responses are desired at different rates than the geometric deformations. Selective evaluation is also useful for optimizing (self) collision detection queries, as well as avoiding simulation of occluded or undesired portions of the model.

In general, any subset of solution components may be determined at a smaller cost than computing $\mathsf{v}$ entirely. Let the solution be desired at nodes specified by the set of indices $\mathsf{D}$, with the desired components of $\mathsf{v}$ extracted by $\mathsf{E}_\mathsf{D}^\mathsf{T}$. Using (2.35), the selected solution components may be evaluated as

$$\mathsf{E}_\mathsf{D}^\mathsf{T}\mathsf{v} \quad = \quad \mathsf{E}_\mathsf{D}^\mathsf{T}\mathsf{v}^{(0)} + \mathsf{E}_\mathsf{D}^\mathsf{T}\left(\mathsf{E} + (\Xi\mathsf{E})\right)\mathsf{C}^{-1}\mathsf{E}^\mathsf{T}\mathsf{v}^{(0)} \tag{2.39}$$

using only $\mathcal{O}(s^2 + s|\mathsf{D}|)$ operations. The case where $\mathsf{S} = \mathsf{D}$ is especially important for force feedback and is discussed exclusively in the following section.

Lastly, we mention that selective evalution already provides a mechanism for multiresolution rendering of displacement fields generated using the CMA algorithm. For example, random access allows displacements to be adaptively computed in a coarse to fine fashion, however, this will not reduce summation costs like the wavelet fast summation of §3.3.

## 2.3 Sequential Capacitance Matrix Inversion

In many physical simulations, there exists temporal coherence. Changes in BVP type are due to only small modifications to the set of updated nodes $\mathsf{S}$ (see Figure 2.5). Small modifications to $\mathsf{S}$ result in low rank modifications to the capacitance matrix $\mathsf{C}$. This temporal coherence can be exploited by again using matrix updating techniques to reduce the cost of matrix inversion (or factorization). In practice this "inversion" cost can be reduced from $\mathcal{O}(s^3)$ to $\mathcal{O}(s^2)$ operations in the presence of temporally coherent BVP types. So not only is the capacitance matrix algorithm useful for solving large systems of equations, but it is also a useful tool in its own implementation [Hag89].

### 2.3.1 Capacitance Matrix Inverse Updating Formulae

A formula for updating a capacitance matrix inverse for a slightly modified BVP, such as spreading contact, will now be presented. The benefit of this inversion approach is that the cost of inverting the capacitance matrix is reduced from $O(s^3)$ to $O(s^2 s_\Delta)$ operations, where $s_\Delta$ is the number of node changes between the two BVP. In practice, it should be expected that $s_\Delta \ll s$. Compared with updating factorizations, it turns out that it is (about four times) cheaper to work explicitly with the matrix inverse (shown in §2.3.2). This is

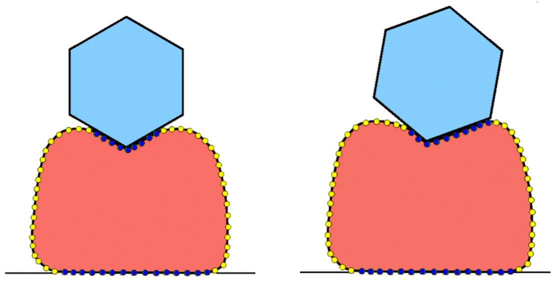


Figure 2.5: *Illustration of temporal coherence in BVP type during a contact sequence.*

also convenient in practice since it provides random access to individual elements of the inverse, e.g., for random access of force feedback stiffnesses.

Consider three BVPs: (#0) the RBVP with known GFs, (#1) the first updated BVP with updated node list $\mathsf{S}_1$ for which we know the capacitance matrix inverse $\mathsf{C}_1^{-1}$, and (#2) the second updated BVP for which we wish to determine the capacitance matrix inverse $\mathsf{C}_2^{-1}$, and which has an updated node list $\mathsf{S}_2 = \mathsf{S}_1 \oplus \mathsf{S}_\Delta$ resulting from $s_\Delta = |\mathsf{S}_\Delta|$ distinct nodes being added to $\mathsf{S}_1$. We will first only consider addition of nodes to $\mathsf{S}_1$ and then show in §2.3.5 that this can be used for efficient add and delete operations. The matrix view of the desired $\mathsf{C}_2^{-1}$ matrix with appended row and column (+) corresponding to an additional updated node being added to four others is

$$\mathsf{C}_2^{-1} = \left[\begin{array}{cccc|c} \times & \times & \times & \times & + \\ \times & \times & \times & \times & + \\ \times & \times & \times & \times & + \\ \times & \times & \times & \times & + \\ \hline + & + & + & + & + \end{array}\right]. \tag{2.40}$$

The necessary row expansion operators are defined as

$$\mathsf{E}_1 \equiv \mathsf{E}_{\mathsf{S}_1} \qquad \mathsf{E}_\Delta \equiv \mathsf{E}_{\mathsf{S}_\Delta}. \tag{2.41}$$

Expressions for the BVP matrices are

$$\mathsf{A}_1 = \mathsf{A}_0 + \left(\bar{\mathsf{A}}_0 - \mathsf{A}_0\right) \mathsf{E}_1 \mathsf{E}_1^T \tag{2.42}$$

$$\bar{\mathsf{A}}_1 = \bar{\mathsf{A}}_0 + \left(\mathsf{A}_0 - \bar{\mathsf{A}}_0\right) \mathsf{E}_1 \mathsf{E}_1^T \tag{2.43}$$

$$\mathsf{A}_2 = \mathsf{A}_1 + \left(\bar{\mathsf{A}}_1 - \mathsf{A}_1\right) \mathsf{E}_\Delta \mathsf{E}_\Delta^T \tag{2.44}$$

$$= \mathsf{A}_1 + \left(\bar{\mathsf{A}}_0 - \mathsf{A}_0\right) \mathsf{E}_\Delta \mathsf{E}_\Delta^T \tag{2.45}$$

where in the last line we made use of an identity which follows from the previous two expressions,

$$\bar{\mathsf{A}}_1 - \mathsf{A}_1 = \left[\bar{\mathsf{A}}_0 + \left(\mathsf{A}_0 - \bar{\mathsf{A}}_0\right) \mathsf{E}_1 \mathsf{E}_1^T\right] - \left[\mathsf{A}_0 + \left(\bar{\mathsf{A}}_0 - \mathsf{A}_0\right) \mathsf{E}_1 \mathsf{E}_1^T\right] \tag{2.46}$$

$$= \quad \bar{\mathsf{A}}_0 - \mathsf{A}_0 - 2\left(\bar{\mathsf{A}}_0 - \mathsf{A}_0\right)\mathsf{E}_1\mathsf{E}_1^{\mathsf{T}} \tag{2.47}$$

$$\Downarrow \tag{2.48}$$

$$\left(\bar{\mathsf{A}}_1 - \mathsf{A}_1\right)\mathsf{E}_\Delta \quad = \quad \left(\bar{\mathsf{A}}_0 - \mathsf{A}_0\right)\mathsf{E}_\Delta \tag{2.49}$$

since $\mathsf{E}_1^{\mathsf{T}}\mathsf{E}_\Delta = 0$ (as $\mathsf{S}_1 \bigcap \mathsf{S}_\Delta = \emptyset$).

We will introduce a convenient shortened notation for the column differences

$$\delta\mathsf{A}_1 \quad = \quad \left(\bar{\mathsf{A}}_0 - \mathsf{A}_0\right)\mathsf{E}_1 \tag{2.50}$$

$$\delta\mathsf{A}_\Delta \quad = \quad \left(\bar{\mathsf{A}}_0 - \mathsf{A}_0\right)\mathsf{E}_\Delta \tag{2.51}$$

so that

$$\mathsf{A}_1 \quad = \quad \mathsf{A}_0 \;+\; \delta\mathsf{A}_1\,\mathsf{E}_1^{\mathsf{T}} \tag{2.52}$$

$$\mathsf{A}_2 \quad = \quad \mathsf{A}_1 \;+\; \delta\mathsf{A}_\Delta\mathsf{E}_\Delta^{\mathsf{T}} \tag{2.53}$$

and the GF related quantities to be used in our final formulae are

$$\mathsf{B}_1 \quad = \quad -\mathsf{A}_0^{-1}\delta\mathsf{A}_1 \;= (\mathsf{I} + \Xi)\mathsf{E}_1 \tag{2.54}$$

$$\mathsf{B}_\Delta \quad = \quad -\mathsf{A}_0^{-1}\delta\mathsf{A}_\Delta = (\mathsf{I} + \Xi)\mathsf{E}_\Delta. \tag{2.55}$$

We define $\mathsf{v}^{(i)}$ as the BVP solutions to

$$\mathsf{A}_0\mathsf{v}^{(0)} \quad = \quad \mathsf{z} \tag{2.56}$$

$$\mathsf{A}_1\mathsf{v}^{(1)} \quad = \quad \mathsf{z} \tag{2.57}$$

$$\mathsf{A}_2\mathsf{v} = \mathsf{A}_2\mathsf{v}^{(2)} \quad = \quad \mathsf{z}. \tag{2.58}$$

The capacitance matrix formulae (2.35) implies that they are thus related by

$$\mathsf{v}^{(1)} \quad = \quad \mathsf{v}^{(0)} \;+ \mathsf{B}_1\,\mathsf{C}_1^{-1}\,\mathsf{E}_1^{\mathsf{T}}\,\mathsf{v}^{(0)} \tag{2.59}$$

and

$$\mathsf{v}^{(2)} \quad = \quad \mathsf{A}_2^{-1}\,\mathsf{z} \tag{2.60}$$

$$= \quad \mathsf{v}^{(1)} \;- \mathsf{A}_1^{-1}\,\delta\mathsf{A}_\Delta\mathsf{C}_\Delta^{-1}\mathsf{E}_\Delta^{\mathsf{T}}\mathsf{v}^{(1)} \tag{2.61}$$

$$= \quad \mathsf{v}^{(0)} \;+ \left[\begin{array}{c|c} \mathsf{B}_1 & \mathsf{B}_\Delta \end{array}\right]\;\mathsf{C}_2^{-1}\left[\begin{array}{c} \mathsf{E}_1^{\mathsf{T}} \\ \hline \mathsf{E}_\Delta^{\mathsf{T}} \end{array}\right]\;\mathsf{v}^{(0)} \tag{2.62}$$

where we wish to obtain $\mathsf{C}_2^{-1}$. Substituting expression (2.59) for $\mathsf{v}^{(1)}$ into expression (2.61) for $\mathsf{v}^{(2)}$, using the updating formula (2.30) for $\mathsf{A}_1^{-1}$, and factoring terms yields

$$\begin{aligned}
\mathsf{C}_2^{-1} \quad &= \quad \left[\begin{array}{c|c} \mathsf{C}_1^{-1} +\; \mathsf{C}_1^{-1}\,\mathsf{E}_1^{\mathsf{T}}\mathsf{B}_\Delta\,\mathsf{C}_\Delta^{-1}\mathsf{E}_\Delta^{\mathsf{T}}\mathsf{B}_1\,\mathsf{C}_1^{-1} & \mathsf{C}_1^{-1}\,\mathsf{E}_1^{\mathsf{T}}\mathsf{B}_\Delta\,\mathsf{C}_\Delta^{-1} \\ \hline \mathsf{C}_\Delta^{-1}\mathsf{E}_\Delta^{\mathsf{T}}\mathsf{B}_1\,\mathsf{C}_1^{-1} & \mathsf{C}_\Delta^{-1} \end{array}\right] \\
\mathsf{C}_\Delta^{-1} \quad &= \quad \left[\mathsf{I} +\; \mathsf{E}_\Delta^{\mathsf{T}}(\;\mathsf{A}_1^{-1}\;\;\delta\mathsf{A}_\Delta)\right]^{-1} \\
&= \quad \left[\mathsf{I} -\; \mathsf{E}_\Delta^{\mathsf{T}}\mathsf{B}_\Delta -\; \mathsf{E}_\Delta^{\mathsf{T}}\mathsf{B}_1\,\mathsf{C}_1^{-1}\,\mathsf{E}_1^{\mathsf{T}}\mathsf{B}_\Delta\right]^{-1}.
\end{aligned} \tag{2.63}$$

In terms of GF matrix blocks this becomes

$$\boxed{\begin{array}{rcl} \mathsf{C}_2^{-1} & = & \left[\begin{array}{c|c} \mathsf{C}_1^{-1} + \mathsf{C}_1^{-1}\mathsf{B}_{1\Delta}\mathsf{C}_\Delta^{-1}\mathsf{B}_{\Delta 1}\mathsf{C}_1^{-1} & \mathsf{C}_1^{-1}\mathsf{B}_{1\Delta}\mathsf{C}_\Delta^{-1} \\ \hline \mathsf{C}_\Delta^{-1}\mathsf{B}_{\Delta 1}\mathsf{C}_1^{-1} & \mathsf{C}_\Delta^{-1} \end{array}\right] \\ \mathsf{C}_\Delta^{-1} & = & \left[\mathsf{I} - \mathsf{B}_{\Delta\Delta} - \mathsf{B}_{\Delta 1}\mathsf{C}_1^{-1}\mathsf{B}_{1\Delta}\right]^{-1} \end{array}} \tag{2.64}$$

where

$$\mathsf{B}_{\mathsf{ab}} = \mathsf{E}_\mathsf{a}^\mathsf{T}\left(\mathsf{I} + \Xi\right)\mathsf{E}_\mathsf{b}, \quad \mathsf{a},\mathsf{b} \in \{1, \Delta\}. \tag{2.65}$$

Notice that the formula consists entirely of precomputed quantities and row extraction operations with the exception of the trivial $s_\Delta$–by–$s_\Delta$ matrix inversion $\mathsf{C}_\Delta^{-1}$.

### 2.3.2 Capacitance Matrix Inverse Updating Algorithm

Explicitly forming the new capacitance matrix inverse $\mathsf{C}_2^{-1}$ instead of leaving it in the factored form of (2.64), leads to more efficient evaluation and use in later updates. By carefully evaluating matrix subexpressions, it can be constructed at cost dominated by 3 $s \times s \times s_\Delta$ block matrix-matrix multiplies. The sequence of operations is

1. Lookup $\mathsf{C}_1^{-1}, \mathsf{B}_{1\Delta}, \mathsf{B}_{\Delta\Delta}, \mathsf{B}_{\Delta 1}$.
2. Dominant matrix-matrix multiply
$$\mathsf{D}_{1\Delta} = \mathsf{C}_1^{-1}\mathsf{B}_{1\Delta}. \tag{2.66}$$
*Cost:* $54 s_\Delta s^2$ flops.
3. Subdominant matrix-matrix multiply
$$\mathsf{D}_{\Delta\Delta} = \mathsf{B}_{\Delta 1}\mathsf{D}_{1\Delta} \tag{2.67}$$
*Cost:* $54 s_\Delta^2 s$ flops.
4. Construct $\mathsf{C}_\Delta$,
$$\mathsf{C}_\Delta = \mathsf{I} - \mathsf{B}_{\Delta\Delta} + \mathsf{D}_{\Delta\Delta} \tag{2.68}$$
*Cost:* $9 s_\Delta^2 + 3 s_\Delta$ flops.
5. Compute $\mathsf{C}_\Delta^{-1}$. *Cost:* $72 s_\Delta^3$ flops.
6. Two matrix-matrix multiplies (only $\mathsf{B}_{\Delta 1}\mathsf{C}_1^{-1}$ dominant)
$$\mathsf{D}_{\Delta 1} = \mathsf{C}_\Delta^{-1}\mathsf{B}_{\Delta 1}\mathsf{C}_1^{-1} \tag{2.69}$$
*Cost:* $54 s_\Delta^2 s + 54 s_\Delta s^2$ flops. (Not all columns are required when deleting nodes.)

7. Evaluate (required) elements of $\mathsf{C}_2^{-1}$ from the formula

$$\mathsf{C}_2^{-1} \quad = \quad \left[\begin{array}{c|c} \mathsf{C}_1^{-1} + \mathsf{D}_{1\Delta}\mathsf{D}_{\Delta 1} & \mathsf{D}_{1\Delta}\mathsf{C}_\Delta^{-1} \\ \hline \mathsf{D}_{\Delta 1} & \mathsf{C}_\Delta^{-1} \end{array}\right] \tag{2.70}$$

which involves the third and final dominant matrix-matrix multiply $\mathsf{D}_{1\Delta}\mathsf{D}_{\Delta 1}$. *Cost:* $54 s_\Delta s^2 + 54 s_\Delta^2 s + 9s^2$.

### 2.3.3 Cost Analysis of Updating Involving Node Addition

The total cost of performing this update to add $s_\Delta$ nodes to a previous system with $s = s_1$ nodes is

$$Cost_{Update} = 162(s_\Delta s_1^2 + s_\Delta^2 s_1 + \frac{4}{9}s_\Delta^3) + \{9(s_1^2 + s_\Delta^2) + 3s_\Delta\} \quad \text{flops} \tag{2.71}$$

where the last set of terms in brackets are subdominant for larger problems. In order to compare this cost to that of just performing LU decomposition (and inversion), it is useful to write the cost in terms of the dimension of the final resulting matrix, $s_2$, where

$$s_2 = s_1 + s_\Delta \quad \rightarrow \quad s_1 = s_2 - s_\Delta. \tag{2.72}$$

In this notation, the cost of LU decomposition is

$$Cost_{LU} = 18 s_2^3 \quad \text{flops} \tag{2.73}$$

and the cost of generating the inverse as well is 4 times larger. Therefore the cost ratio of updating to LU decomposition is

$$R = \frac{Cost_{Update}}{Cost_{LU}} = 9r(1 - r + \frac{4}{9}r^2) + \frac{9s_2^2 - 18 s_\Delta s_2 + 3s_\Delta}{18 s_2^3} \tag{2.74}$$

where $r = s_\Delta / s_2$. Neglecting the lower order fraction term which is only significant for small problems, e.g., $s_2 < 10$, we observe that updating is more efficient ($R<1$) when

$$9r(1 - r + \frac{4}{9}r^2) < 1 \quad \Rightarrow \quad r < 0.1261... \approx 0.12 \tag{2.75}$$

or when $s_\Delta < 0.12 s_2$. On the other hand, *inverse updating for node addition always involves fewer operations than generating the matrix inverse directly* (using LU decomposition and subsequent matrix inversion), since

$$9r(1 - r + \frac{4}{9}r^2) < 4 \quad \Rightarrow \quad r < 1. \tag{2.76}$$

For modest updates these methods perform very well in simulations; timings are shown in the Results chapter (§7). We will return to cost analysis for general add and delete update operations in §2.3.6.

### 2.3.4 Comparison to Factorization Updating

We note that the updated node set can be incremented more efficiently using this algorithm for inverses than existing algorithms for factorizations. As mentioned earlier, due to pivoting problems LU factorizations are not suitable for updating [GL96, GGMS74, Ste79], unlike QR factorizations which are commonly used [GL96].

For a cost comparison we refer to ([GL96], §12.4 "Updating Matrix Factorizations") where it is shown (in §12.5.1) that updating even a rank-one change to a QR factorization of an $n$-by-$n$ matrix requires about $26n^2$ flops. On the other hand, the Sherman-Morrison-Woodbury formula can be used to construct the inverse after a rank-one change using approximately 2 matrix-vector multiplies and one vector outer product for a total cost of $6n^2$ flops. Similarly, the algorithm just presented also only requires about $6n^2$ flops to append (or delete) a single *scalar* row and column to the capacitance matrix inverse.

### 2.3.5 Supporting Addition and Deletion of Updated Nodes

The formulae presented for updated node addition only are also useful for determining an algorithm supporting simultaneous addition and deletion. Updated nodes may be deleted by formally adding the node to be deleted a second time but negating the corresponding columns in $\delta \mathsf{A}_\Delta$ (and therefore $\mathsf{B}_\Delta$ also). This redundant update has the effect of subtracting the contribution to the solution produced by the undesired nodes, but conveniently provides a formula for the new capacitance matrix inverse: it is then obtained by only calculating those rows and columns of the large redundant capacitance matrix inverse not associated with deleted nodes.

Specifically, let the updating node sets be

$$\begin{aligned} \mathsf{S}_1 &= \mathsf{S}_{1_p} \oplus \mathsf{S}_{\Delta_-} && (2.77) \\ \mathsf{S}_\Delta &= \mathsf{S}_{\Delta_+} \oplus \mathsf{S}_{\Delta_-} && (2.78) \end{aligned}$$

where

$$\begin{aligned} \mathsf{S}_{1_p} &: \text{nodes to be persistent} && (2.79) \\ \mathsf{S}_{\Delta_+} &: \text{nodes to be added} && (2.80) \\ \mathsf{S}_{\Delta_-} &: \text{nodes to be deleted.} && (2.81) \end{aligned}$$

The redundant updating formula is then

$$\begin{aligned} \mathsf{v}^{(2)} &= \mathsf{v}^{(1)} - \mathsf{A}_1^{-1}\,\delta \mathsf{A}_\Delta \mathsf{C}_\Delta^{-1} \mathsf{E}_\Delta^T \mathsf{v}^{(1)} && (2.82) \\ &= \mathsf{v}^{(0)} + \left[\begin{array}{c|c} \mathsf{B}_1 & \mathsf{B}_\Delta \end{array}\right] \tilde{\mathsf{C}}_2^{-1} \left[\begin{array}{c} \mathsf{E}_1^T \\ \hline \mathsf{E}_\Delta^T \end{array}\right] \mathsf{v}^{(0)} && (2.83) \end{aligned}$$

$$= \quad \mathsf{v}^{(0)} + \left[\begin{array}{cc|cc} \mathsf{B}_{1_p} & \mathsf{B}_{\Delta_-} & \mathsf{B}_{\Delta_+} & (-\mathsf{B}_{\Delta_-}) \end{array}\right] \tilde{\mathsf{C}}_2^{-1} \left[\begin{array}{c} \mathsf{E}_{1_p}^T \\ \mathsf{E}_{\Delta_-}^T \\ \hline \mathsf{E}_{\Delta_+}^T \\ \mathsf{E}_{\Delta_-}^T \end{array}\right] \mathsf{v}^{(0)} \qquad (2.84)$$

where the over-sized redundant capacitance inverse $\tilde{\mathsf{C}}_2^{-1}$ contains unnecessary row and columns associated with the deletion process ($-$ elements)

$$\tilde{\mathsf{C}}_2^{-1} = \left[\begin{array}{cccc|c||cc|c} \times & \times & \times & \times & - & + & + & - \\ \times & \times & \times & \times & - & + & + & - \\ \times & \times & \times & \times & - & + & + & - \\ \times & \times & \times & \times & - & + & + & - \\ \hline - & - & - & - & - & - & - & - \\ \hline\hline + & + & + & + & - & + & + & - \\ + & + & + & + & - & + & + & - \\ \hline - & - & - & - & - & - & - & - \end{array}\right] \qquad (2.85)$$

$$(2.86)$$

and is not explicitly computed. Instead, the desired capacitance matrix inverse $\mathsf{C}_2^{-1}$ only contains elements not associated with deletion

$$\mathsf{C}_2^{-1} = \left[\begin{array}{cccc|cc} \times & \times & \times & \times & + & + \\ \times & \times & \times & \times & + & + \\ \times & \times & \times & \times & + & + \\ \times & \times & \times & \times & + & + \\ \hline + & + & + & + & + & + \\ + & + & + & + & + & + \end{array}\right]. \qquad (2.87)$$

The nonredundant updating formula for the BVP solution corresponding to updated nodes

$$\mathsf{S}_2 = \mathsf{S}_{1_p} \oplus \mathsf{S}_{\Delta_+} \qquad (2.88)$$

is then

$$\mathsf{v}^{(2)} \quad = \quad \mathsf{v}^{(0)} + \left[\begin{array}{c|c} \mathsf{B}_{1_p} & \mathsf{B}_{\Delta_+} \end{array}\right] \mathsf{C}_2^{-1} \left[\begin{array}{c} \mathsf{E}_{1_p}^T \\ \hline \mathsf{E}_{\Delta_+}^T \end{array}\right] \mathsf{v}^{(0)}\,. \qquad (2.89)$$

Note that while the inverse of the rank-deficient matrix $\tilde{\mathsf{C}}_2$ does not exist, applying the algorithm of §2.3.2 to compute the nonredundant entries of $\mathsf{C}_2^{-1}$ is not only efficient but numerically stable in practice. Finally, for implementation our updated node lists are sorted for efficient searching, and the block structure shown in the capacitance matrix inverse illustrations is for pedagogical purposes only.

Lastly we mention that a more direct approach to deletion is to use updating on the capacitance matrix to zero the interaction between deleted nodes and persistent (and added) nodes. This is accomplished by using updating to replace the capacitance matrix's deleted node's $d^{th}$ row and column with zeros except for a one on the diagonal,

$$\mathsf{C}_1 = \left[\begin{array}{cc|c|cc} \times & \times & - & \times & \times \\ \times & \times & - & \times & \times \\ \hline - & - & - & - & - \\ \hline \times & \times & - & \times & \times \\ \times & \times & - & \times & \times \end{array}\right] \qquad \Rightarrow \qquad \tilde{\mathsf{C}}_2 = \left[\begin{array}{cc|c|cc} \times & \times & 0 & \times & \times \\ \times & \times & 0 & \times & \times \\ \hline 0 & 0 & 1 & 0 & 0 \\ \hline \times & \times & 0 & \times & \times \\ \times & \times & 0 & \times & \times \end{array}\right]. \tag{2.90}$$

The new inverse is then only evaluated for the persistent (and added) elements. This is a rank two update, which would seem to cost about twice as much to evaluate as the previous approach, but because of sparsity in the update (let $\mathsf{C} = \mathsf{C}_1$)

$$\mathsf{UV}^\mathsf{T} = [(\mathsf{C}_{dd}\mathsf{I}_{:d} - \mathsf{C}_{:d}) \quad \mathsf{I}_{:d}][\mathsf{I}_{:d} \quad (-\mathsf{C}_{d:})]^\mathsf{T}, \tag{2.91}$$

it is about the same. The advantage of the algorithm from this section is that it does not require a sparse matrix implementation to be efficient, and the dominant updating work only involves optimized Level 3 BLAS operations.

### 2.3.6 Cost Analysis of Updates Involving Addition and Deletion of Nodes

This section provides floating point operation counts for the general updating case, and is slightly more involved than the case involving only added nodes in §2.3.3, since for deletion not all elements of the expanded $\mathsf{C}_2^{-1}$ matrix require computation. We shall first use the analysis of §2.3.3, and then subtract the redundant operations. As in §2.3.3, we will define the final number of nodes,

$$s_2 = s_1 - s_- + s_+ \quad \rightarrow \quad s_1 = s_2 + s_- - s_+ \tag{2.92}$$

where $s_2 = |\mathsf{S}_2|$, $s_1 = |\mathsf{S}_1|$, $s_- = |\mathsf{S}_{\Delta_-}|$ and $s_+ = |\mathsf{S}_{\Delta_+}|$ The number of updated nodes is

$$s_\Delta = s_- + s_+. \tag{2.93}$$

Therefore from (2.71) the cost of performing the update, including the computation of redundant entries associated with deletion, is (leading order terms)

$$Cost_{Update} = 162(s_- + s_+)\left((s_2 + s_- - s_+)(s_2 + 2s_-) + \frac{4}{3}(s_- + s_+)^2\right) \quad \text{flops.} \tag{2.94}$$

Some avoidable redundant deletion-related operations listed in the algorithm of §2.3.2 are as follows:

1. (Step 2) Avoid computing $s_-$ rows of

$$\mathsf{D}_{1\Delta} = \mathsf{C}_1^{-1}\mathsf{B}_{1\Delta}. \tag{2.95}$$

*Avoid:* $54 s_- s_\Delta s_1$ flops.

2. (Step 6) Avoid computing $s_-$ columns of

$$\mathsf{D}_{\Delta 1} = \mathsf{C}_\Delta^{-1}\mathsf{B}_{\Delta 1}\mathsf{C}_1^{-1} \tag{2.96}$$

*Avoid:* $54 s_- s_\Delta s_1$ flops.

3. (Step 7) Use row and column reduced forms of $\mathsf{D}_{1\Delta}$ and $\mathsf{D}_{\Delta 1}$, respectively, to compute $\mathsf{D}_{1\Delta}\mathsf{D}_{\Delta 1}$ and $\mathsf{D}_{1\Delta}\mathsf{C}_\Delta^{-1}$. *Avoid:* $54 s_- s_\Delta (2s_1 - s_-) + 54 s_- s_\Delta^2$ flops.

In total we may avoid

$$108 s_- s_\Delta s_1 + 54 s_- s_\Delta (2s_1 - s_-) + 54 s_- s_\Delta^2 \quad \text{flops.} \tag{2.97}$$

Subtracting these operations from (2.94) and using (2.92) for $s_1$, the total updating cost (to leading order) in units of the LU decomposition cost is

$$\begin{aligned} R &= \frac{Cost_{Update}}{Cost_{LU}} & (2.98)\\ &= \{9\,r_+ + 9\,r_-\} + \left\{-9\,r_+^2 + 6\,r_- r_+ + 15\,r_-^2\right\} & (2.99)\\ &\quad + \left\{+4\,r_+^3 + 3\,r_- r_+^2 + 9\,r_-^2 r_+ + 10\,r_-^3\right\} \end{aligned}$$

where

$$r_+ = \frac{s_+}{s_2}, \qquad r_- = \frac{s_-}{s_2}. \tag{2.100}$$

The cost function reduces to (2.74) for purely additive updating ($r_- = 0$)

$$R = 9\,r_+ - 9\,r_+^2 + 4\,r_+^3, \tag{2.101}$$

while for updates only involving deletion ($r_+ = 0$) it is

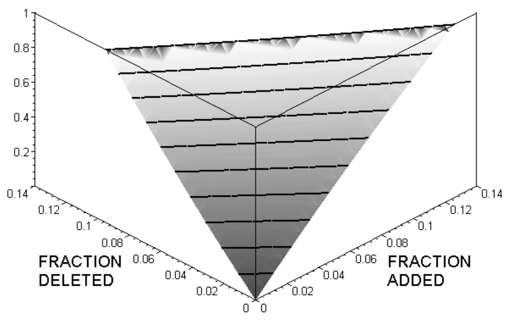
$$R = 9\,r_- + 15\,r_-^2 + 10\,r_-^3, \tag{2.102}$$

which justifies the claim that pure deletion updates are more costly than pure addition. For comparison, pure deletion updates are cheaper than LU decomposition ($R < 1$) and LU inverse generation ($R < 4$) when

$$R < 1 \quad \to \quad r_- < 0.0950... \approx 0.095 \tag{2.103}$$

$$R < 4 \quad \to \quad r_- < 0.2842... \approx 0.28 \tag{2.104}$$

For the general case, plots of the relative cost functional (2.99) are shown in Figure 2.6.



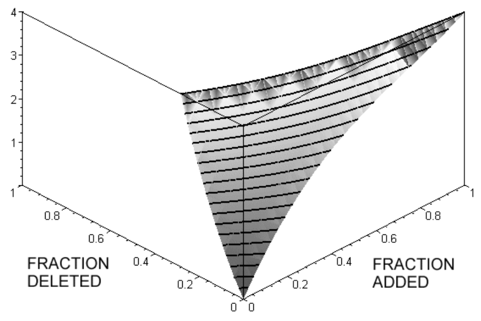


Figure 2.6: *Plot of capacitance matrix inverse updating costs* in units of corresponding LU decomposition. Plots of $R$ (equation 2.99) versus the fraction of updated nodes added ($r_+$, "ADDED") and deleted ($r_-$, "DELETED"). (Left) $R < 1$ range associated with updates cheaper than LU decomposition, (Right) $R < 4$ range associated with updates cheaper than LU decomposition followed by inverse generation. In each case, it is clearly evident that it is more costly to delete nodes than it is to add them. In fact, updating is always more efficient than LU inversion when deletion is not required. (Note: Shading irregularities are artifacts of Maple 6.)

### 2.3.7 Prediction of Updating Costs

For real time applications, predicting the cost of updating a capacitance matrix inverse for a set of node addition and deletion operations is an important topic. We do this by constructing an updating cost functional $\rho(\mathsf{S}_1, \mathsf{S}_2)$ which estimates the cost of updating from a BVP with updated node set $\mathsf{S}_1$ to one with $\mathsf{S}_2$ (where the lists are nonempty and unequal). This is especially useful when maintaining a cache of capacitance matrix inverses (topic of following section), for then it can be decided which of several cached BVP inverses may be updated most easily, or if the inverse should be constructed from scratch, or that the fastest solution is too slow and something more dramatic must occur, e.g., multiresolution degradation (§3.6).

Prediction of updating and direct inversion costs is performed using a polynomial cost model calibrated using software run times on the same computer platform used for simulation. This takes implementation into account, since flop counts are not always a good indication of run times. The cost models for direct inversion and updating are

$$\rho_{LUDInverse} = \sum_{0\le\alpha\le 3} a_\alpha s_2^\alpha \tag{2.105}$$

$$\rho_{Updating} = \sum_{0\le\alpha_2,\alpha_+,\alpha_-\le 3} a_{\alpha_2\alpha_+\alpha_-} s_2^{\alpha_2} s_+^{\alpha_+} s_-^{\alpha_-} \tag{2.106}$$

The coefficients were calibrated using preevaluated timings for a relevant range of values.

### 2.3.8 Capacitance Matrix Inverse Caching Strategies

When relatively small changes occur in the capacitance node set, it is more efficient to update the previous capacitance matrix inverse than it is to compute the inverse from scratch. For situations in which the temporal coherence is reduced and larger changes occur, it can be very useful to use a more sophisticated caching strategy than simply trying to update the current capacitance matrix inverse.

Quickly querying a modest database of already computed updates is worthwhile if it results in sufficiently less updating work. Given a set of candidate updating node lists

$$\{\mathsf{S}_1, \mathsf{S}_2, \dots, \mathsf{S}_d\} \tag{2.107}$$

for which inverses are cached, we compute the minimum predicted updating cost

$$\rho^{min}_{Updating} = \min_i \rho(\mathsf{S}_i, \mathsf{S}') \tag{2.108}$$

to obtain the desired BVP list, $\mathsf{S}'$. This minimum cost computation is very cheap, and lookup tables can be used to reduce this cost further. The cheapest update is performed if its estimated cost is less than that of computing the inverse from scratch, $\rho_{LUDInverse}$. For large datasets, it will be important to cull "distant" elements so that time is not wasted computing the cost functional; we suspect that this could be efficiently achieved by cross-indexing BVPs on different criteria, e.g., using a spatial hierarchy of updated node populations.

The effectiveness of a caching strategy depends heavily on the sequence of BVPs required by an application, however for physical simulations with temporal coherence we have found that caching can be very useful. In the presence of extreme temporal coherence, e.g., only one add/delete at a time, only the previous inverse is required, however for larger updates caching is effective. A useful strategy is to cache each constructed inverse with a key indicating the time it was last used. Additions are accompanied by randomized deletion of older entries such that the total memory usage is bounded. A small nursery of recent inverses contains the entries which are impervious to deletion and are best updating candidates in the presence of coherence.

# Chapter 3

# Enhancements for Multiresolution Simulation

The Green's function (GF) based capacitance matrix algorithm (CMA) has many appealing qualities for simulation, however it does suffer inefficiencies when used for complex geometric models or large systems of updated constraints. Fortunately, these limitations mostly arise from using dense matrix representations for the discrete GF integral operator, and can be overcome by using multiresolution bases to control the amount of data that must be manipulated. This chapter provides several multiresolution enhancements for the CMA which generally extend its applicability. We begin with a summary of the multiresolution analysis tools that will be used throughout the remainder of this work.

## 3.1 Summary of Fast Lifted Wavelet Transforms on Manifolds

This section provides the necessary notation and background to describe our use of second generation biorthogonal wavelets based on the *lifting scheme* [Swe98, SS95a, DS96] and a common notation to describe hierarchical bases. This section is a summary of material presented in [Swe98, SS95a] and uses similar notation. Greater mathematical detail on wavelets can be found elsewhere, e.g., [Dau92, CDF92]. With the exception of a very minor clarification of the construction of filters at surface domain boundaries, we have nothing new to add to this material.

The reader who is familiar with this material may skip this summary upon first reading and proceed directly to "Wavelet Green's Functions" in Section 3.2.

### 3.1.1 Multiresolution Analysis

This section describes the concepts and equations behind vertex-based multiresolution analysis using biorthogonal wavelets.

Given a space $L_2 = L_2(\Gamma, d\Gamma)$ describing vertex-based functions on the surface, the first step is to construct a sequence of nested spaces $V_j \subset L_2$, $j \geq 0$, where [SS95a]

- $V_j \subset V_{j+1}$ (finer spaces have higher index),
- $\bigcup_{j \geq 0} V_j$ is dense in $L_2$,
- for each level $j$, scaling functions $\phi_{j,k}$ corresponding to vertices $k \in \mathcal{K}(j)$ exist so that $\{\phi_{j,k} | k \in \mathcal{K}(j)\}$ is a Riesz basis of $V_j$.

For our semi-regular meshes, the nested multiresolution vertex index sets $\mathcal{K}(j)$ are illustrated in Figure 3.1; the finest level $L$ vertex index set contains all vertices in the (domain) set and is denoted by $\mathcal{K}(L)$. The measure $d\Gamma$ describes differential areas on our polyhedral boundary $\Gamma$. The fact that the spaces are nested implies that each scaling function may be written as a linear combination of finer scaling functions using a refinement relation,

$$\phi_{j,k} = \sum_{l \in \mathcal{K}(j+1)} h_{j,k,l} \phi_{j+1,l}, \tag{3.1}$$

where $h_{j,k,l}$ are defined for $j \geq 0, k \in \mathcal{K}(j)$, and $l \in \mathcal{K}(j+1)$.

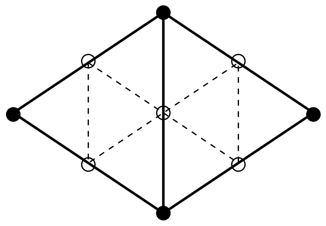


Figure 3.1: *Illustration of Wavelet Vertex Sets:* The image shows a simple two-level surface mesh patch on level $j+1$ (here $j=0$). The four *even* vertices (solid dots) belong to the base mesh and constitute $\mathcal{K}(j)$, whereas the *odd* vertices of $\mathcal{M}(j)$ all correspond to edge-splits ("midpoints") of parent edges. The union of the two sets is the set of all vertices on level $j+1$, namely $\mathcal{K}(j+1) = \mathcal{K}(j) \cup \mathcal{M}(j)$.

Each MRA is accompanied by a dual MRA consisting of spaces $\tilde{V}_j$, with dual scaling functions $\tilde{\phi}_{j,k}$ biorthogonal to the scaling functions

$$< \phi_{j,k}, \tilde{\phi}_{j,k'} >= \delta_{k,k'} \quad \text{for} \quad k, k' \in \mathcal{K}(j). \tag{3.2}$$

where $< f, g >= \int f g d\Gamma$ is the inner product on the surface $\Gamma$. The dual scaling functions also satisfy a refinement relation

$$\tilde{\phi}_{j,k} = \sum_{l \in \mathcal{K}(j+1)} \tilde{h}_{j,k,l} \tilde{\phi}_{j+1,l}. \tag{3.3}$$

In the biorthogonal setting, it is assumed that the primal and dual scaling functions are not equal (hence not orthogonal), and that the primal and dual spaces are also not equal (hence not semi-orthogonal). This is useful in practice for constructing fast wavelet transforms which have sparse filters and tunable wavelet properties.

Wavelets on level $j$ describe the difference between adjacent levels of the multiresolution representation by forming the basis of $W_j$ where $V_{j+1} = V_j \oplus W_j$. For our vertex bases, the wavelet functions are $\{\phi_{j,m} | j \geq 0, m \in \mathcal{M}(j)\}$, where $\mathcal{M}(j) \subset \mathcal{K}(j+1)$ is an index set corresponding to odd vertices in $\mathcal{K}(j+1)$ which are associated with "edge split" subdivision operations (see Figure 3.2). Ideally one would like the wavelets to form a Riesz basis for $L_2(\Gamma)$, and the set $\{\phi_{j,m} | m \in \mathcal{M}(j)\}$ to form a Riesz basis of $W_j$. Since $W_j \subset V_{j+1}$ the wavelets on level $j$ may be written in terms of scaling functions on level $j+1$

$$\psi_{j,m} = \sum_{l \in \mathcal{K}(j+1)} g_{j,m,l} \phi_{j+1,l} \quad \text{for} \quad m \in \mathcal{M}(j). \tag{3.4}$$

A similar relation occurs for the dual wavelet basis functions

$$\tilde{\psi}_{j,m} = \sum_{l \in \mathcal{K}(j+1)} \tilde{g}_{j,m,l} \tilde{\phi}_{j+1,l} \quad \text{for} \quad m \in \mathcal{M}(j). \tag{3.5}$$

The dual wavelet basis functions are biorthogonal to the wavelets and satisfy

$$< \psi_{j,m}, \tilde{\psi}_{j',m'} > \; = \; \delta_{m,m'} \delta_{j,j'}, \quad j, j' \geq 0, m \in \mathcal{M}(j), m' \in \mathcal{M}(j') \tag{3.6}$$

$$< \tilde{\psi}_{j,m}, \phi_{j,k} > \; = \; < \tilde{\phi}_{j,k}, \psi_{j,m} >= 0, \quad m \in \mathcal{M}(j), k \in \mathcal{K}(j) \tag{3.7}$$

so that for $f$ in $L_2$ we may write

$$f = \sum_{j,m} < \tilde{\psi}_{j,m}, f > \psi_{j,m} = \sum_{j,m} \gamma_{j,m} \psi_{j,m}. \tag{3.8}$$

It then follows that the scaling functions satisfy the relationship

$$\phi_{j+1,l} = \sum_{k \in \mathcal{K}(j)} \tilde{h}_{j,k,l} \phi_{j,k} + \sum_{m \in \mathcal{M}(j)} \tilde{g}_{j,m,l} \psi_{j,m}, \quad l \in \mathcal{K}(j+1). \tag{3.9}$$

The forward fast wavelet transform maps the scaling function coefficients of a function $f$,

$$\{\lambda_{j,k} =< f, \tilde{\phi}_{j,k} > | k \in \mathcal{K}(j)\}, \tag{3.10}$$

at the finest level of resolution $j = L$ to the wavelet coefficients

$$\{\gamma_{j,m} | 0 \leq j < L, m \in \mathcal{M}(j)\} \quad \text{and} \quad \{\lambda_{0,k} | k \in \mathcal{K}(0)\}. \tag{3.11}$$

This is recursively computed one level at a time, from fine to coarse scales, as

$$\lambda_{j,k} = \sum_{l\in\mathcal{K}(j)} \tilde{h}_{j,k,l}\lambda_{j+1,l} \tag{3.12}$$

$$\gamma_{j,m} = \sum_{l\in\mathcal{M}(j)} \tilde{g}_{j,m,l}\lambda_{j+1,l}. \tag{3.13}$$

The inverse transform undoes this process one level at a time, from coarse to fine scales, using

$$\lambda_{j+1,l} = \sum_{k\in\mathcal{K}(j)} h_{j,k,l}\lambda_{j,k} + \sum_{m\in\mathcal{M}(j)} g_{j,m,l}\gamma_{j,m}. \tag{3.14}$$

### 3.1.2 The Lifting Scheme and Fast Lifted Wavelet Tranform

The lifting scheme provides a straightforward means of constructing small filters $g, h, \tilde{g}, \tilde{h}$ corresponding to wavelets with desired properties. This is done by starting from a simple MRA and using lifting to construct a better performing set of filters.

As in [SS95a], the old quantities are denoted by "$^o$", so that the filters of the original MRA are $h^o_{j,k,l}$, $\tilde{h}^o_{j,k,l}$, $g^o_{j,k,l}$, and $\tilde{g}^o_{j,k,l}$. Then the *lifting scheme* relates the old and new filters by

$$h_{j,k,l} = h^o_{j,k,l} \tag{3.15}$$

$$\tilde{g}_{j,m,l} = \tilde{g}^o_{j,m,l} \tag{3.16}$$

$$g_{j,m,l} = g^o_{j,m,l} - \sum_{k\in\mathcal{K}(j)} s_{j,k,m}h_{j,k,l} \tag{3.17}$$

$$\tilde{h}_{j,k,l} = \tilde{h}^o_{j,k,l} - \sum_{m\in\mathcal{M}(j)} s_{j,k,m}\tilde{g}_{j,m,l} \tag{3.18}$$

and guarantees that the resulting filters are biorthogonal and invertible for any values of $\{s_{j,k,m}\}$. The scaling functions are unchanged but the dual scaling function and the primal and dual wavelets have all changed. The new basis functions are related by the refinement relation

$$\psi_{j,m} = \sum_{l\in\mathcal{K}(j+1)} g^o_{j,m,l}\phi_{j+1,l} - \sum_{k\in\mathcal{K}(j)} s_{j,k,m}\phi_{j,k} \tag{3.19}$$

$$\tilde{\phi}_{j,k} = \sum_{l\in\mathcal{K}(j+1)} \tilde{h}^o_{j,k,l}\tilde{\phi}_{j+1,l} + \sum_{m\in\mathcal{M}(j)} s_{j,k,m}\tilde{\psi}_{j,m} \tag{3.20}$$

It follows that the fast wavelet transform after lifting may be written as a sequence of two steps on each level

$$\gamma_{j,m} = \sum_{l\in\mathcal{K}(j+1)} \tilde{g}^o_{j,m,l}\lambda_{j+1,l} \tag{3.21}$$

$$\lambda_{j,k} \quad = \quad \sum_{l \in \mathcal{K}(j+1)} \tilde{h}^o_{j,k,l} \lambda_{j+1,l} + \sum_{m \in \mathcal{M}(j)} s_{j,k,m} \gamma_{j,m}. \tag{3.22}$$

Finally the inverse fast wavelet transform is

$$\lambda_{j+1,l} = \sum_{k \in \mathcal{K}(j)} h^o_{j,k,l} \left( \lambda_{j,k} - \sum_{m \in \mathcal{M}(j)} s_{j,k,m} \gamma_{j,m} \right) + \sum_{m \in \mathcal{M}(j)} g^o_{j,m,l} \gamma_{j,m}. \tag{3.23}$$

The fast lifted wavelet transform is then written factored into lifting steps as

```
Analysis
    For level = leafLevel to rootLevel
        AnalysisI (level)
        AnalysisII(level)
Synthesis
    For level = rootLevel to leafLevel
        SynthesisI (level)
        SynthesisII(level)
```

We shall consider specific filters in the following sections.

### 3.1.3 Interpolating Scaling Functions for Vertex Bases; Hierarchical Bases

The lifting scheme allows filters of interpolating transforms to be interpreted as a lifting of the Lazy wavelet transform, a trivial orthogonal transform which simply subsamples data

$$h^o_{j,k,l} = \tilde{h}^o_{j,k,l} = \delta_{k,l} \qquad \text{and} \qquad g^o_{j,m,l} = \tilde{g}^o_{j,m,l} = \delta_{m,l}. \tag{3.24}$$

By application of dual lifting to the Lazy wavelet, the filters associated with interpolating scaling functions with Dirac delta functions as their formal dual may be written as

$$h_{j,k,l} = \begin{cases} \delta_{k,l} & l \in \mathcal{K}(j) \\ \tilde{s}_{j,k,l} & l \in \mathcal{M}(j) \end{cases} \qquad \text{and} \qquad \tilde{g}_{j,m,l} = \begin{cases} -\tilde{s}_{j,l,m} & l \in \mathcal{K}(j) \\ \delta_{m,l} & l \in \mathcal{M}(j) \end{cases} \tag{3.25}$$

The coefficients $\{\tilde{s}_{j,l,m}\}_{l \in \mathcal{K}(j)}$ interpolate data from nearby even vertices of $\mathcal{K}(j)$ at the odd vertex $m \in \mathcal{M}(j)$. Let the required even vertices near $m$ be denoted by $\mathcal{K}_m$. The case of linear interpolation has the two vertex stencil $\mathcal{K}_m = \{v_1, v_2\}$ illustrated in Figure 3.2, with interpolation weights set to $\frac{1}{2}$. Other interpolants may also be used to increase the dual wavelet's number of vanishing moments, e.g., quadratic interpolation, or its degree of smoothness, e.g., Butterfly interpolation [DLG90]. In practice, smoother bases work better for smoother data, and lifted Butterfly wavelets appear to work well in this respect [SS95a]. The Butterfly interpolant's stencil is shown in Figure 3.3.

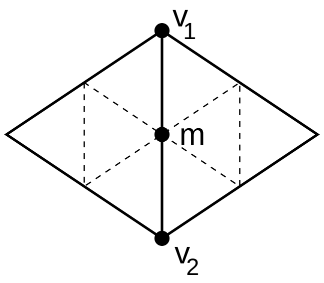


Figure 3.2: *The edge-split stencil for linear interpolation* is used to interpolate at an odd vertex $m \in \mathcal{M}(j)$ using the (weighted) average of values at its two *aunt* vertices $\mathcal{K}_m = \{v_1, v_2\} \subset \mathcal{K}(j)$.

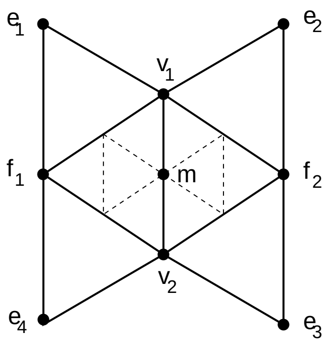


Figure 3.3: *The edge-split stencil for Butterfly subdivision* is used to interpolate at an odd vertex $m \in \mathcal{M}(j)$ using the (weighted) average of values in $\mathcal{K}_m = \{v_1, v_2, f_1, f_2, e_1, e_2, e_3, e_4\} \subset \mathcal{K}(j)$. As with linear interpolation, the *aunts* of vertex $m$ will be defined as $v_1$ and $v_2$.

The interpolating transform analysis and synthesis stages are then

$$\begin{aligned}
&\texttt{AnalysisI}(\texttt{j}) : \\
&\forall k \in \mathcal{K}(j) : \quad \lambda_{j,k} := \lambda_{j+1,k} && (3.26)\\
&\forall m \in \mathcal{M}(j) : \quad \gamma_{j,m} := \lambda_{j+1,m} - \sum_{k \in \mathcal{K}_m} \tilde{s}_{j,k,m} \lambda_{j,k} && (3.27)\\
&\texttt{SynthesisII}(\texttt{j}) : \\
&\forall k \in \mathcal{K}(j) : \quad \lambda_{j+1,k} := \lambda_{j,k} && (3.28)\\
&\forall m \in \mathcal{M}(j) : \quad \lambda_{j+1,m} := \gamma_{j,m} + \sum_{k \in \mathcal{K}_m} \tilde{s}_{j,k,m} \lambda_{j,k}. && (3.29)
\end{aligned}$$

Using these transform stages (without `AnalysisII` and `SynthesisI` from the following section) lead to interpolating transforms. While we are not interested in using these unlifted transforms for compressing Green's functions, the interpolating primal basis functions $\{\phi_{j,k}\}$, e.g., hierarchical basis functions [Yse86] for linear interpolation, are used for describing hierarchical constraints in §3.4. In a slight abuse of terminology, we will sometimes refer to interpolating basis functions as hierarchical basis functions.

### 3.1.4 Lifted Vertex Bases

The previous section introduced filters for interpolating scaling functions which we use to define variants of hierarchical basis functions for the definition of hierarchical constraints in §3.4. Unfortunately, for efficiently representing the GFs (columns of GF matrix $\Xi$) the wavelets are too simple; the primal wavelets do not have any vanishing moments and actually correspond to primal scaling functions. Repeated lifting can improve the situation, and it is used here to ensure that primal wavelet has one vanishing moment, and will have better convergence properties and behavior when thresholding.

Following [SS95a] the lifting coefficients of the lifted wavelets (3.19) are determined by requiring that the new primal wavelets have a vanishing moment. Given the lifted wavelet definition

$$\psi_{j,m} = \phi_{j+1,m} - s_{j,v_1,m}\phi_{j,v_1} - s_{j,v_2,m}\phi_{j,v_2} \tag{3.30}$$

the weights $s_{j,v_*,m}$ are chosen by requiring

$$\int_\Gamma \psi_{j,m} d\Gamma = 0 = I_{j+1,m} - s_{j,v_1,m}I_{j,v_1} - s_{j,v_2,m}I_{j,v_2} \tag{3.31}$$

where

$$I_{j,k} = \int_\Gamma \phi_{j,k} d\Gamma \tag{3.32}$$

then simply choosing

$$s_{j,k,m} = \frac{I_{j+1,m}}{2I_{j,k}} \tag{3.33}$$

for each coefficient. The scaling function integrals $I_{j,k}$ are computed recursively in practice since the refinement relation (3.1) implies that

$$I_{j,k} = \sum_{l \in \mathcal{K}(j+1)} h_{j,k,l} I_{j+1,l} \tag{3.34}$$

and the values of $\{I_{L,l}\}_{l\in\mathcal{K}(L)}$ are trivial to compute. Here $h_{j,k,l}$ is given in terms of $\{\tilde{s}_{j,k,m}\}$ by (3.25).

Once lifting coefficients $\{s_{j,k,m}\}$ are determined the primal wavelet lifting stages of the fast wavelet transform become

```
AnalysisII(j) :
    ∀m ∈ M(j) :   { $\lambda_{j,v_1} += s_{j,v_1,m}\gamma_{j,m}$
                  { $\lambda_{j,v_2} += s_{j,v_2,m}\gamma_{j,m}$          (3.35)
```

$$\texttt{SynthesisI(j)}:$$

$$\forall m \in \mathcal{M}(j): \quad \begin{cases} \lambda_{j,v_1} -= s_{j,v_1,m}\gamma_{j,m} \\ \lambda_{j,v_2} -= s_{j,v_2,m}\gamma_{j,m} \end{cases} \tag{3.36}$$

### 3.1.5 Adapting Transforms To Surface Domains

It remains to describe how the vertex bases' interpolating filter coefficients $\{\tilde{s}_{j,k,m}\}$ and lifting coefficients $\{s_{j,k,m}\}$ are determined near the boundaries of the surface domains $\{D_i\}_{i=1}^d$. In such cases it is possible that, respectively, an odd vertex $m$ does not have both its $v_1, v_2$ aunts (see Figures 3.2 and 3.3) contained in the same domain, e.g., to interpolate from (`AnalysisI or SynthesisII`), or to provide a lifting update to (`AnalysisII or SynthesisI`). These problems can be overcome; our use of multiresolution analyses adapted to surface domains resulted in no complications.

Modified lifting filters at domain boundaries were constructed as follows (see also [SS95a]). In cases for which only one aunt, $v_1$, belongs to the domain, this aunt was used for lifting in an analogous manner: the lifted wavelet was defined as

$$\psi_{j,m} = \phi_{j+1,m} - s_{j,v_1,m}\phi_{j,v_1} \tag{3.37}$$

and the weight $s_{j,v_1,m}$ is chosen by requiring

$$\int_\Gamma \psi_{j,m} d\Gamma = 0 = I_{j+1,m} - s_{j,v_1,m} I_{j,v_1} \tag{3.38}$$

so that

$$s_{j,v_1,m} = \frac{I_{j+1,m}}{I_{j,v_1}}. \tag{3.39}$$

In the rare case for which both aunts are outside the domain the wavelet is currently left unlifted.

For linear interpolants, boundary stencils lacking both aunts $(v_1, v_2)$ were approximated using a constant estimate: vertices with only one aunt used the aunt's value as the estimate, and vertices with no aunts in the domain used the nearest neighbour (on the aunt's level) for the estimate. For butterfly interpolants, boundary cases occur more often due to the larger butterfly stencil. Cases for which the butterfly stencil is not contained in the domain are handled by resorting to linear interpolation, and the aforementioned constant approximations when linear is not possible. No interesting consequences of these boundary approximations were observed.

## 3.2 Wavelet Green's Functions

Displacement and traction fields associated with deformations arising from localized loads exhibit significant multiscale properties such as being very smooth in large areas away from the loading point (and other constraints) and achieving a local maxima about the point. For this reason, free space GFs (or fundamental solutions) of unbounded elastic media and our boundary GFs of 3D objects are efficiently represented by multiresolution bases. And just as this property of free space fundamental solutions allows for effective wavelet discretization methods for a wide range of integral equations [BCR91a], it will also allow us to construct sparse wavelet representations of discrete elastostatic GF integral operators obtained from numerical solutions of constrained geometric models as well as measurements of real world objects.

One could treat the GF matrix $\Xi$ as a generic operator to be efficiently represented for full matrix-vector multiplication, but this is inappropriate for our application. In the CMA constraint solver, column-based GF operations such as weighted summations of selected GF columns dominate the fast solution process, and GF element extraction must be a relatively cheap operation in order for capacitance matrices to be obtained cheaply at run-time. Because of this, during precomputation we represent individual GF columns of the large GF matrix, $\Xi$, in the wavelet basis, but we do not transform across GF rows. Requirements affecting the particular choice of wavelet scheme are discussed in §3.2.4, and row-based multiresolution constraints are addressed in §3.4

### 3.2.1 Domain Structure of the Green's Function Matrix

Each GF column vector describes nodal traction and displacement distributions on different domains of the boundary, both of which have different smoothness characteristics. Interfaces between domains are therefore associated with discontinuities and the adjacent traction and displacement function values can have very different magnitudes and behaviors. For these reasons, multiresolution analysis of GFs is performed separately on each domain to achieve best results. From a practical standpoint, this also aids in simulating individual domains of the model independently (§2.2.6).

Domains are constructed by first partitioning nodes into $\Lambda_u^0$ and $\Lambda_p^0$ lists for which the GFs describe tractions and displacements, respectively. These lists are again split into disjoint subdomains if the particular wavelet transform employed can not exploit coherence between these nodes. Let the boundary nodes be partitioned into $d$ domains

$$D = \{D_1, \ldots, D_d\} \quad \text{with} \quad \bigcap_{i=1}^{d} D_i = \emptyset \tag{3.40}$$

where $D_i$ is a list of nodes in the natural coarse to fine resolution order of that domain's wavelet transform.

The $d$ domains introduce a natural row and column ordering for the GF matrix $\Xi$

which results in a clear block structure

$$\Xi = \sum_{i,j=1}^{d} \mathsf{E}_{D_i} \Xi_{D_i D_j} \mathsf{E}_{D_j}^{\mathsf{T}} \tag{3.41}$$

$$= [\mathsf{E}_{D_1} \mathsf{E}_{D_2} \dots \mathsf{E}_{D_d}] \begin{bmatrix} \Xi_{D_1D_1} & \Xi_{D_1D_2} & \cdots & \Xi_{D_1D_d} \\ \Xi_{D_2D_1} & \Xi_{D_2D_2} & & \vdots \\ \vdots & & \ddots & \\ \Xi_{D_dD_1} & \cdots & & \Xi_{D_dD_d} \end{bmatrix} \begin{bmatrix} \mathsf{E}_{D_1}^{\mathsf{T}} \\ \mathsf{E}_{D_2}^{\mathsf{T}} \\ \vdots \\ \mathsf{E}_{D_d}^{\mathsf{T}} \end{bmatrix} \tag{3.42}$$

where the $(i, j)$ GF block

$$\Xi_{D_iD_j} = \mathsf{E}_{D_i}^{\mathsf{T}} \Xi \mathsf{E}_{D_j} \tag{3.43}$$

maps data from domain $D_i$ to $D_j$ as illustrated in Figure 3.4.

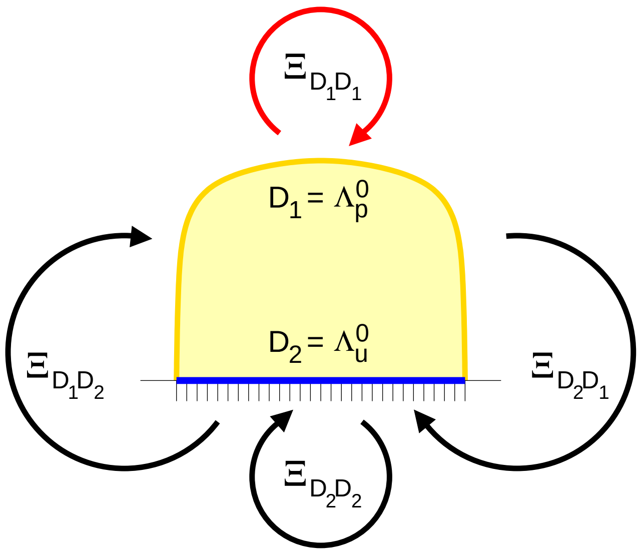


Figure 3.4: *Illustration of correspondence between boundary domain influences and domain block structure of the GF matrix* $\Xi$*:* The influences between two boundary domains are illustrated here by arrows; each arrow represents the role of a GF block, $\Xi_{D_iD_j}$, in the flow of information from specified BVs on domain $D_j$ to unspecified BVs on domain $D_i$. The *self-effect* of the exposed contactable surface (red arrow at top) is of primary practical interest for deformation visualization. Each column of $\Xi_{D_1D_1}$ represents a displacement field on $D_1$ which decribes the effect of a force applied over some portion of $D_1$; this displacement field is efficiently represented using wavelets in §3.2.

### 3.2.2 Wavelet Transforms on Surface Domains

Consider the forward and inverse fast wavelet transform (FWT) pair, $(\mathsf{W}, \mathsf{W}^{-1})$, itself composed of FWT pairs

$$\mathsf{W} = \sum_{i=1}^{d} \mathsf{E}_{D_i} \mathsf{W}_i \mathsf{E}_{D_i}^{\mathsf{T}} = \mathsf{E}_D \left(\mathrm{diag}_i(\mathsf{W}_i)\right) \mathsf{E}_D^{\mathsf{T}} \tag{3.44}$$

$$\mathsf{W}^{-1} = \sum_{i=1}^{d} \mathsf{E}_{D_i} \mathsf{W}_i^{-1} \mathsf{E}_{D_i}^{\mathsf{T}} = \mathsf{E}_D \left(\mathrm{diag}_i(\mathsf{W}_i^{-1})\right) \mathsf{E}_D^{\mathsf{T}} \tag{3.45}$$

with the $i^{th}$ pair $(\mathsf{W}_i, \mathsf{W}_i^{-1})$ is defined on domain $D_i$.

### 3.2.3 Wavelet Green's Functions

Putting things together, the wavelet transform of the GF matrix is then

$$\mathsf{W}\Xi = [(\mathsf{W}\xi_1)\,(\mathsf{W}\xi_2)\cdots(\mathsf{W}\xi_n)] = \sum_{i,j=1}^{d} \mathsf{E}_{D_i} \left(\mathsf{W}_i \Xi_{D_i D_j}\right) \mathsf{E}_{D_j}^{\mathsf{T}} \tag{3.46}$$

or with a shorthand "tilde" notation for transformed quantities,

$$\tilde{\Xi} = \left[\tilde{\xi}_1\,\tilde{\xi}_2\cdots\tilde{\xi}_n\right] = \sum_{i,j=1}^{d} \mathsf{E}_{D_i} \left(\tilde{\Xi}_{D_i D_j}\right) \mathsf{E}_{D_j}^{\mathsf{T}} \tag{3.47}$$

The individual block component of the $j^{th}$ *wavelet GF* $\tilde{\xi}_j = \tilde{\Xi}_{:j}$ corresponding to vertex $i$ on level $l$ of domain $d$ will be denoted with rounded bracket subscripts as

$$\left(\tilde{\xi}_j\right)_{(l,i;d)} = \tilde{\Xi}_{(l,i;d)j}. \tag{3.48}$$

This notation is complicated but no more than necessary, since it corresponds directly to the multiresolution data structure used for implementation.

### 3.2.4 Choice of MRA and Fast Wavelet Transforms

By design, various custom multiresolution analyses and fast wavelet transforms can be plugged into the framework developed here. However, there are several practical requirements: interactive inverse transform speeds (for fast summation), good GF compression (even for small models given that precomputation costs quickly increase), support for level of detail computations, ease of transform definition on user-specified surface domains $D_i$, as well as support for data from a wide range of discretizations.

An interactive fast summation clearly dictates the need for a fast inverse wavelet transform, while motivations for symmetric forward and inverse wavelet transform costs are runtime change of constraint bases (such as with pressure masks) and reference constraint domain $D$ modification, as well as reduction of precomputation time and flexibility when initially transforming GFs into the wavelet basis. We note that, at the cost of some runtime flexibility, it may be possible to employ a more sophisticated (and slower) forward wavelet transform during precomputation provided it yields significantly greater compression [KSS00] and thus faster runtime BVP solution speeds. Such issues are also important considerations for storage and transmission (the central motivation in [KSS00]), however we specifically mention that we are separating this from the current discussion on simulation methods.

Given the many constraints, we use biorthogonal lifted fast wavelet transforms based on second generation wavelets derived from the lifting scheme of Sweldens et al. [Swe98, DS96, SS95a], which were summarized in §3.1. We shall initially consider only FWTs based on a single lifting of hierarchical basis functions, i.e., the dual wavelet has one vanishing moment, as well as the smoother lifted butterfly wavelets. Intuitively, one might expect the butterfly scheme to perform better at characterizing smooth portions of the GF, while the linear wavelet may be better at describing their spike-like behavior. Lifted linear wavelets have cheaper transforms than lifted butterfly, unless the compression obtained by butterfly is significantly greater than linear, which does not seem to be the case (§7). The compact edge-split stencil is also effective for geometric models of very modest geometric complexity. Finally, a drawback of linear wavelets is that for very high compression ratios, they can introduce polygonal artifacts when adding GF displacements to a very flat surface, unlike the smoother butterfly reconstructions.

### 3.2.5 Tensor Wavelet Thresholding

Each 3-by-3 block of the GF matrix describes a tensor influence between two nodes. The wavelet transform of a GF (whose row elements are $3 \times 3$ matrix blocks) is mathematically equivalent to 9 scalar transforms, one for each tensor component. However, in order to reduce runtime sparse matrix overhead, it is desireable to evaluate all transforms at the block level. For this reason, we use a thresholding operation which either accepts or rejects an entire block.

Our oracle for wavelet GF thresholding compares the Frobenius norm of each block wavelet coefficient[1] to a domain and level specific thresholding tolerance, and sets the co-

[1]The Frobenius norm of a real-valued 3-by-3 matrix $a$ is

$$\|a\|_F = \sqrt{\sum_{ij} a_{ij}^2}.$$

efficient to zero if it is smaller. Thresholding of the $j^{th}$ wavelet GF, $\tilde{\xi}_j$, on a domain $d$ is performed for the $i^{th}$ coefficient iff $i \in D_d$, $i \in \mathcal{M}(l)$ and

$$\|\tilde{\Xi}_{ij}\|_F < \varepsilon_l \|\mathsf{E}_{D_d}^{\mathsf{T}} \xi_j\|_{\infty F} \tag{3.49}$$

where

$$\|\mathsf{E}_{D_d}^{\mathsf{T}} \xi_j\|_{\infty F} \equiv \max_{i \in D_d} \|\Xi_{ij}\|_F \tag{3.50}$$

is a weighted measure of GF amplitude on domain $d$, and $\varepsilon_l$ is a level dependent relative threshold parameter decreased on coarser levels (smaller $l$) as

$$\varepsilon_l = 2^{l-L}\varepsilon, \qquad l = 1, ..., L, \tag{3.51}$$

with $\varepsilon$ the user-specified threshold parameter. We usually do not threshold base level ($l = 0$) coefficients even when this introduces acceptable errors because the lack of response, e.g., pixel motion, in these regions can be perceptually bothersome.

For our models we have observed stable reconstruction of thresholded data, e.g.,

$$\|\mathsf{E}_{D_d}^{\mathsf{T}} \left( \xi_j - \mathsf{W}^{-1}\tilde{\xi}_j \right) \|_{\infty F} < C\varepsilon \|\mathsf{E}_{D_d}^{\mathsf{T}} \xi_j\|_{\infty F} \tag{3.52}$$

typically for some constant $C$ near 1. Examples are shown in §7. Although there are no guarantees that wavelet bases constructed on any particular model will form an unconditional basis, and so the thresholding operation will lead to stable reconstructions, none of our numerical experiments with discrete GFs have suggested anything to the contrary. Similar experiences were reported by the pioneers of the lifting scheme in [SS95a] for wavelets on the sphere. Some formal conditions on the stability of multiscale transformations are proven in [Dah96]. Numerous experimental results showing the relationship between error and thresholding tolerance can be found in §7.4.

### 3.2.6 Multiresolution Mesh Issues

We use multiresolution triangle meshes with subdivision connectivity to conveniently define wavelets and MR constraints (§3.4) as well as provide detailed graphical and haptic rendering (§3.7). Many of our meshes have been modeled using Loop subdivision [Loo87] which trivializes the generation of multiresolution meshes, and some examples are shown in Figures 1.2 (p. 5) and 7.18 (p. 117). For general meshes which have not been modeled as subdivision surfaces, several successful reparameterization approaches for producing multiresolution meshes have appeared in the literature [EDD$^+$95, KL96, LSS$^+$98, GVSS00, LMH00] and there are commercially available packages, e.g., Raindrop Geomagic [Rai] and Paraform [Par]. For our purposes, we have implemented reparameterization algorithms based on normal meshes [GVSS00, LMH00], and have used the related displaced subdivision surface approach [LMH00] for rendering detailed deforming models and generating displacement map. Examples of models we have reparameterized are the rabbit model (Figures 1.1 (p. 2) and 3.9 (p. 64)) (original mesh courtesy of Cyberware [Cyb]), the dragon model (Figure 3.5, p. 56; original mesh courtesy of the Stanford Computer Graphics Laboratory), and the reality-based tiger model (Figure 6.4, p. 6.4). For the dragon model[2] some undesireable parameterization artifacts are present near sharp features such as the horns, however this could be avoided with more care.

For good wavelet compression results, it is desireable to have many subdivision levels for a given model. This also aids in reducing the size of the dense base level GF data, if it is left unthresholded. In cases where the coarsest resolution of the mesh is still large, reparamerization should be considered but it is still possible to consider more exotic lifted wavelets on arbitrary point sets. To maximize the number of levels for small models, e.g., for the rabbit model, we resorted to manual fitting of coarse base level parameterizations, although more sophisticated approaches are available [EDD$^+$95, KL96, LSS$^+$98, GVSS00].

Lastly we mention that adaptive meshing of geometry must be used with care since it can limit the scales at which displacement and traction fields may resolve surface deformations and constraints.

### 3.2.7 Storage and Transmission of Green's Functions

Wavelets provide bases for sparsely representing GFs, but further compression is possible for storage formats. Given the potentially vast amount of information precomputed, an efficient file format is an important practical concern for data storage and transmission. In this respect, efficient wavelet quantization and coding schemes [DJL92, Sha93, SP96] have already been extended to dramatically reduce the file sizes of surface functions compressed using the lifting scheme [KL97]. Similar approaches could be applied to GF data.

## 3.3 CMA with Fast Summation of Wavelet GFs

The CMA is only slightly more involved when the GFs are represented in wavelet bases. The chief benefit is the performance improvement obtained by using the FWT for fast summation of GF and body force responses. While MR solution reconstruction for rendering was possible previously by exploiting random access selective computation, it will be more efficient using the sparse wavelet GFs and the following algorithm.

[2]The dragon model also required special care due to numerous holes present in the original mesh. Although there are techniques to fill these [GW01], a significantly worse problem was nonphysical interior cavity meshing on the bottom of the object, at what probably were injection moulding inlets; these were remodeled by hand before applying the reparameterization process.

### 3.3.1 Motivation

In addition to reducing memory usage, it is well known that by sparsely representing our GF columns in a wavelet basis we can use the FWT for fast matrix multiplication [BCR91a]. For example, consider the central task of computing a weighted summation of $s$ GFs

$$\sum_{j \in \mathsf{S}} \xi_j \bar{\mathsf{v}}_j, \tag{3.53}$$

involving $sn$ $3 \times 3$ matrix-vector multiply-accumulate operations. Quick evaluation of such expressions is crucial for fast BVP solution (c.f. (2.15)) and graphical rendering of deformations, and it is required at least once by the CMA solver. Unfortunately, as $s$ increases this operation quickly becomes more and more costly and as $s \to n$ eventually involves $O(n^2)$ operations. However, by using a fast wavelet transform (FWT) it is possible to perform such sums more efficiently in a space in which the GF columns may be approximately represented with sparse representations.

The weighted GF summation can be rewritten by premultiplying (3.53) with the identity operator $\mathsf{W}^{-1}\mathsf{W}$:

$$\sum_{j \in \mathsf{S}} \xi_j \bar{\mathsf{v}}_j = \mathsf{W}^{-1} \sum_{j \in \mathsf{S}} \tilde{\xi}_j \bar{\mathsf{v}}_j. \tag{3.54}$$

By precomputing sparse thresholded approximations of the wavelet transformed GFs, $\tilde{\Xi}$, a fast summation will result in (3.54) provided that the advantage of sparsely representing $\Xi$, more than compensates for the extra cost of applying $\mathsf{W}^{-1}$ to the vector data. This occurs in practice, due to the FWT's speed and excellent decorrelation properties for GF data.

### 3.3.2 Formulae

The necessary formulae result from substituting

$$\Xi = \mathsf{W}^{-1}\mathsf{W}\Xi \tag{3.55}$$

into the CMA formulae (2.35-2.38), and using the GF expression (3.47). The result may be written as

$$\begin{aligned}
\mathsf{v} &= \mathsf{v}^{(0)} + \left(\mathsf{E} + \mathsf{W}^{-1}(\tilde{\Xi}\mathsf{E})\right) \mathsf{C}^{-1} \left(\mathsf{E}^\mathsf{T}\mathsf{v}^{(0)}\right) && (3.56)\\
\mathsf{C} &= -\left(\mathsf{E}^\mathsf{T}\mathsf{W}^{-1}\right)(\tilde{\Xi}\mathsf{E}) && (3.57)\\
\mathsf{v}^{(0)} &= \mathsf{W}^{-1}\left[\tilde{\Xi}\left(\mathsf{I} - \mathsf{E}\mathsf{E}^\mathsf{T}\right)\bar{\mathsf{v}} + (\mathsf{W}\mathsf{A}_0^{-1}\mathsf{B})\beta\right] - \mathsf{E}\mathsf{E}^\mathsf{T}\bar{\mathsf{v}} && (3.58)\\
\mathsf{E}^\mathsf{T}\mathsf{v}^{(0)} &= \left(\mathsf{E}^\mathsf{T}\mathsf{W}^{-1}\right)\left[\tilde{\Xi}\left(\mathsf{I} - \mathsf{E}\mathsf{E}^\mathsf{T}\right)\bar{\mathsf{v}} + (\mathsf{W}\mathsf{A}_0^{-1}\mathsf{B})\beta\right] - \mathsf{E}^\mathsf{T}\bar{\mathsf{v}} && (3.59)
\end{aligned}$$

where we have taken the liberty of sparsely representing the parameterized body force contributions in the wavelet basis. With these formulae, it is possible to evaluate the solution $\mathsf{v}$ using only one inverse FWT evaluation and some partial reconstructions $\mathsf{E}^\mathsf{T}\mathsf{W}^{-1}$.

### 3.3.3 Selective Wavelet Reconstruction Operation, $(\mathsf{E}^\mathsf{T}\mathsf{W}^{-1})$

The operator $(\mathsf{E}^\mathsf{T}\mathsf{W}^{-1})$ represents the reconstruction of a wavelet transformed function at the updated nodes $\mathsf{S}$. This is required in at most two places: (1) capacitance matrix element extraction from $\tilde{\Xi}$; (2) evaluation of $(\mathsf{E}^\mathsf{T}\mathsf{v}^{(0)})$ in cases when the first term of $\mathsf{v}^{(0)}$ (in square brackets) is nonzero. It follows from the tree structure of the wavelet transform that these extraction operations can be evaluated efficiently with worst-case per-element cost proportional to the logarithm of the domain size. In practice, several optimizations related to spatial and temporal data structure coherence can significantly reduce this cost. For example, portions of $\mathsf{C}$ are usually cached and so extraction costs are amortized over time, with typical very few entries required per new BVP. Also, spatial clustering of updated nodes leads to the expected cost of extracting several clustered elements being not much more than the cost of extracting one. Furthermore, spatial clustering in the presence of temporal coherence allows us to exploit coherence in a sparse GF wavelet reconstruction tree, so that nodes which are topologically adjacent in the mesh can expect to have elements reconstructed at very small costs. For these reasons, it is possible to extract capacitance matrix entries at a fraction of the cost of LU factorization. Performance results for block extraction operations are given in §7.4.4. The logarithmic cost penalty introduced by wavelet representations is further reduced in the presence of hierarchical constraints, and a hierarchical variant of the fast summation CMA is discussed in §3.6.

### 3.3.4 Algorithm

An efficient algorithm for computing the entire solution vector $\mathsf{v}$ is possible by carefully evaluating subexpressions as follows:

1. Given constraints, $\bar{\mathsf{v}}$, and list nodes to be updated, $\mathsf{S}$.

2. Obtain $\mathsf{C}^{-1}$ (or factorization) for this BVP type either from the cache (*Cost*: Free), using updating (§2.3) (*Cost:* $\mathcal{O}(s^2 s_\Delta)$ flops), or from scratch (*Cost:* $2s^3/3$ flops).

3. If nonzero, evaluate the sparse summation

$$\tilde{\mathsf{g}}_1 = \left[\tilde{\Xi}\left(\mathsf{I} - \mathsf{E}\mathsf{E}^\mathsf{T}\right)\bar{\mathsf{v}} + (\mathsf{W}\mathsf{A}_0^{-1}\mathsf{B})\beta\right]. \tag{3.60}$$

   (*Cost:* $18\bar{s}\tilde{n}$ flops from first term where where $\tilde{n}$ is the average number of nonzero 3-by-3 blocks per wavelet GF being summed (in practice $\tilde{n} \ll n$), and $\bar{s}$ is the number of nonupdated nonzero constraints. Second body force term is similar but ignored due to ambiguity. Cost can be reduced by exploiting temporal coherence, e.g., see (2.21).).

4. Compute the block $s$-vector

$$\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} = \left(\mathsf{E}^{\mathsf{T}}\mathsf{W}^{-1}\right)\tilde{\mathsf{g}}_1 - \mathsf{E}^{\mathsf{T}}\bar{\mathsf{v}}. \tag{3.61}$$

(*Cost:* Selective reconstruction cost (if nontrivial $\mathsf{g}_1$) $3sR_{\mathsf{S}}$ where $R_{\mathsf{S}}$ is the effective cost of reconstructing a scalar given $\mathsf{S}$ (discussed in §3.3.3; expected cost is $R_{\mathsf{S}} = \mathcal{O}(1)$, worst case cost is $R_{\mathsf{S}} = \mathcal{O}(\log n)$), plus $3s$ flops for addition).

5. Evaluate the block $s$-vector

$$\mathsf{g}_2 = \mathsf{C}^{-1}(\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)}) \tag{3.62}$$

(*Cost:* $18s^2$ flops).

6. Perform the sparse summation

$$\tilde{\mathsf{g}}_1 \mathrel{+}= (\tilde{\Xi}\mathsf{E})\mathsf{g}_2 \tag{3.63}$$

(*Cost:* $18s\tilde{n}$ flops).

7. Perform inverse FWT (can be performed in place on block 3-vector data)

$$\mathsf{v} = \mathsf{W}^{-1}\tilde{\mathsf{g}}_1 \tag{3.64}$$

(*Cost:* $3C_{\mathsf{IFWT}}n$ flops; where $C_{\mathsf{IFWT}}$ is approximately 4 for lifted linear wavelets.).

8. Correct updated values to obtain the final solution,

$$\mathsf{v} \mathrel{+}= \mathsf{E}(\mathsf{g}_2 - \mathsf{E}^{\mathsf{T}}\bar{\mathsf{v}}) \tag{3.65}$$

(*Cost:* $6s$ flops).

### 3.3.5 Cost Analysis

The total cost of evaluating the solution is

$$Cost = 3C_{\mathsf{IFWT}}n + 18(s+\bar{s})\tilde{n} + 18s^2 + 3s(\mathsf{R}_{\mathsf{S}} + 3) \quad \text{flops} \tag{3.66}$$

where the notable improvement introduced by fast summation is the replacement of the $18sn$ dense summation cost with that of the sparse summation and inverse FWT. This excludes the cost of capacitance matrix inverse construction (or factorization or updating), if updating is performed, since this is experienced only once per BVP type and amortized over frames.

Two interesting special cases are when nonzero constraints are either all updated ($\bar{s}=0$) or when no constraints are updated ($s=0$). In the case where all nonzero constraints are updated ($\bar{s}=0$), and therefore step 3 has zero $\mathsf{g}_1$, the total cost of the calculation is

$$Cost = 3C_{\mathsf{IFWT}}n + 18s\tilde{n} + 18s^2 + 3s(\mathsf{R}_{\mathsf{S}} + 3) \quad \text{flops.} \tag{3.67}$$

Cases in which updated nodes have zero constraints are slightly cheaper. When no constraints are updated ($s=0$) only GF fast summation is involved, and the cost is

$$Cost = 3C_{\mathsf{IFWT}}n + 18\bar{s}\tilde{n} \quad \text{flops.} \tag{3.68}$$

In practice we have reduced these costs by only reconstructing the solution on subdomains (reduces FWT cost and summation cost) where it is required, e.g., for graphical rendering. It clearly follows that it is possible to reconstruct the solution at coarser resolutions for multiple LOD rendering, i.e., by only evaluating $\mathrm{g}_1$ and the IFWT in step 7 for coarse resolutions, and this issue is discussed futher in §3.7.

We found this algorithm to be very effective for interactive applications, and especially for force feedback simulation with point-like contacts (§4.2; small $\bar{s}$ and $s = 0$). Timings and typical flop counts are provided in the Results chapter (§7). For large models with many updated constraints, the $s\tilde{n}$ and $s^2$ contributions, in addition to the capacitance matrix inversion, can become costly. This issue is addressed in the following section by introducing multiresolution constraints which can favourably reduce the effective size of $s$.

## 3.4 Hierarchical Constraints

The MR GF representations make it feasible to store and simulate geometrically complex elastic models by eliminating the dominant bottlenecks associated with dense GF matrices. However, finer discretizations can introduce complications for real time simulations which impose numerous constraints on these same fine scales: (1) even sparse fast summation will eventually become too costly as more GF columns contribute to the sum, and (2) updating numerous constraints with the CMA incurs costly capacitance matrix inversion costs.

We provide a practical solution to this problem which can also optionally reduce precomputation costs. Our approach is to reduce the number of constraints by imposing constraints at a coarser resolution than the geometric model (see Figure 3.5). This eliminates the aforementioned bottlenecks without sacrificing model complexity. Combined with wavelet GFs which enable true multiresolution BVP simulation and solution output, multiresolution constraints provide the BVP's complementary multiresolution input control. Such an approach is well-suited to the CMA which effectively works by updating constraints defined over finite areas; in the continuous limit, as $n \to \infty$ and scaling function measures go to zero, the area affected by the uniresolution finite-rank-updating CMA also goes to zero and the CMA would have no effect.

The multiresolution constraints are described by nested spaces with node interpolating basis functions defined on each domain. Using interpolating scaling functions allows hierarchical constraints to coexist with nodal constraint descriptions, which is useful for defining the hierarchical version of the CMA (in §3.6). For our piecewise linear function

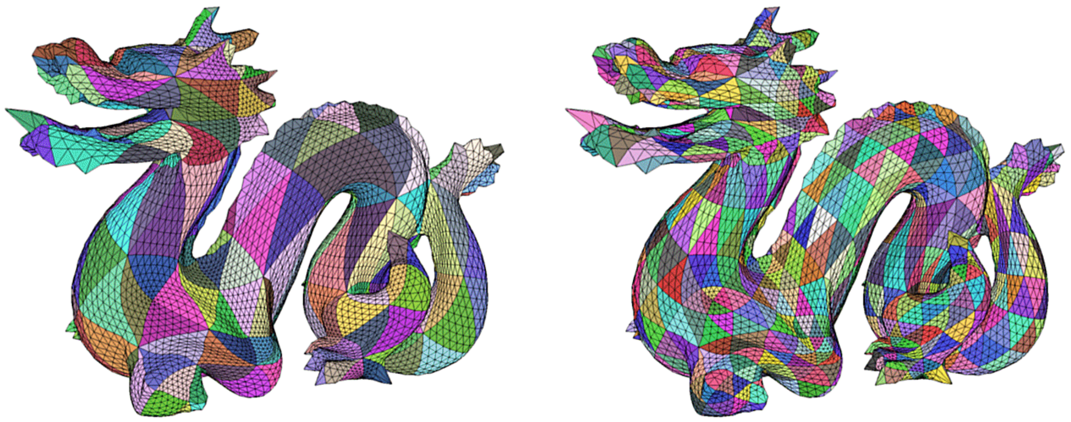


Figure 3.5: *Multiresolution Constraint Parameterizations:* Two dragon meshes (L=3) with coarser constraint parameterizations indicated for different resolutions of the Green's function hierarchy; (left) constraints on level 0, and (right) on level 1. In this way, interactive traction constraints can be applied on the coarse scale while deformations are rendered using fine scale displacement fields. (Reparameterized dragon model generated from mesh courtesy of Stanford Computer Graphics Laboratory.)

spaces these correspond to "hierarchical basis functions" [Yse86] and the interpolation filters are already available from the unlifted portion of the linear FWT used for the MR GFs.

Let the scalar hierarchical basis function

$$\phi_{[l,k;d]} = \phi_{[l,k;d]}(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{3.69}$$

correspond to vertex index $k$ belonging to level $l$ and domain $D_d$. Here the square subscript bracket is used to indicate an hierarchical basis function; recall (equation 3.48) that rounded subscript brackets are used to refer to row components of wavelet transformed vectors or matrix columns. In this notation, the traditional "hat functions" on the finest scale are

$$\phi_k(\mathbf{x}) = \phi_{[L,k;d]}(\mathbf{x}), \qquad k \in D_d. \tag{3.70}$$

In bracket notation, the refinement relation satisfied by these interpolating scaling functions is

$$\phi_{[l,k;d]} = \sum_{j \in \mathcal{K}(l+1)} h_{[l,k,j;d]}\, \phi_{[l+1,j;d]}, \tag{3.71}$$

where $h$ is defined by (3.25) from §3.1.3. As a result, the surface hierarchical basis functions are unit normalized

$$\phi_{[l,i;d]}(\mathbf{x}_{(l,j;d)}) = \delta_{ij} \tag{3.72}$$

where $\delta_{ij}$ is the Kronecker delta function. The refinement relation for hierarchical basis functions also means that hierarchical constraint boundary values are defined on finer constraint scales by interpolating subdivision (with the $h$ refinement filter (3.25)) as

$$\bar{\mathsf{v}}_{[l,:;:]} = \mathsf{H}_l^{\mathsf{T}} \bar{\mathsf{v}}_{[l+1,:;:]}, \tag{3.73}$$

where we have used a brief operator notation, or simply

$$\bar{\mathsf{v}}_{[l]} = \mathsf{H}_l^{\mathsf{T}} \bar{\mathsf{v}}_{[l+1]}. \tag{3.74}$$

However, as we shall see in the next section, while the hierarchical constraints are described at a coarse resolution, the corresponding deformation response computed with hierarchical GFs involves all scales.

## 3.5 Hierarchical Green's Functions

The GF responses corresponding to each hierarchical constraint basis function are named *hierarchical GFs*. From a GF matrix perspective, the coarsening of the constraint scales is associated with a reduction in GF columns (see Figure 3.6). A graphical illustration of hierarchical GFs is given in Figure 7.1 (p. 98).

### 3.5.1 Notation

The hierarchical GFs are identified using the square bracket notation introduced for HBFs: let

$$\xi_{[l,k;d]} = \Xi_{:,[l,k;d]} \tag{3.75}$$

denote the hierarchical GF associated with the $k^{th}$ vertex contained on level $l$ and domain $D_d$. Therefore

$$\xi_{[0,k;d]}, \xi_{[1,k;d]}, \ldots, \xi_{[L,k;d]} \tag{3.76}$$

are all hierarchical GFs associated with the $k^{th}$ vertex here contained on the base level of the subdivision connectivity mesh. The hierarchical *wavelet* GFs (illustrated in Figure 3.6) are easily identified by both a tilde and square brackets, e.g.,

$$\tilde{\xi}_{[l,k;d]} = \tilde{\Xi}_{:,[l,k;d]}. \tag{3.77}$$

### 3.5.2 Refinement Relation

Hierarchical GFs and hierarchical basis functions share the same refinement filters since each hierarchical GF is expressed in terms of a linear combination of GFs on finer levels by

$$\xi_{[l,k;d]} = \sum_{j \in \mathcal{K}(l+1)} h_{[l,k,j;d]}\, \xi_{[l+1,j;d]} \tag{3.78}$$

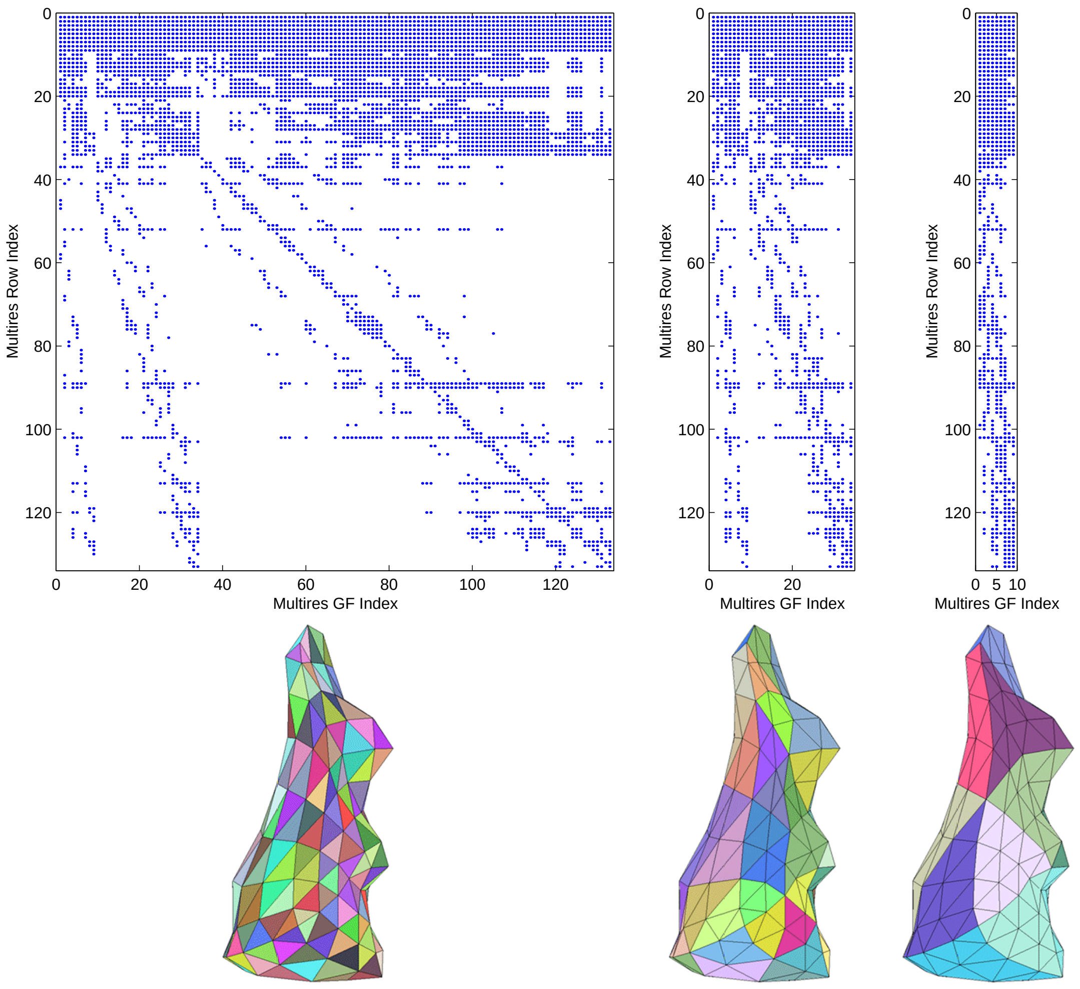


Figure 3.6: *Illustration of Hierarchical Wavelet GF Matrix Structure:* Sparsity patterns and constraint parameterizations of the coarse level 2 (L=2) rabbit model's three level GF hierarchy for the main $\Lambda_p^0$ "free-boundary" self-effect block $\Xi_{\Lambda_p^0 \Lambda_p^0}$ (illustrated in Figure 3.4). This model has 160 vertices, with the lifted linear FWT defined on a domain of 133 vertices partitioned into three levels with sizes (9,25,99). The matrices are: (left) finest scale GF square matrix block (# nonzero blocks, nnz=4444), (middle) once-coarsened constraint scale GF block (nnz=1599), (right) twice-coarsened constraint scale GF block (nnz=671). In each case, sparsity resulting from thresholding the wavelet transformed GF columns clearly illustrates the wavelet transform's excellent decorrelation ability. The multiresolution structure of the wavelet coefficients is apparent in each matrix as a result of multiresolution reordering of rows and columns; notice the dense unthresholded base level coefficients in the top most rows. Perhaps surprising for such a small model, modest compression ratios are already being obtained: here $\varepsilon = 0.10$ and the large block has retained nnz=4444 elements or $25\%$ of the original size.

or in operator notation

$$\Xi^l = \Xi^{l+1}\mathsf{H}_l^{\mathsf{T}}. \tag{3.79}$$

This follows from the hierarchical GF ansatz

$$\Xi^l \bar{\mathsf{v}}_{[l]} = \Xi^{l+1}\bar{\mathsf{v}}_{[l+1]}, \tag{3.80}$$

for a level $l$ hierarchical constraint $\bar{\mathsf{v}}_{[l]}$, after substituting the hierarchical boundary condition subdivision equation (3.74),

$$\bar{\mathsf{v}}_{[l]} = \mathsf{H}_l^{\mathsf{T}}\bar{\mathsf{v}}_{[l+1]}. \tag{3.81}$$

Figure 3.6 provides intuitive pictures of the induced GF hierarchy,

$$\xi_{[L,*;d]}, \ldots, \xi_{[1,*;d]}, \xi_{[0,*;d]}. \tag{3.82}$$

### 3.5.3 Matrix BVP Definition

While the refinement relation (3.79) can be used to compute coarse scale hierarchical GFs from finer resolutions, it is also possible to compute them directly using the definition of the accompanying hierarchical boundary value constraints. From (2.8), the matrix BVP satisfied by hierarchical GFs is

$$0 = \mathsf{A}\xi_{[l,k;d]} + \bar{\mathsf{A}}\bar{\mathsf{V}}_{[l,k;d]} \tag{3.83}$$

where the right-hand side constraint matrix $\bar{\mathsf{V}} \in \mathbb{R}^{3n\times 3}$ contains all zero nodal blocks except for nodes in the support of $\phi_{[l,k;d]}$,

$$(\bar{\mathsf{V}}_{[l,k;d]})_i = \phi_{[l,k;d]}(\mathbf{x}_i)\,\mathsf{I}_3. \tag{3.84}$$

This provides an attractive approach to (hierarchically) precomputing (and transmitting) very large models, and this is considered further in §6.

## 3.6 Hierarchical CMA

It is possible to use the hierarchical GFs to produce variants of the CMA from §2.2. The key benefits obtained from using hierarchical GFs are related to the smaller number of constraints (see Figure 3.8): (1) an accelerated fast summation (since fewer weighted columns need be summed), (2) smaller capacitance matrices, and (3) improved feasibility of caching potential capacitance matrix elements at coarse scales. Due to the 4-fold change in vertex count per resolution level, the expected impact of reducing the constraint resolution by $J$ levels is

1. $4^J$ reduction in constraint count and number of GFs required in CMA summations,

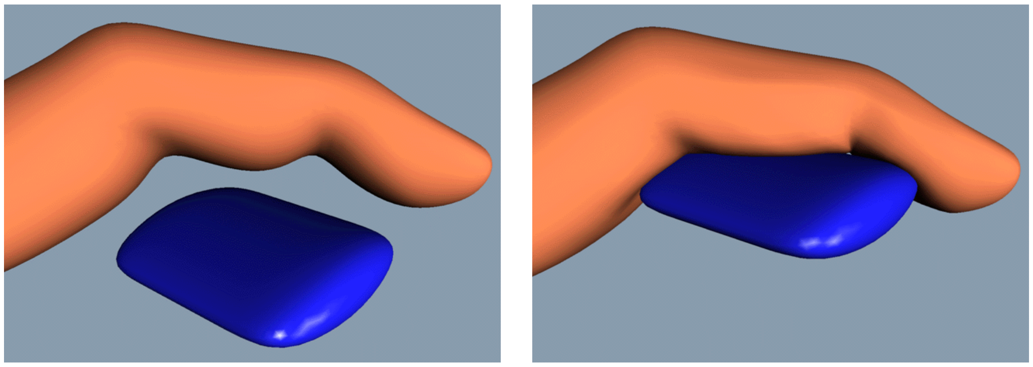


Figure 3.7: *Example where Hierarchical GFs are Useful:* A finger pad in contact with a flat surface is a good example of where hierarchical GFs are beneficial, as is any case where numerous dense surface constraints occur. Although the traction field may contain little information, e.g., smooth or nearly constant, large runtime costs can result from the number of GFs being summed and/or by the number of constraints being updated with a CMA. Whether the deformation is computed with the finger pad's free boundary constraints modeled by the user specifying tractions directly, or indirectly using displacements and a CMA, in both cases hierarchical GFs result in smaller boundable runtime costs.

2. $16^J$ reduction in number of capacitance matrix elements and cost of updating capacitance matrix inverses (when $s_\Delta \ll s$),
3. $64^J$ reduction in cost of factoring or directly inverting capacitance matrix,
4. $4^J - 64^J$ reduction in CMA cost.

An illustration of a situation where the hierarchical CMA can be beneficial is given in Figure 3.7.

It is relatively straight-forward to construct a nonadaptive hierarchical CMA that simply limits updated displacement constraints to fixed levels of resolution. This is the easiest mechanism for providing graceful degradation when large sets of nodes require updating: if too many constraints are being too densely applied they may simply be resolved on a coarser scale. This is analogous to using a coarser level model, with the exception that the solution, e.g., displacements, are available at a finer scale. We have found this simple approach works well in practice for maintaining interactivity during otherwise intensive updating cases. One drawback of the nonadaptive approach is that it can lead to "popping" when changing between constraint resolutions, and the investigation of adaptive CMA variants for which this problem is reduced are future work.

### 3.6.1 Hierarchical Capacitances

Similar to the nonhierarchical case, hierarchical capacitance matrices are submatrices of the hierarchical GFs. We can generalize the capacitance node list definition to include updated nodal constraints corresponding to hierarchical basis functions at different resolutions. We first generalize the notation of the original (fine scale) capacitance node list and capacitance matrix elements as

$$\begin{aligned} \mathsf{S} &= (k_1, k_2, \ldots, k_s) && (3.85) \\ &= ([L, k_1; d_1], [L, k_2; d_2], \ldots, [L, k_s; d_s]) && (3.86) \\ \mathsf{C}_{ij} &= -\Xi_{k_i[L,k_j;d_j]}. && (3.87) \end{aligned}$$

Hierarchical constraints then follow by replacing $L$ with the appropriate level. The CMA corresponding to coarsened constraint scales follows immediately, as well as the fact that hierarchical capacitance matrix inverses can be updated to add and delete hierarchical constraints. Furthermore, it is also possible to mix constraint scales and construct true multiresolution updates using the generalized definition

$$\begin{aligned} \mathsf{S} &= ([l_1, k_1; d_1], [l_2, k_2; d_2], \ldots, [l_s, k_s; d_s]) && (3.88) \\ \mathsf{C}_{ij} &= -\Xi_{k_i[l_j,k_j;d_j]}. && (3.89) \end{aligned}$$

However, due to the additional complexity of specifying adaptive multiresolution constraints at runtime, e.g., for an interactive contact mechanics problem, we have yet to exploit this CMA solver functionality in practice.

Finally, due to the reduced number of constraints, there are fewer and smaller capacitance matrices, and this improves the effectiveness of caching strategies (see Figure 3.8).

### 3.6.2 Graceful Degradation

For real time applications, hierarchical capacitances play an important role for resolving constraints on coarser constraint scales (or adaptively in general). Consider a simulation with constraints resolved on level $H$. If it encounters a capacitance matrix inverse update task which requires too much time it can abort and resort to resolving the problem at a coarser constraint resolution, e.g., $H - 1$ or lower. In this way it is possible to find a coarse enough level at which things can proceed quickly.

As with all variants of the CMA, these direct matrix solution algorithms provide predictable operation counts, which may be used to choose an effective real time solution strategy.

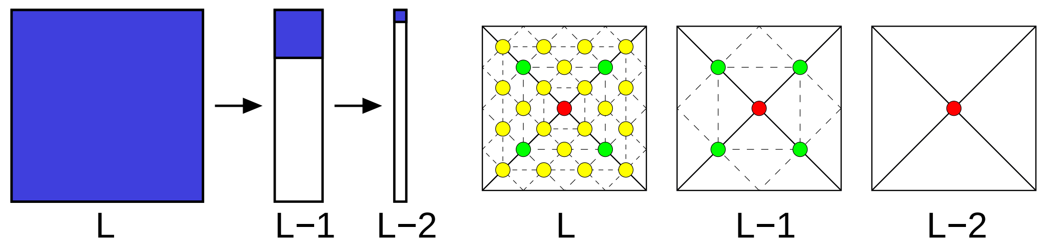


Figure 3.8: *Hierarchical Capacitance Matrices:* (Left) As in Figure 3.6, the matrix view of hierarchical GF indicates an approximately four-fold reduction in columns at each coarser constraint resolution. As a result, the number of possible capacitance matrix elements are reduced accordingly, as represented by the blue matrix blocks. (Right) An illustration of the corresponding spatial hierarchy for the support of a coarse level (extraordinary) "linear hat" scaling function. Circles indicate the vertex nodes (and basis functions) required to represent the coarse level scaling function at each level.

## 3.7 Detailed Graphical and Haptic Rendering

At some scale, there is little practical benefit in seeking higher resolution elastic models, and geometric detail can be introduced by local mapping.

### 3.7.1 LOD and Multresolution Displacement Fields

The fast summation CMA with wavelet GFs (§3.3) immediately provides an obvious mechanism for real time adaptive level-of-detail (LOD) rendering [XESV97]. This process is slightly complicated by the fact that the geometry is deforming, thereby reducing dependence on statically determined geometric quantities, e.g., visibility. While we have not explored real time LOD in our implementation, it was an important algorithm design consideration. It also provides an extra mechanism for real time graceful degradation for difficult CMA constraint problems.

### 3.7.2 Hierachical GFs and Geometric Detail

A favourable exploitation of spatial scales is obtained by using hierarchical GFs, since interactions resolved on relatively coarse constraints scales naturally allow visualization of fine scale geometry and displacement fields. Even when coarse level constraints are used, finer scale displacement fields are still available–possibly computed from an highly accurate discretization. An important point is that detail described by hierarchical GFs is represented in a global coordinate frame due to the geometrically linear elastic approximation, and therefore local displacement mapping is likely to be a better approximation for larger deformations. This is considered further in the following section.

### 3.7.3 Deformable Displaced Subdivision Surfaces

There is an interesting transition at some scale for which the GF displacement fields contain little more information than those obtained by displacement mapping a geometrically coarser resolution of the same model, the latter being possible in (future) graphics hardware. By only storing GF detail or computing multiresolution displacement fields to a suitable level, the deformed geometry can be mapped to finer scales via bump and/or displacement mapping. We have used displaced subdivision surfaces (DSS) [LMH00] for this purpose because it works well with deforming meshes.

A significant concern when displacement mapping coarse models is that it leads to inexact displacement constraints. This problem is exaggerated by DSS even for small changes due to mapping, because the Loop subdivision process converts our interpolating constraint scaling functions into noninterpolating ones. Intuitively, this occurs because adjacent vertex displacements computed by CMA for the coarse control mesh are averaged during the subdivision process, thus leading to inexact constraint values. This is in contrast to the interpolating constraints achieved with hierarchical GFs. Nevertheless, for finely meshed models the mismatch caused by displacement mapping is reduced.

One setting for which we have found DSS to be still very useful is for haptic force feedback applications involving point-like contacts. Here perceptual problems related to surface penetration due to inaccurate surface displacement constraints are commonly overcome by a “god-object approach” [ZS94] in which a proxy for the object in contact with the surface is always drawn on the surface (the “god” object) regardless of whether or not penetration occurs. We have successfully used this in several point-like contact interactive force feedback simulations, and pictures are shown in Figures 3.9 and 6.4.

### 3.7.4 Force feedback Rendering of Detail

In addition to graphical rendering, surface detail may also enhance force feedback rendering by using normal maps to modulate point contact friction forces [MS96] as is done in commercial force feedback systems, e.g., [Rea]. In this way, the hierarchical GFs parameterize the coarse scale force response of the compliant surface, while the normal maps render surface detail. Force feedback rendering of point-like contacts are considered further in §4.2.

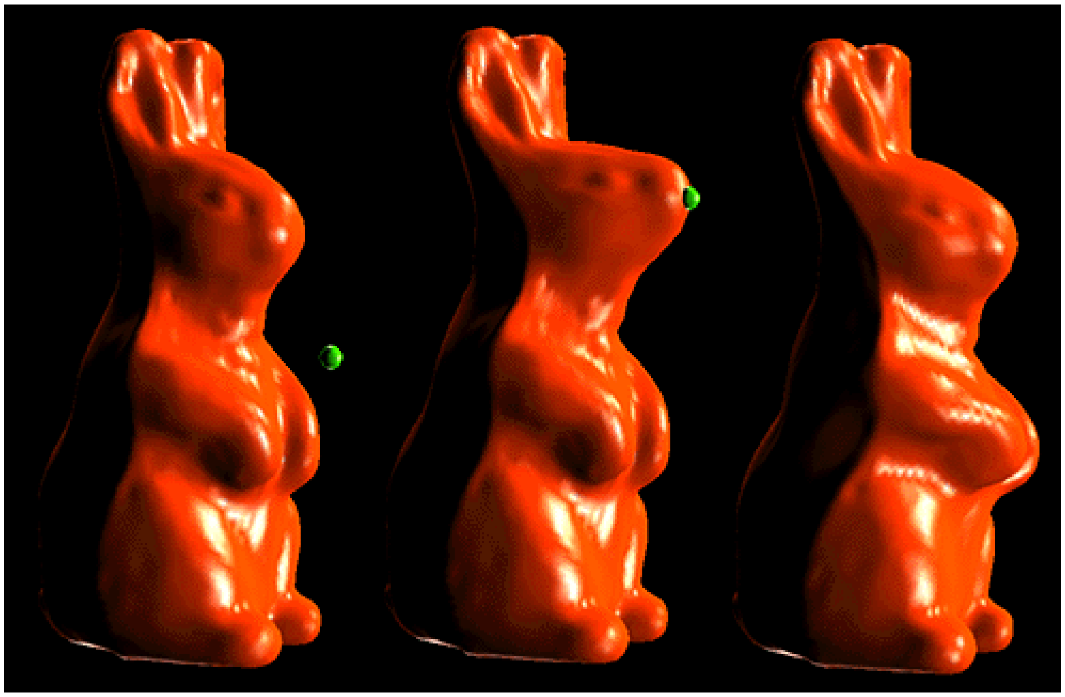


Figure 3.9: *Elastically Deformed Displaced Subdivision Surfaces:* Displaced subdivision surfaces provide a natural extension to an hierarchy of elastic spatial scales. In this example, a level 2 elastic rabbit model is rendered on level 5 using displacement mapping (computed in software). In addition to providing exact displacement constraints on detailed (or just subdivided) surfaces, hierarchical GFs allow greater elastic content to be depicted than simple displacement mapping of coarse geometry. In either case, such approaches can effectively transfer the runtime simulation burden almost entirely to graphical rendering.

# Chapter 4

# Haptic Interaction

## 4.1 Capacitance Matrices as Local Buffer Models

For force feedback enabled simulations in which user interactions are modeled as displacement constraints applied to an otherwise free boundary, the capacitance matrix has a very important role: it constitutes an exact contact force response model by describing the compliance of the contact zone. Borrowing terminology from [Bal00], we say that the capacitance matrix can be used as a *local buffer model*. While the capacitance matrix is used in §2.2.4 to determine the linear combination of GFs required to solve a particular BVP and reconstruct the global deformation, it also has the desirable property that it effectively decouples the global deformation calculation from that of the local force response. The most haptically relevant benefit is that the local contact force response may be computed at a much faster rate than the global deformation.

### 4.1.1 Capacitance Matrix Local Buffer Model

From (2.35), the $\mathsf{S}$ components of the solution $\mathsf{v}$ are

$$\begin{aligned}
\mathsf{E}^{\mathsf{T}}\mathsf{v} &= \mathsf{E}^{\mathsf{T}}\left[\mathsf{v}^{(0)} + (\mathsf{E} + (\Xi\mathsf{E}))\,\mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)}\right] && (4.1)\\
&= \mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} + \underbrace{(\mathsf{E}^{\mathsf{T}}\mathsf{E})}_{}\,\mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} + \underbrace{(\mathsf{E}^{\mathsf{T}}\Xi\mathsf{E})}_{}\,\mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} && (4.2)\\
\downarrow \qquad & \qquad\qquad\quad \mathsf{I} \qquad\qquad\qquad\quad -\mathsf{C} \quad \text{(from (2.36))} \\
&= \mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} + \mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} - \mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} && (4.3)\\
&= \mathsf{C}^{-1}\left(\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)}\right). && (4.4)
\end{aligned}$$

Consider the situation, which naturally arises in haptic interactions, in which the only nonzero constraints are updated displacement constraints, i.e.,

$$\bar{\mathsf{v}} = \mathsf{E}\mathsf{E}^{\mathsf{T}}\bar{\mathsf{v}} \quad \Rightarrow \quad \mathsf{v}^{(0)} = -\bar{\mathsf{v}} \quad \text{(using (2.37))}. \qquad (4.5)$$

In this case, the capacitance matrix completely characterizes the local contact response, since (using (4.5) in (4.1))

$$\mathsf{E}^\mathsf{T}\mathsf{v} = -\mathsf{C}^{-1}\mathsf{E}^\mathsf{T}\bar{\mathsf{v}}. \tag{4.6}$$

This in turn parametrizes the global response since these components (not in $\mathsf{S}$) are

$$\begin{aligned}(\mathsf{I}-\mathsf{E}\mathsf{E}^\mathsf{T})\mathsf{v} &= (\mathsf{I}-\mathsf{E}\mathsf{E}^\mathsf{T})\left[\mathsf{v}^{(0)} + (\mathsf{E}+(\Xi\mathsf{E}))\,\mathsf{C}^{-1}\mathsf{E}^\mathsf{T}\mathsf{v}^{(0)}\right] && (4.7)\\ &= \underbrace{(\mathsf{I}-\mathsf{E}\mathsf{E}^\mathsf{T})\mathsf{v}^{(0)}}_{0} + \underbrace{(\mathsf{I}-\mathsf{E}\mathsf{E}^\mathsf{T})\mathsf{E}}_{0}\,\mathsf{C}^{-1}\mathsf{E}^\mathsf{T}\mathsf{v}^{(0)} + (\mathsf{I}-\mathsf{E}\mathsf{E}^\mathsf{T})(\Xi\mathsf{E})\,\underbrace{\mathsf{C}^{-1}(\mathsf{E}^\mathsf{T}\mathsf{v}^{(0)})}_{\mathsf{E}^\mathsf{T}\mathsf{v}}\\ &\downarrow\\ &= (\mathsf{I}-\mathsf{E}\mathsf{E}^\mathsf{T})(\Xi\mathsf{E})(\mathsf{E}^\mathsf{T}\mathsf{v}) && (4.8)\end{aligned}$$

Such properties allow the capacitance matrix and $\Xi$ to be used to derive efficient local models for surface contact.

For example, given the specified contact zone displacements

$$\mathsf{u}_\mathsf{S} = \mathsf{E}^\mathsf{T}\bar{\mathsf{v}}, \tag{4.9}$$

the resulting tractions are

$$\mathsf{p}_\mathsf{S} = \mathsf{E}^\mathsf{T}\mathsf{v} = -\mathsf{C}^{-1}\left(\mathsf{E}^\mathsf{T}\bar{\mathsf{v}}\right) = -\mathsf{C}^{-1}\mathsf{u}_\mathsf{S}, \tag{4.10}$$

and the rendered contact force is

$$\mathbf{f} = \mathsf{a}_\mathsf{S}^\mathsf{T}\mathsf{p}_\mathsf{S} = \left(-\mathsf{a}_\mathsf{S}^\mathsf{T}\mathsf{C}^{-1}\right)\mathsf{u}_\mathsf{S} = \mathsf{K}_\mathsf{S}\mathsf{u}_\mathsf{S}, \tag{4.11}$$

where

$$\mathsf{a}_\mathsf{S} = (a_{\mathsf{S}_1}, a_{\mathsf{S}_2}, \ldots, a_{\mathsf{S}_s})^\mathsf{T} \otimes I_3 \tag{4.12}$$

represents the effective nodal areas (from (2.5, p. 16)) and $\mathsf{K}_\mathsf{S}$ is the effective stiffness of the contact zone used for force feedback rendering. A similar expression may be obtained for torque feedback. The visual deformation corresponding to solution components outside the contact zone is then given by (4.7) using $\mathsf{p}_\mathsf{S} = \mathsf{E}^\mathsf{T}\mathsf{v}$.

### 4.1.2 Example: Single Displacement Constraint

A simple case that is relevant for haptics (generalized in §4.2) consists of imposing a displacement constraint on a single node $k$ which otherwise had a traction constraint in the RBVP[1]. The new BVP therefore has only a single constraint switch with respect to the

[1]This case occurs, for instance, when the tip of a haptic device comes into contact with the free surface of an object.

RBVP, and so $s = 1$ and $\mathsf{S} = \{k\}$. The capacitance matrix here is just $\mathsf{C} = -\Xi_{kk}$ so that the $k^{th}$ nodal values are related by

$$\mathsf{p}_k = -\mathsf{C}^{-1}\mathsf{u}_k = (\Xi_{kk})^{-1}\,\mathsf{u}_k \qquad \text{or} \qquad \mathsf{u}_k = \Xi_{kk}\mathsf{p}_k. \tag{4.13}$$

The capacitance matrix can generate the force response, $\mathbf{f} = a_k \mathsf{p}_k$, required for haptics in $\mathcal{O}(1)$ operations, and for graphical feedback the corresponding global solution is $\mathsf{v} = \xi_k \mathsf{p}_k$.

### 4.1.3 Force feedback for Multiple Displacement Constraints

When multiple force feedback devices are interacting with the model by imposing displacement constraints, the force and stiffness felt by each device are tightly coupled in equilibrium. For example, the stiffness felt by the thumb in the grasping simulation shown in Figure 4.1 will depend on how other fingers are supporting the object. For multiple contacts like this, the capacitance matrix again provides an efficient force response model for haptics.

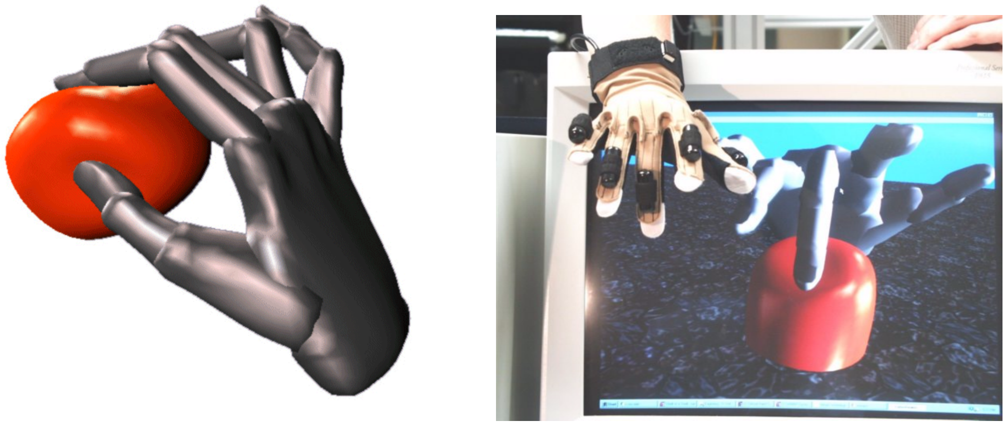


Figure 4.1: *Interactive grasping simulation* using a CyberGlove data input device (manufactured by Virtual Technologies Inc.). The virtual hand seen here was used to interactively deform a smooth elastostatic BEM model with approximately 900 surface degrees of freedom (dof) at graphical frame rates (30 frames per second) on a personal computer (dual Pentium II 450 MHz). The capacitance matrix algorithm was used to impose displacement constraints on an otherwise free boundary, often updating over 100 dof per frame. While force feedback was not present, this section shows how the capacitance matrices computed could also have been used to render contact forces at a rate much higher than that of the graphical simulation.

The force responses for each of the contact patches can be derived from the capacitance matrix inverse in a manner similar to equations (4.9)-(4.12). In the simple case of

$d$ contact patches with updated node sets $\{\mathsf{S}^i\}_{i=1}^d$, the block partitioned capacitance matrix inverse, $\mathsf{P} = \mathsf{C}^{-1}$, describes each patch's traction response due to displacements at other patches,

$$\mathsf{P} = \sum_{i,j=1}^{d} \mathsf{E}_{\mathsf{S}^i} \mathsf{P}_{\mathsf{S}^i\mathsf{S}^j} \mathsf{E}_{\mathsf{S}^j}^{\mathsf{T}} \tag{4.14}$$

$$= \left[\mathsf{E}_{\mathsf{S}^1} \mathsf{E}_{\mathsf{S}^2} \dots \mathsf{E}_{\mathsf{S}^d}\right] \begin{bmatrix} \mathsf{P}_{\mathsf{S}^1\mathsf{S}^1} & \mathsf{P}_{\mathsf{S}^1\mathsf{S}^2} & \cdots & \mathsf{P}_{\mathsf{S}^1\mathsf{S}^d} \\ \mathsf{P}_{\mathsf{S}^2\mathsf{S}^1} & \mathsf{P}_{\mathsf{S}^2\mathsf{S}^2} & & \vdots \\ \vdots & & \ddots & \\ \mathsf{P}_{\mathsf{S}^d\mathsf{S}^1} & \cdots & & \mathsf{P}_{\mathsf{S}^d\mathsf{S}^d} \end{bmatrix} \begin{bmatrix} \mathsf{E}_{\mathsf{S}^1}^{\mathsf{T}} \\ \mathsf{E}_{\mathsf{S}^2}^{\mathsf{T}} \\ \vdots \\ \mathsf{E}_{\mathsf{S}^d}^{\mathsf{T}} \end{bmatrix} \tag{4.15}$$

where the $(i, j)$ stiffness matrix block

$$\mathsf{P}_{\mathsf{S}^i\mathsf{S}^j} = \mathsf{E}_{\mathsf{S}^i}^{\mathsf{T}} \mathsf{P} \mathsf{E}_{\mathsf{S}^j} \tag{4.16}$$

describes traction contributions to contact nodes in $\mathsf{S}^i$ from displacements at nodes in $\mathsf{S}^j$. The benefit of having an explicit capacitance matrix inverse, instead of a factorization, is clearly evident in this case.

## 4.2 Surface Stiffness Models for Point-like Contact

Point-like interactions are commonly discussed in the haptics literature for rigid surface models [MS94, HBS99], and also for linear elastic objects [CDA99], largely due to the availability of hardware for rendering 3 DOF force feedback. For elastic models, the benefit of point-like contacts is the convenience of the point-like parameterization of contact and not because the contact is highly concentrated or "pin-like". In fact, unlike their rigid counterparts, special care must be taken with elastic models to define meaningful contact areas for point-like interactions; point-like contacts defined only as single- or adjacent-vertex constraints will produce mesh-related artifacts when the mesh is refined (see Figure 4.2). We present an approach using *vertex pressure masks* which maintains the point contact description yet distributes forces on a specified scale. This allows point contact stiffnesses to be consistently defined as the mesh scale is refined, and provides an efficient method for force feedback rendering of forces with regular surface variations. Such a technique is presented here for point contacts, but could be generalized to other ill-posed or unresolved contact situations, e.g., contact with a line.

### 4.2.1 Vertex Pressure Masks for Distributed Point-like Contacts

In this section, the distribution of force is described using compactly-supported per-vertex pressure masks defined on the free boundary in the neighbourhood of each vertex.

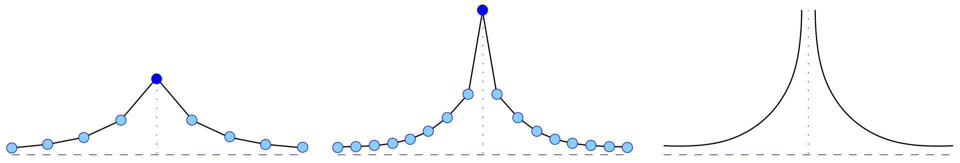


Figure 4.2: *Point Contact Must Not be Taken Literally for Elastic Models :* This figure illustrates the development of a displacement singularity associated with a concentrated surface force as the continuum limit is approached. In the left image, a unit force applied to a vertex of a discrete elastic model results in a finite vertex displacement. As the model's mesh is refined (middle and right image), the same concentrated force load eventually tends to produce a singular displacement at the contact location, and the stiffness of any single vertex approaches zero (see Table 4.1). Such constraints are mathematically ill-posed for linear models based on a small-strain assumption, and care should be taken to meaningfully define the interaction.

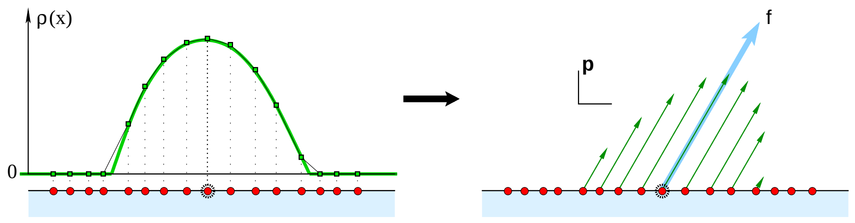


Figure 4.3: *Collocated Scalar Masks:* A direct means for obtaining a relative pressure amplitude distribution about each node, is to employ a user-specified scalar functional of the desired spatial scale. The scalar pressure mask is then given by nodal collocation (left), after which the vector traction distribution associated with a nodal point load is then computed as the product of the applied force vector and the (compactly supported) scalar mask (right).

### Vertex Pressure Mask Definition

Scalar pressure masks provide a flexible means for modeling vector pressure distributions associated with each node. This allows a force applied at the $i^{th}$ node to generate a traction distribution which is a linear combination of $\{\phi_j(\mathbf{x})\}$ and not just $\phi_i(\mathbf{x})$.

In the continuous setting, a scalar surface density $\rho(\mathbf{x}) : \Gamma \rightarrow \mathbb{R}$ will relate the localized contact force $\mathbf{f}$ to the applied traction $\mathbf{p}$ via[2]

$$\mathbf{p}(\mathbf{x}) = \rho(\mathbf{x})\mathbf{f} \tag{4.17}$$

which in turn implies the normalization condition

$$\int_\Gamma \rho(\mathbf{x}) d\Gamma_\mathbf{x} = 1. \tag{4.18}$$

[2]In a similar manner, tensor-valued masks for torque-feedback can also be computed.

In the discrete setting, the piecewise linear surface density on $\Gamma$ is

$$\rho(\mathbf{x}) = \sum_{j=1}^{n} \phi_j(\mathbf{x})\rho_j \in \mathcal{L}, \tag{4.19}$$

and is parameterized by the discrete scalar vertex mask vector,

$$\rho = [\rho_1, \rho_2, \ldots, \rho_n]^T. \tag{4.20}$$

Substituting (4.19) into (4.18), the discrete normalization condition satisfied becomes

$$a^T \rho = 1, \tag{4.21}$$

where $a$ are the vertex areas from (2.5). Notice that the mask density $\rho$ has units of $\frac{1}{\text{area}}$.

In practice, the vertex pressure mask $\rho$ may be specified in a variety of ways. It could be specified at runtime, e.g., as the byproduct of a physical contact mechanics solution, or be a user specified quantity. We shall consider the case where there is a compactly supported scalar function $\rho(\mathbf{x})$ specified at each vertex on the free boundary. The corresponding discrete vertex mask $\rho$ may then be defined using nodal collocation (see Figure 4.3),

$$\rho_j = \begin{cases} \rho(\mathbf{x}_j), & j \in \Lambda_p^0, \\ 0, & j \in \Lambda_u^0. \end{cases}, \tag{4.22}$$

followed by suitable normalization,

$$\rho := \frac{\rho}{a^T \rho}, \tag{4.23}$$

to ensure the satisfaction of (4.21).

In the following, denote the density mask for the $i^{th}$ vertex by the $n$-vector $\rho^i$, with nonzero values being indicated by the set of masked nodal indices $\mathcal{M}_i$. Since the intention is to distribute force on the free boundary, masks will only be defined for $i \in \Lambda_p^0$. Additionally, these masks will only involve nodes on the free boundary, $\mathcal{M}_i \subset \Lambda_p^0$, as well as be nonempty, $|\mathcal{M}_i| > 0$.

## Example: Spherical Mask Functionals

Spherically symmetric radially decreasing mask functionals with a scale parameter were suitable candidates for constructing vertex masks via collocation on smooth surfaces. One functional we used (see Figure 4.4 and 4.5) had linear radial dependence,

$$\rho^i(\mathbf{x}; r) = \begin{cases} 1 - \frac{|\mathbf{x} - \mathbf{x}_i|}{r}, & |\mathbf{x} - \mathbf{x}_i| < r, \\ 0, & otherwise. \end{cases}, \tag{4.24}$$

where $r$ specifies the radial scale[3]. The effect of changing r is shown in Figure 4.4.

[3] $r$ may be thought of as the size of the haptic probe's tip.

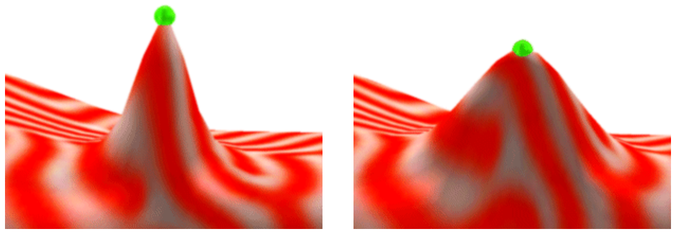


Figure 4.4: *Illustration of Changing Mask Scale:* An exaggerated pulling deformation illustrates different spatial scales in two underlying traction distributions. In each case, pressure masks were generated using the linear spherical mask functional (see §4.2.1) for different values of the radius parameter, $r$.

### 4.2.2 Vertex Stiffnesses using Pressure Masks

Having consistently characterized point-like force loads using vertex pressure masks, it is now possible to calculate the stiffness of each vertex. In the following sections, these vertex stiffnesses will then be used to compute the stiffness at any point on model's surface for haptic rendering of point-like contact.

#### Elastic Vertex Stiffness, $\mathsf{K}^{\mathsf{E}}$

For any single node, $i$, on the free boundary, $i \in \Lambda_p^0$, a finite force stiffness, $\mathsf{K}_i \in \mathbb{R}^{3\times3}$, may be associated with its displacement, i.e.,

$$\mathbf{f} = \mathsf{K}_i \mathsf{u}_i, \quad i \in \Lambda_p^0. \tag{4.25}$$

As a sign convention, it will be noted that for any single vertex displacement

$$\mathsf{u}_i \cdot \mathbf{f} = \mathsf{u}_i \cdot (\mathsf{K}_i \mathsf{u}_i) \geq 0, \quad i \in \Lambda_p^0 \tag{4.26}$$

so that positive work is done deforming the object.

Given a force $\mathbf{f}$ applied at vertex $i \in \Lambda_p^0$, the corresponding distributed traction constraints are

$$\mathsf{p}_j = \rho_j^i \mathbf{f}. \tag{4.27}$$

Since the displacement of the $i^{th}$ vertex is

$$\mathsf{u}_i = \sum_{j \in \mathcal{M}_i} \rho_j^i \Xi_{ij} \mathbf{f}, \tag{4.28}$$

the effective stiffness of the masked vertex is

$$\boxed{\mathsf{K}_i = \mathsf{K}_i^{\mathsf{E}} = \left( \sum_{j \in \mathcal{M}_i} \rho_j^i \Xi_{ij} \right)^{-1}, \quad i \in \Lambda_p^0.} \tag{4.29}$$

Some examples are provided in Table 4.1 and Figure 4.5.

Therefore, in the simple case of a single masked vertex displacement constraint $\mathsf{u}_i$, the local force response model exactly determines the resulting force, $\mathbf{f} = \mathsf{K}_i \mathsf{u}_i$, distributed in the masked region. The corresponding globally consistent solution is

$$\mathsf{v} = \zeta_i \mathbf{f} = \left( \sum_{j \in \mathcal{M}_i} \rho_j^i \xi_j \right) \mathbf{f} \tag{4.30}$$

where $\zeta_i$ is the convolution of the GFs with the mask $\rho$, and characterizes the distributed force load. The limiting case of a single vertex constraint corresponds to $\mathcal{M}_i = \{i\}$ with $\rho_j^i = \delta_{ij}/a_i$ so that the convolution simplifies to $\zeta_i = \xi_i / a_i$.

| Subdivision Level | # Vertices | Single vertex $\|\mathsf{K}_{top}\|_F$ | Masked vertex $\|\mathsf{K}_{top}\|_F$ |
|---|---|---|---|
| 1 | 34 | 7.3 | 13.3 |
| 2 | 130 | 2.8 | 11.8 |
| 3 | 514 | 1.1 | 11.2 |

Table 4.1: *Vertex Stiffness Dependence on Mesh Resolution:* This table shows vertex stiffness magnitudes (arbitrary units) at the top center vertex of the BEM model in Figure 7.18(a), as geometrically modeled using Loop subdivision meshes for three different resolutions. The stiffness corresponding to a single vertex constraint exhibits a large dependence on mesh resolution, and has a magnitude which rapidly decreases to zero as the mesh is refined. On the other hand, the stiffness generated using a vertex pressure mask (collocated linear sphere functional (see §4.2.1) with radius equal to the coarsest mesh's mean edge length) has substantially less mesh dependence, and quickly approaches a nonzero value.

### Rigid Vertex Stiffness, $\mathsf{K}^{\mathsf{R}}$

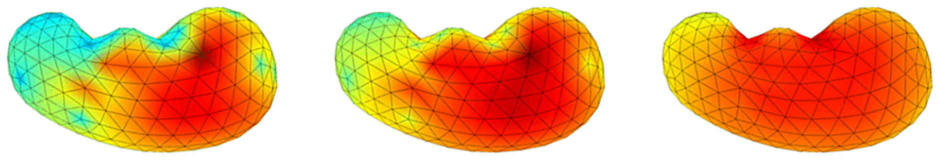
For rigid surfaces a finite force response may be defined using an isotropic stiffness matrix,

$$\mathsf{K}^{\mathsf{R}} = k^{\text{Rigid}} \mathsf{I}_3 \in \mathbb{R}^{3\times 3}, \qquad k^{\text{Rigid}} > 0. \tag{4.31}$$

This is useful for defining responses at position constrained vertices of a deformable model,

$$\mathsf{K}_i = \mathsf{K}^{\mathsf{R}}, \quad i \in \Lambda_u^0, \tag{4.32}$$

for at least two reasons. First, while it may seem physically ambiguous to consider contacting a constrained node of a deformable object, it does allow us to define a response for these vertices without introducing other simulation dependencies, e.g., how the haptic interaction with the elastic object support is modeled. Second, we shall see in §4.2.3 that defining stiffness responses at these nodes is important for determining contact responses on neighbouring triangles which are not rigid.



(a) $a(\mathbf{x})$ (b) $\|\mathsf{K}(\mathbf{x})\|$ (c) masked $\|\mathsf{K}(\mathbf{x})\|$

Figure 4.5: *Effect of Pressure Masks on Surface Stiffness:* Even models with reasonable mesh quality, such as this simple BEM kidney model, can exhibit perceptible surface stiffness irregularities when single-vertex stiffnesses are used. A plot (a) of the vertex area, $a$, clearly indicates regions of large (dark red) and small (light blue) triangles. In (b) the norm of the single-vertex surface stiffness, $\|\mathsf{K}(\mathbf{x})\|$, reveals a noticeable degree of mesh-related stiffness artifacts. On the other hand, the stiffness plotted in (c) was generated using a pressure mask (collocated linear sphere functional (see §4.2.1) of radius twice the mesh's mean edge length) and better approximates the regular force response expected of such a model.

### 4.2.3 Surface Stiffness from Vertex Stiffnesses

Given the vertex stiffnesses, $\{\mathsf{K}_i\}_{i=1}^n$, the stiffness of any location on the surface is defined using nodal interpolation

$$\mathsf{K}(\mathbf{x}) = \sum_{i=1}^{n} \phi_i(\mathbf{x})\mathsf{K}_i, \quad \mathbf{x} \in \Gamma, \tag{4.33}$$

so that $(\mathsf{K}(\mathbf{x}))_{kl} \in \mathcal{L}$. Note that there are no more than three nonzero terms in the sum of (4.33), corresponding to the vertices of the face in contact. In this way, the surface stiffness may be continuously defined using only $|\Lambda_p^0|$ free boundary vertex stiffnesses and a single rigid stiffness parameter, $k^{\text{Rigid}}$, regardless of the extent of the masks. The global deformation is then visually rendered using the corresponding distributed traction constraints.

For a point-like displacement constraint applied at $\mathbf{x}\in\Gamma$ on a triangle having vertex indices $\{i_1, i_2, i_3\}$, the corresponding global solution is

$$\mathsf{v} = \sum_{i\in\{i_1,i_2,i_3\}} \zeta_i \phi_i(\mathbf{x})\mathbf{f}. \tag{4.34}$$

This may be interpreted as the combined effect of barycentrically distributed forces, $\phi_i(\mathbf{x})\mathbf{f}$, applied at each of the triangle's three masked vertex nodes, which is consistent with (C.8).

### 4.2.4 Rendering with Finite Stiffness Haptic Devices

Similar to haptic rendering of rigid objects, elastic objects with stiffnesses greater than some maximum renderable magnitude (due to hardware limitations) are haptically displayed as

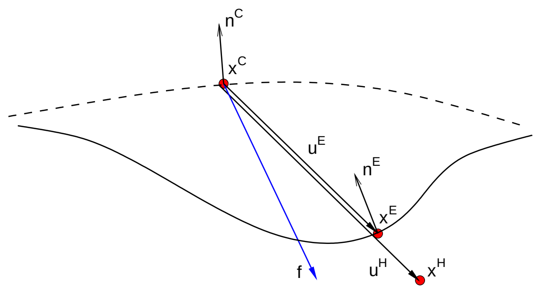


Figure 4.6: *Geometry of a Point-like Contact:* The surface of the static/undeformed geometry (curved dashed line) and that of the deformed elastic model (curved solid line) are shown along with: applied force ($\mathbf{f}$), static contact location ($\mathsf{x}^\mathsf{C}$), deformed elastic model contact location ($\mathsf{x}^\mathsf{E}$), haptic probe-tip location ($\mathsf{x}^\mathsf{H}$), haptic contact displacement ($\mathsf{u}^\mathsf{H} = \mathsf{x}^\mathsf{H} - \mathsf{x}^\mathsf{C}$), elastic contact displacement ($\mathsf{u}^\mathsf{E} = \mathsf{x}^\mathsf{E} - \mathsf{x}^\mathsf{C}$), static contact normal ($\mathsf{n}^\mathsf{C}$) and elastic contact normal ($\mathsf{n}^\mathsf{E}$). Once the contact is initiated by the collision detector, the sliding contact can be tracked in surface coordinates at force feedback rates.

softer materials during continuous contact. This can be achieved using a *haptic vertex stiffness*, $\mathsf{K}^\mathsf{H}_j$, which is proportional to the elastic vertex stiffness. While the stiffnesses could all be uniformly scaled on the free boundary, this can result in very soft regions if the model has a wide range of surface stiffness. Another approach is to set

$$\mathsf{K}^\mathsf{H}_j = \eta_j \mathsf{K}^\mathsf{E}_j \quad \text{where} \quad \eta_j = \min\left(1, \frac{\|\mathsf{K}^\mathsf{R}\|}{\|\mathsf{K}^\mathsf{E}_j\|}\right), \tag{4.35}$$

so that the elastic haptic model is never more stiff than a rigid haptic model. The surface's haptic stiffness $\mathsf{K}^\mathsf{H}(\mathbf{x})$ is then determined as before, using equation (4.33), so that $\|\mathsf{K}^\mathsf{H}(\mathbf{x})\| \le \|\mathsf{K}^\mathsf{R}\|, \forall \mathbf{x} \in \Gamma$.

In accordance with force reflecting contact, the deformed elastic state corresponds to the haptic force applied at the contact location $\mathsf{x}^\mathsf{C}$. This produces geometric contact configurations similar to that shown in Figure 4.6, where the haptic displacement $\mathsf{u}^\mathsf{H}$ can differ from the elastic displacement $\mathsf{u}^\mathsf{E}$. The geometric deformation is determined from the applied force $\mathbf{f}$ and equation (4.34). Note that when the haptic and elastic stiffnesses are equal, such as for soft materials, then so are the elastic and haptic displacements. In all cases, the generalized "god object" [ZS94] or "surface contact point" [Sen] is defined as the parametric image of $\mathbf{x}^\mathsf{C}$ on the deformed surface.

# Chapter 5

# Advanced Modeling Techniques

In this chapter we describe applications of the CMA formalism to some advanced modeling scenarios.

## 5.1 Multizone Kinematic Green's Function Models

Multizone models refer to models described by several LEGFMs ("zones") joined by some physical constraints (see Figure 5.1 and 5.2). This type of domain decomposition is useful because (1) certain models are easily described or constructed using separate components, (2) it can reduce precomputation and storage costs for boundary descriptions, and (3) it is useful for simulating kinematic structures involving elastic material, as well as for approximating nonlinear strain in structures with rotation.

Multizone substructuring methods are commonly used in boundary element analysis, e.g., [KKP91], to avoid the construction of large dense BEM matrices ($\mathsf{H}$ and $\mathsf{G}$) such as for direct solvers and also for use with Krylov iterative methods. A common strategy is to use condensation on the system of multizone equations to arrive at a reduced system of equations relating only boundary values belonging to interzonal interfaces. Once the condensed system is solved, the interface values are used to construct the solution on each domain. Such a multizone approach is centrally related to domain decomposition.

The reason multizone models are of interest here is that precomputed GFs for each zonal LEGFM provide the condensed multizone matrix description "for free" since the response of each zone is known in the form of the GF lookup table. The solution to these condensed interface equations is then used to compute the deformation of each zonal model, similar to the capacitance matrix algorithm. Provided this can be done efficiently, large coupled systems can be interactively simulated. It follows immediately that multiresolution GFs and constraint descriptions can be used for multizone models, and in fact our implementation (discussed later) can transparently use the MR GF implementation to gain speed-up and storage benefits.

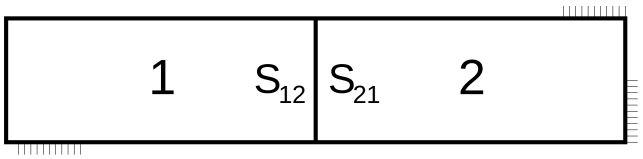


Figure 5.1: *Two zone model* with zones and interface node sets indicated.

Finally, material related to this section appears in [JP02].

### 5.1.1 Multizone Model Description

For the simple two zone model illustrated in Figure 5.1 consisting of zones 1 and 2, let the quantities associated with each LEGFM zone be denoted by a superscript 1 or 2. For example, the GFs of each zone are $\Xi^1$ and $\Xi^2$, the traction fields are $\mathsf{p}^1$ and $\mathsf{p}^2$, and similarly for displacement fields. Initially, all quantities will be defined in a common coordinate system, although this will be generalized later in §5.1.3 for more general kinematic relationships.

Let the interface between the zones be defined by two ordered lists of nodes: let $\mathsf{S}_{12}$ denote nodes in zone 1 interfacing with zone 2, and similarly let $\mathsf{S}_{21}$ represent nodes in zone 2 contacting zone 1. Without loss of generality, we shall assume that the interface discretizations conform, so that the interface lists are the same size ($|\mathsf{S}_{12}| = |\mathsf{S}_{21}|$), and that the $j^{th}$ node of each list corresponds to the same interface vertex. The interface displacements and tractions for zone 1 are then the arrays $\mathsf{u}^1_{\mathsf{S}_{12}}$ and $\mathsf{p}^1_{\mathsf{S}_{12}}$, while for zone 2 they are $\mathsf{u}^2_{\mathsf{S}_{21}}$ and $\mathsf{p}^2_{\mathsf{S}_{21}}$.

### 5.1.2 Multizone Equations and Condensation

As an example, consider the two-zone model joined along a seam containing nodes with traction boundary conditions in the reference BVP[1], i.e., seam nodes in $\Lambda^0_p$ for each system. The interface displacements and tractions are governed by

$$\mathsf{u}^1_{\mathsf{S}_{12}} = \hat{\mathsf{u}}^1_{\mathsf{S}_{12}} + \Xi^1_{\mathsf{S}_{12}\mathsf{S}_{12}}\mathsf{p}^1_{\mathsf{S}_{12}} \tag{5.1}$$
$$\mathsf{u}^2_{\mathsf{S}_{21}} = \hat{\mathsf{u}}^2_{\mathsf{S}_{21}} + \Xi^2_{\mathsf{S}_{21}\mathsf{S}_{21}}\mathsf{p}^2_{\mathsf{S}_{21}} \tag{5.2}$$

where the interfaces' self-influence matrices are

$$\Xi^1_{\mathsf{S}_{12}\mathsf{S}_{12}} = \mathsf{E}^{\mathsf{T}}_{\mathsf{S}_{12}}\Xi^1\mathsf{E}_{\mathsf{S}_{12}} \tag{5.3}$$
$$\Xi^2_{\mathsf{S}_{21}\mathsf{S}_{21}} = \mathsf{E}^{\mathsf{T}}_{\mathsf{S}_{21}}\Xi^2\mathsf{E}_{\mathsf{S}_{21}} \tag{5.4}$$

and $\hat{\mathsf{u}}^1_{\mathsf{S}_{12}}$ and $\hat{\mathsf{u}}^2_{\mathsf{S}_{21}}$ describe the displacement contribution at the interface due to constraints outside the seam, e.g., a zone 1 nonzero traction constraint $\mathsf{p}^1_{j_1}$ at node $j_1 \notin \mathsf{S}_{12}$ would

[1]We will consider displacement reference BVP conditions in the following section.

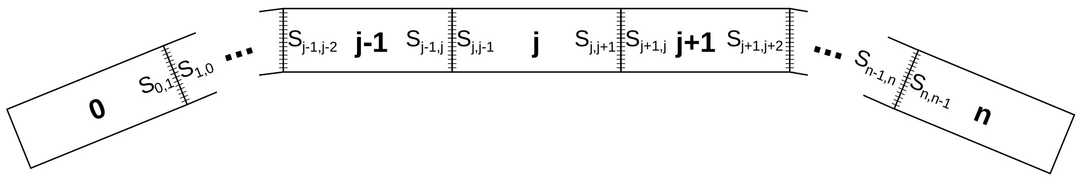


Figure 5.2: *Multizone elastic kinematic chain model schematic* with zones and interface node sets indicated. Here interface nodes have displacement constraints in each zone's RBVP, and external constraints are not shown.

imply

$$\hat{\mathsf{u}}^1_{\mathsf{S}_{12}} = \mathsf{E}^{\mathsf{T}}_{\mathsf{S}_{12}} \xi^1_{j_1} \mathsf{p}^1_{j_1}. \tag{5.5}$$

The benefit of the multizone approach when combined with LEGFMs is immediately apparent from (5.1-5.2): condensed interface equations are available practically "for free" when given the precomputed GFs. Similar to the force feedback application, the major benefit is that the equilibrium can be determined by solving problems involving only interface variables. For example, in order to physically bond the zonal LEGFM models at the interface one may use the interface boundary conditions

$$\mathsf{u}^1_{\mathsf{S}_{12}} = +\mathsf{u}^2_{\mathsf{S}_{21}} \qquad \text{(Continuity condition)} \tag{5.6}$$

$$\mathsf{p}^1_{\mathsf{S}_{12}} = -\mathsf{p}^2_{\mathsf{S}_{21}} \qquad \text{(Newton's } 3^{rd} \text{ law).} \tag{5.7}$$

Substituting these conditions in (5.1-5.2) yields the linear system to be solved to determine the interface constraints required to simulate the bonded material

$$\left(\Xi^1_{\mathsf{S}_{12}\mathsf{S}_{12}} + \Xi^2_{\mathsf{S}_{21}\mathsf{S}_{21}}\right) \mathsf{p}^2_{\mathsf{S}_{21}} = \hat{\mathsf{u}}^1_{\mathsf{S}_{12}} - \hat{\mathsf{u}}^2_{\mathsf{S}_{21}}. \tag{5.8}$$

Once the interface constraints are determined from this condensed interface equation they are applied to the LEGFMs in the usual way to compute the desired deformation. Because such condensed equations are available via a GF lookup operation, constructing such matrix systems is trivial.

### 5.1.3 Elastic Kinematic Chains

In many ways, LEGFMs should be thought of as nearly rigid objects with an inherent frame of reference. Attaching the elastostatic model to a kinematic structure, such as a rigid "bone" in a skeleton-based character animation (discussed in §5.1.4), is a natural way to associate this relationship in a simulation. By connecting LEGFMs with multiple frame of references together, much larger relative deformations can be achieved than would otherwise be possible with a single LEGFM attached to multiple moving bones. While LEGFMs' linear Cauchy strain approximation of Green strain is not invariant under rotation, it is quite possible to rotate LEGFMs relative to each other. As an example, later in §5.1.4 we shall consider secondary animation of finger pad deformation during interactive grasping and contact tasks with a simplified finger model.

In this section, we shall consider a multizone kinematic chain of coupled LEGFMs as schematically shown in Figure 5.2. This multizone kinematic configuration results in a block tridiagonal system of equations comprising a nonlinear compliance equation relating the interface displacements defined in each LEGFM's inherent coordinate frame. The stiffness response is nonlinear in the sense that it depends on the configuration of the joints in the chain, but for any single configuration it is linear.

We will describe kinematic chains only for notational simplicity, but the same arguments apply to more general multizone interface topologies and kinematic structures. The practical consequence is that the solution of the seam equations will cease to be block tridiagonal, but may still be efficiently solved. The issues, and means of handling them, are similar to those in multi-rigid body dynamics [APC97, LNPE92, Bar96, Fea87].

Similar boundary influence equations for coupling elastic bodies in equilibrium arise when computing contact constraints between multiple elastic objects [AB93, MAR93, EO89]. Using the precomputed GF models, the contact response could be efficiently computed and integrated at high rates to solve simplified contact problems interactively with haptic force feedback.

The LEGFM's zonal frames of reference are related by coordinate transformations: let the operator ${}^{j}_{j+1}\mathsf{F}$ map quantities from frame $j+1$ to $j$, and ${}^{j+1}_{j}\mathsf{F}$ from $j$ to $j+1$. A left superscript will denote the frame of reference of a quantity, e.g., ${}^{j}a^{i}$ represents a quantity $a$ of zone $i$ in frame of zone $j$. Some illustrative coordinate transformations of quantities in zone $j$ from frame $j$ to $j+1$ are

$$\begin{aligned} {}^{j+1}\mathsf{u}^{j}_{\mathsf{S}} &= {}^{j+1}_{j}\mathsf{F}\ {}^{j}\mathsf{u}^{j}_{\mathsf{S}} \qquad (5.9) \\ {}^{j+1}\Xi^{j}_{\mathsf{SS}} &= {}^{j+1}_{j}\mathsf{F}\ {}^{j}\Xi^{j}_{\mathsf{SS}}. \qquad (5.10) \end{aligned}$$

Taking the RBVP of each zone to have specified displacement boundary conditions for seam constraints (see Figure 5.2), the LEGFM equations describing the traction response on both sides ($j$ and $j+1$) of seam $j = 0, 1, \ldots, n-1$ are

$$\begin{aligned} {}^{j}\mathsf{p}^{j}_{\mathsf{S}_{j,j+1}} &= {}^{j}\hat{\mathsf{p}}^{j}_{\mathsf{S}_{j,j+1}} + {}^{j}\Xi^{j}_{\mathsf{S}_{j,j+1}\mathsf{S}_{j,j+1}}\ {}^{j}\mathsf{u}^{j}_{\mathsf{S}_{j,j+1}} + {}^{j}\Xi^{j}_{\mathsf{S}_{j,j+1}\mathsf{S}_{j,j-1}}\ {}^{j}\mathsf{u}^{j}_{\mathsf{S}_{j,j-1}} \\ {}^{j+1}\mathsf{p}^{j+1}_{\mathsf{S}_{j+1,j}} &= {}^{j+1}\hat{\mathsf{p}}^{j+1}_{\mathsf{S}_{j+1,j}} + {}^{j+1}\Xi^{j+1}_{\mathsf{S}_{j+1,j}\mathsf{S}_{j+1,j}}\ {}^{j+1}\mathsf{u}^{j+1}_{\mathsf{S}_{j+1,j}} + {}^{j+1}\Xi^{j+1}_{\mathsf{S}_{j+1,j}\mathsf{S}_{j+1,j+2}}\ {}^{j+1}\mathsf{u}^{j+1}_{\mathsf{S}_{j+1,j+2}} \end{aligned}$$

where the node sets $\mathsf{S}_{0,-1}$ and $\mathsf{S}_{n,n+1}$ may be taken as the empty set to yield the end seam equations

$$\begin{aligned} {}^{0}\mathsf{p}^{0}_{\mathsf{S}_{01}} &= {}^{0}\hat{\mathsf{p}}^{0}_{\mathsf{S}_{01}} + {}^{0}\Xi^{0}_{\mathsf{S}_{01}\mathsf{S}_{01}}\ {}^{0}\mathsf{u}^{0}_{\mathsf{S}_{01}} \qquad (5.11) \\ {}^{n}\mathsf{p}^{n}_{\mathsf{S}_{n,n-1}} &= {}^{n}\hat{\mathsf{p}}^{n}_{\mathsf{S}_{n,n-1}} + {}^{n}\Xi^{n}_{\mathsf{S}_{n,n-1}\mathsf{S}_{n,n-1}}\ {}^{n}\mathsf{u}^{n}_{\mathsf{S}_{n,n-1}}. \qquad (5.12) \end{aligned}$$

The seam bonding conditions, accounting for transformations between zone frames of reference, are

$$\begin{aligned} {}^{j}\mathsf{x}^{j}_{\mathsf{S}_{j,j+1}} + {}^{j}\mathsf{u}^{j}_{\mathsf{S}_{j,j+1}} &= {}^{j}_{j+1}\mathsf{F}\left({}^{j+1}\mathsf{x}^{j+1}_{\mathsf{S}_{j+1,j}} + {}^{j+1}\mathsf{u}^{j+1}_{\mathsf{S}_{j+1,j}}\right) \qquad \text{(Continuity)} \quad (5.13)\\ 0 &= {}^{j}\mathsf{p}^{j}_{\mathsf{S}_{j,j+1}} + {}^{j}_{j+1}\mathsf{F}\,{}^{j+1}\mathsf{p}^{j+1}_{\mathsf{S}_{j+1,j}} \qquad \text{(Newton's } 3^{rd} \text{ law)} \quad (5.14)\end{aligned}$$

where ${}^{j}\mathsf{x}^{j}_{*}$ represent undisplaced vertex positions. These seam conditions may be rewritten for substitution as

$$\begin{aligned} {}^{j+1}\mathsf{u}^{j+1}_{\mathsf{S}_{j+1,j}} &= {}^{j+1}_{j}\mathsf{F}\left({}^{j}\mathsf{x}^{j}_{\mathsf{S}_{j,j+1}} + {}^{j}\mathsf{u}^{j}_{\mathsf{S}_{j,j+1}}\right) - {}^{j+1}\mathsf{x}^{j+1}_{\mathsf{S}_{j+1,j}} \quad (5.15)\\ {}^{j+1}\mathsf{p}^{j+1}_{\mathsf{S}_{j+1,j}} &= -{}^{j+1}_{j}\mathsf{F}\,{}^{j}\mathsf{p}^{j}_{\mathsf{S}_{j,j+1}}. \quad (5.16)\end{aligned}$$

Substituting the bonding conditions into the LEGFM equations by eliminating variables with non-increasing interface index pairs[2] we obtain the nonsymmetric block tridiagonal seam equations relating each set of seam displacements (each in its natural frame of reference) to adjacent seams,

$$\begin{aligned} 0 &= \left({}^{j}\Xi^{j}_{\mathsf{S}_{j,j+1}\mathsf{S}_{j,j-1}}\,{}^{j}_{j-1}\mathsf{F}\right){}^{j-1}\mathsf{u}^{j-1}_{\mathsf{S}_{j-1,j}} \quad (5.17)\\ &+ \left({}^{j}\Xi^{j}_{\mathsf{S}_{j,j+1}\mathsf{S}_{j,j+1}} + {}^{j}\Xi^{j+1}_{\mathsf{S}_{j+1,j}\mathsf{S}_{j+1,j}}\,{}^{j+1}_{j}\mathsf{F}\right){}^{j}\mathsf{u}^{j}_{\mathsf{S}_{j,j+1}} \quad (5.18)\\ &+ \left({}^{j}\Xi^{j+1}_{\mathsf{S}_{j+1,j}\mathsf{S}_{j+1,j+2}}\right){}^{j+1}\mathsf{u}^{j+1}_{\mathsf{S}_{j+1,j+2}} \quad (5.19)\\ &+ {}^{j}\hat{\mathsf{p}}^{j}_{\mathsf{S}_{j,j+1}} + {}^{j}\hat{\mathsf{p}}^{j+1}_{\mathsf{S}_{j+1,j}} \quad (5.20)\\ &+ {}^{j}\Xi^{j}_{\mathsf{S}_{j,j+1}\mathsf{S}_{j,j-1}}\left({}^{j}\mathsf{x}^{j-1}_{\mathsf{S}_{j-1,j}} - {}^{j}\mathsf{x}^{j}_{\mathsf{S}_{j,j-1}}\right) \quad (5.21)\\ &+ {}^{j}\Xi^{j+1}_{\mathsf{S}_{j+1,j}\mathsf{S}_{j+1,j}}\left({}^{j+1}\mathsf{x}^{j}_{\mathsf{S}_{j,j+1}} - {}^{j+1}\mathsf{x}^{j+1}_{\mathsf{S}_{j+1,j}}\right) \quad (5.22)\end{aligned}$$

for $j = 0, 1, \ldots, n-1$ and $\mathsf{S}_{0,-1} = \mathsf{S}_{n,n+1} = \emptyset$.

This system of equations may be efficiently solved using block tridiagonal LU factorization [GL96] which is effective given small sets of interface variables. However, the system matrix's nonlinear dependence on chain orientation means that this factorization must be performed for each orientation, or that suitable interpolation must be used.

Two means for accelerating this process, both based on reducing the total number of interface variables, are as follows. The first approach, useful for thin seams, is to only constrain degrees of freedom near or on the surface. This approach was tried for the finger example after discovering that the difference was visually undetectable. A second and more general approach is to apply hierarchical GFs in seam regions to coarsen the seam constraints, thereby specifiably reducing the effort required to solve the coupled interface equations as well as simulate the LEGFM zone.

[2] For example, eliminate $\mathsf{u}_{\mathsf{S}_{j+1,j}}$ but keep $\mathsf{u}_{\mathsf{S}_{j,j+1}}$.

### 5.1.4 Implementation: Secondary Deformation for Character Animation

Multizone kinematic descriptions can make LEGFMs a useful tool for interactive animation of plausible body tissue deformations: not only does the model respond plausibly to skeletal motions, but the physical model also supports interaction. Just as clothing is draped over scripted animations of moving characters for *secondary animation*, deformations associated with motion, contact tasks and other forms of interaction may also be added. Considering only secondary animation avoids limitations associated with linear strain and a linear stress-strain relationship because the physical model is only used to compute plausible deformations for scripted motions and not to compute the motions themselves. Since LEGFMs are sufficiently fast, it is possible for them to be used in interactive applications such as video games. The interface-only feature of precomputed LEGFMs allows them to be easily combined with other types of physical models, e.g., a sophisticated model for wrinkled skin regions, as well as more traditional character animation approaches like vertex blending.

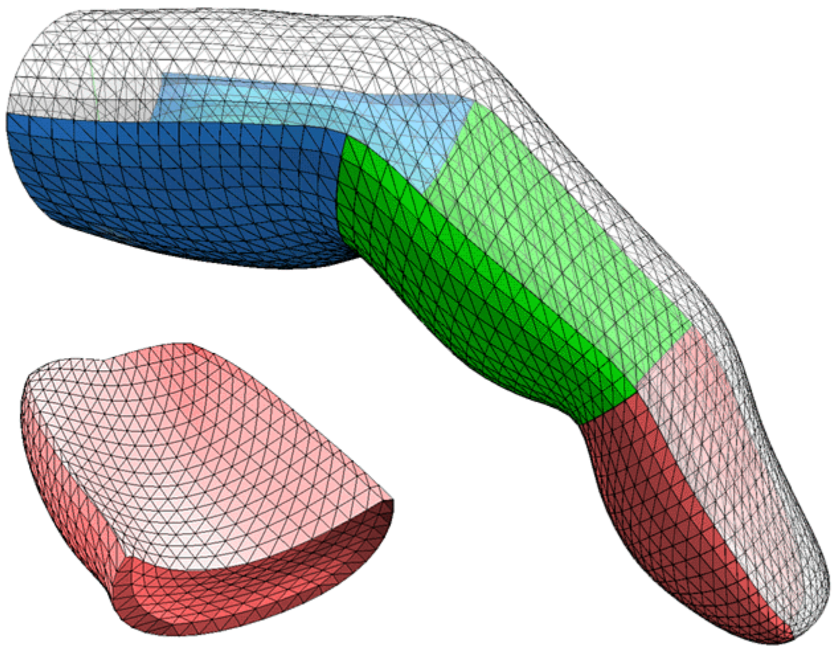


Figure 5.3: *Finger with elastic finger pads used for interactive simulation:* A simplified finger model composed of three individual multiresolution elastostatic finger pads ($L = 3$) with internal structure visible. The distal bone's fingertip pad is also drawn inset for clarity. Each finger pad is defined in a frame of reference rigidly attached to its corresponding "bone" and this allows large relative strains to be simulated during interactive character animation (see Figure 5.4). Significant wavelet compression of the finger pad models is also possible (§7.4).

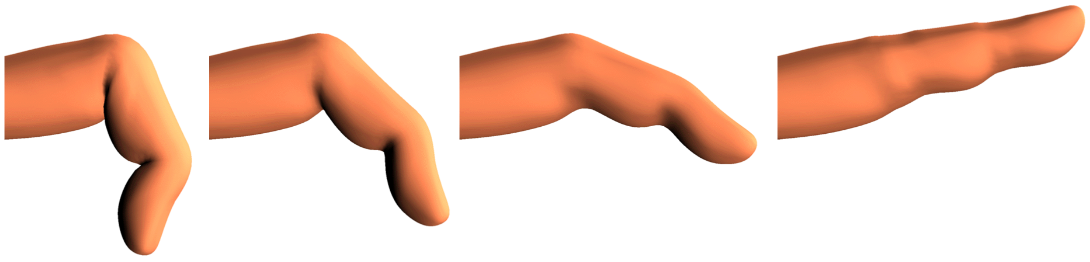


Figure 5.4: *Simulation of finger with elastic finger pads:* An illustrative bone-based skeletal animation of a finger modeled using three individual multiresolution elastostatic finger pads. The finger extension motion shown was computed using the fast summation CMA with hierarchical wavelet GFs, and runs at near 60 FPS in our ArtDefo simulator. The model may also be interacted with directly, and this is shown in Figure 3.7.

We have implemented a multizone model for interactive finger animation and this is described further in Figures 5.3 and 5.4. Precomputed multiresolution LEGFMs are used for each finger pad zone, with displacement constraints specified at seams (shown in Figure 5.4) in each zone's reference BVP. The finger pads are connected using continuity constraints at interface seams, e.g., located near kinematic joints. Imposing seam bonding constraints as in the previous section leads to a nonlinear interface compliance system.

For real time animation applications such as video games, we suggest an alternative approximation which decouples each zone and thus avoids the solution of a changing system matrix. This is achieved by specifying interface position constraints kinematically using bone-weighted vertex blending, a technique commonly used for character animation. As the kinematic joints move, the displacement constraints applied in each LEGFM's frame of reference also change. Because the constraints are determined kinematically, adjacent zones are decoupled from each other and this makes it unnecessary to solve for seam constraints. In this manner, incorporating LEGFMs into skeletal animations is a straightforward task, and the resulting system may be simulated in real time. This approach is discussed further in Figure 5.4 for the finger example.

### 5.1.5 Hybrid Models

Just as LEGFMs may be attached together, so may other physical models. Such hybrid models allow a larger variety of deformable objects to be constructed, with more expensive or accurate models used in areas where they are needed. Such approaches have been used in the surgical simulation community to allow more costly dynamic models to be used in regions of interest where dynamic tissue cutting is to be performed [HL98, CDA99]. It seems also possible to perform such decompositions adaptively using a hierarchy of multizone models, as well as with adaptive constraint resolution for each zonal interface.

## 5.2 Relationship of CMAs to Simulation of Nonlinear Physics

This section briefly mentions two relationships that CMA has with simulating nonlinear equilibrium physical systems. We remind the reader that material in this section has not been implemented, and is the subject of future work.

### 5.2.1 Interpretation of CMA as Sensitivity Analysis

It is possible to generalize the CMA to nonlinear elastostatics by interpreting GFs as local system responses obtained by sensitivity analysis methods. For example, a given RBVP has solutions characterized by the linear equation (2.15),

$$\mathsf{v} = \Xi \bar{\mathsf{v}}.$$

This is a linear approximation of the nonlinear relationship

$$\mathsf{v} = \mathsf{G}(\bar{\mathsf{v}}) \tag{5.23}$$

whose Taylor series expansion about $\bar{\mathsf{v}} = 0$ is

$$\begin{aligned} \mathsf{v} &= \mathsf{G}(\mathbf{0}) + (\nabla_{\bar{\mathsf{v}}} \mathsf{G}(\mathbf{0}))\, \bar{\mathsf{v}} + \dots && (5.24) \\ &= (\nabla_{\bar{\mathsf{v}}} \mathsf{G}(\mathbf{0}))\, \bar{\mathsf{v}} + \dots && (5.25) \end{aligned}$$

(since $\mathsf{G}(\mathbf{0}) = \mathbf{0}$ in the undeformed state), which implies the numerical approximation

$$\Xi \approx (\nabla_{\bar{\mathsf{v}}} \mathsf{G}(\mathbf{0}))\,. \tag{5.26}$$

By expanding about a nonzero deformed state $(\mathsf{w}, \bar{\mathsf{w}})$, where

$$\begin{aligned} \mathsf{v} &= \mathsf{w} + \delta \mathsf{v} && (5.27) \\ \bar{\mathsf{v}} &= \bar{\mathsf{w}} + \delta \bar{\mathsf{v}}, && (5.28) \end{aligned}$$

and $\mathsf{w} = \mathsf{G}(\bar{\mathsf{w}})$, the expansion

$$\mathsf{v} = \mathsf{w} + \delta \mathsf{v} = \mathsf{G}(\bar{\mathsf{w}}) + (\nabla_{\bar{\mathsf{w}}} \mathsf{G}(\bar{\mathsf{w}}))\, \delta \bar{\mathsf{v}} + \dots, \tag{5.29}$$

implies the local linear approximation

$$\begin{aligned} \delta \mathsf{v} &= (\nabla_{\bar{\mathsf{w}}} \mathsf{G}(\bar{\mathsf{w}}))\, \delta \bar{\mathsf{v}} && (5.30) \\ &= \Xi(\bar{\mathsf{w}}) \delta \bar{\mathsf{v}}. && (5.31) \end{aligned}$$

While it is not practical to precompute all $\Xi(\bar{\mathrm{w}})$ arising in a simulation[3] for arbitrary $\bar{\mathrm{w}}$, with sufficient computing resources, sensitivities due to changes in the BVP input, or GF responses such as $\Xi(\bar{\mathrm{w}})_{:j}$ could be computed. Temporal coherence and multiresolution approximations could make this process more feasible. By computing selected local GF sensitivites, the CMA can be used to update this subspace of constraints, and the provide stable haptic force feedback.

In this manner, a precomputed LEGFM and CMA could be used for fast simulation of small strain responses, including (sliding) contact, while a nonlinear CMA with runtime generated sensitivites is used to handle large deformations, and so interpolate between the more costly solutions and also derive local buffer models for force feedback.

### 5.2.2 Nonlinear Reanalysis

The idea of adaptively modifying zones to simulate nonlinear material response is directly related to recent developments in nonlinear reanalysis [AGH01] for which nonlinear material modifications for individual elements can be updated, e.g., to account for yielding or buckling. It is shown that the Sherman-Morrison and Woodbury formulas can be extended to allow for nonlinear changes in matrix quantities. In [AGH01], it is shown that a FEM stiffness equation

$$\mathbf{K}\mathbf{d_0} = \mathbf{f}. \tag{5.32}$$

can be updated efficiently to account for both linear and nonlinear modifications. In general, the modified equilibrium equation resulting from s members behaving nonlinearly takes the form

$$\mathbf{K}\mathbf{d} + \sum_{i=1}^{s} a_i(\mathbf{d})\mathbf{w}_i = \mathbf{f}, \tag{5.33}$$

where the volumetric displacement solution is of the form

$$\mathbf{d} = \mathbf{d_0} - \mathbf{R}\alpha \tag{5.34}$$

and

$$\mathbf{R} = \mathbf{K}^{-1}\mathbf{W}, \quad \mathbf{W} = [\mathbf{w_1}, \ldots, \mathbf{w_s}], \quad \alpha = [\alpha_\mathbf{1}, \ldots, \alpha_\mathbf{s}]^T. \tag{5.35}$$

Substituting (5.34) into (5.33) and using (5.32) and (5.35), lead to the condition

$$\sum_{i=1}^{s} \left(a_i(\mathbf{d}) - \alpha_i\right) \mathbf{w}_i = \mathbf{0}. \tag{5.36}$$

If the vectors $\mathbf{w}_i$ are linearly independent, this leads to s coupled nonlinear equations in $\alpha_i$

$$a_i(\mathbf{d}_0 - \mathbf{R}\alpha) - \alpha_i = 0, \quad i = 1, \ldots, s. \tag{5.37}$$

[3]For simplified interactions such as point-like contact, limited forms of precomputation may be possible, but this is not true in general.

If not all the changes are nonlinear then the problem is simpler: first a linear updating problem is solved and then a smaller nonlinear system of equations.

In order to produce a form of this that is suitable for interactive simulation, we observe that it may be possible to precompute quantities such as $\mathbf{K}^{-1}\mathbf{W}$ and efficiently represent them in volumetric wavelet bases for efficient storage, fast summation and matrix element extraction.

## 5.3 Multiresolution Constraint Generation

In this section we briefly address the problem of determining what constraints are to be imposed on multiresolution models during a simulation.

Currently our ARTDEFO simulator supports two modes for constraint generation. The first mode has been used to model displacement constraints for unilateral contact with rigid objects, and this uses a node-based collision and CMA constraint generation approach. Hierarchical constraints are easily supported given the nodal definition of contact. This works well provided contact zones are sufficiently well resolved by the constraint resolution so that significant interpenetration does not occur.

The second constraint generation mode supports spatially localized contacts using pressure masks, and currently this is used for force feedback rendering of point-like contacts. For hierarchical constraints, the distribution of contact forces is not determined by the barycentric coordinate of contact within a single triangle, but by the parametric barycentric coordinate within a triangular constraint patch at the appropriate constraint resolution. Triangular constraint parameterizations are illustrated in Figure 3.5 for the dragon model.

# Chapter 6

# Generation of Green's Functions

Two significantly different approaches exist for constructing GF models. The first approach is to use numerical methods for linear elasticity to precompute the required GFs, while the second approach [PvdDJ$^+$01] is based on the direct measurement of real physical objects.

## 6.1 Numerical Precomputation of Green's Functions

Precomputation of discrete GFs is a straightforward application of standard direct or iterative "black box" BVP solution techniques for any suitable discretization technique, e.g., FEM, BEM, FDM, FVM. For example, finite element formulations with direct and iterative solvers were used in [BC96, CDA99], and direct BEM solution in [JP99a]. For very large models, direct approaches eventually become less appealing and fast preconditioned iterative matrix solvers, such as multigrid [Hac85], can be used. However, because very many, e.g., $\mathcal{O}(n)$, GFs may be precomputed, factorization costs for direct solution methods are offset by the large number of solves the factorizations are used for, whereas iterative methods typically become competitive only for very large models in which direct matrix factorizations are infeasible, e.g., due to memory limitations.

### 6.1.1 Direct Solution Using BEM

Our implementation is based on the Boundary Element Method (BEM) [BTW84, JP99a] for discretizing boundary integral equation descriptions of Navier's equation of homogeneous, isotropic linear elastostatics. BEM provides an appealing and direct approach, since only the boundary geometry must be considered, and the $\mathsf{A}$ and $\bar{\mathsf{A}}$ matrices correspond to the columns of the BEM's $\mathsf{G}$ and $\mathsf{H}$ matrices. The fact that many GFs are computed makes direct matrix factorization approaches quite effective relative to iterative approaches even for moderately sized problems, e.g., $n = 2500$; once the LU decomposition of $\mathsf{A}$ is constructed it may be reused to solve (in parallel) for every single GF[1]. In practice it is only necessary to store the one large dense matrix (A or its LU factors) in memory at once, and the size of this matrix is the one practical limitation of the method. Despite the relatively common comments in the literature deriding BEMs dense matrix, this is one application in which the dense factorization is well used. Precomputation times are presented in §7.6, and are for example significantly faster than those presented in [CDA99].

### 6.1.2 Nonoverlapping Block Preconditioned Multiresolution BEM Iterative Solver

Due to the limitations of the direct BEM solver for very large models, a more sophisticated solver is required. For this purpose, we have implemented an iterative boundary integral equation solver for constructing (hierarchical) GFs.

Very recently, progress has been made on generalizing the fast multipole method (FMM) for Laplace's equation to 3D elastostatics [PN95, FKR$^+$98, YNK01], and the work of Nishimura et al. [YNK01] is particularly promising. However, after a preliminary investigation and partial software implementation, this approach was aborted because it appeared that solving for hundreds, let alone thousands, of GFs would be impractical[2]. While the FMM achieved optimal $\mathcal{O}(n)$ memory and $\mathcal{O}(n(\log n)^\alpha)$ $(\alpha \geq 0)$ CPU complexity, the CPU constant was too large for our precomputation needs.

Instead, a relatively simple, engineering accuracy, fast iterative solver was implemented by computing the unlifted wavelet coefficients of the columns of the BEM's $\mathsf{G}$ and $\mathsf{H}$ matrices to a specified level of accuracy. This resulted in sparse matrix representations, so that fast summation, for $\mathsf{A}$ and $\bar{\mathsf{A}}$, was possible using the unlifted inverse FWT. For a given iterative solver (discussed below) a preconditioner is highly effective here, and we used a nonoverlapping block preconditioner [NH97] based on an adaptive octtree partitioning of nodes. Matrix $\mathsf{A}$ influences are assembled between nodes contained within each octtree leaf cell, and these square blocks are inverted and used as a diagonal block preconditioner. Preliminary investigation suggested that overlapping block preconditioners did perform slightly better [NKLW94], however the minor improvement did not merit the additional construction cost. It makes sense to construct a relatively large preconditioner for this application since larger blocks can improve convergence rates, and the construction

[1] Similarly, building a sparse Cholesky factorization for FEM models would likely have led to a more efficient solver than the condensation solver used in [BC96] or the conjugate gradient solver in [CDA99], especially considering the modest sizes of liver models considered.

[2] The borehole geometry example in [TKN99] (and similar problems in [YNK01]) required more than 5 minutes per matrix multiply for 30000 DOFs in Fortran 77 on a DEC Alpha 21164 (600MHz). At this cost, the comparable $L = 3$ dragon model's hierarchical GF calculation (assuming average number of preconditioned GMRES multiplies was 500 (based on experiment), and 123 block GFs or 369 solves) would require *over six hundred days* for iterative solver matrix multiplication alone!

cost is amortized over multiple GF solves. Regarding the choice of iterative method, preliminary experiments with Matlab suggested that GMRES [SS86] required fewer iterations than other available methods, e.g., BiCGSTAB, QMR, CGS, and with the block preconditioner, GMRES without restarts was most effective; this was consistent with the choice of [YNK01]. Further large model precomputation details are discussed in §7.6.2.

An interesting practical point is that iterative solvers for BEM discretizations on our Loop subdivision surfaces may be exaggerating iteration counts due to problems introduced by the steep mesh grading at extraordinary vertices [KKP91], e.g., valence 3. This problem might be reduced by the preconditioner used, but we plan to investigate this issue in the future.

### 6.1.3 Hierarchical GF

Fast iterative solution methods can reduce the cost of computing a GF from a large models, while using hierarchical GFs for coarse constraint scales can reduce the total number of GFs that must be precomputed. Instead of computing coarse scale GFs from the GF refinement relations (3.78), here it is desireable to iteratively solve (3.83), e.g., see the dragon model in §7. This provides an attractive precomputation alternative that is particularly suited to interactive applications such as games which are unlikely to require tens of thousands of constraint DOFs (see Figure 3.9). This approach could also be used for modeling to provide support for semi-interactive modification of the model. Hierarchical GFs also provide a practical mechanism for computing GFs at very fine resolutions to ensure numerical convergence.

## 6.2 Reality Based Deformation Modeling

A promising alternative to numerical GF precomputation is the active robotic measurement and estimation of GFs corresponding to real deformable objects. The boundary-only input-output description provided by linear elastostatic GFs is a convenient and natural model to estimate. By applying known contact forces to the surface of an object, such as the stuffed toy tiger in Figure 6.1, it is possible to estimate the associated displacement of the free surface using computer vision techniques, while the fixed (bottom) portion of the surface is assumed to have zero displacements specified. From a GF matrix perspective, this corresponds to applying nonzero tractions nodes to, and measuring displacements of nodes in $\Lambda_p^0$. By applying spanning tractions to enough surface nodes, it is possible to estimate the corresponding free surface GF block, $\Xi_{\Lambda_p^0,\Lambda_p^0}$ (see Figure 3.4). We have discussed this approach in [PvdDJ+01], and greater detail can be found in the Ph.D. thesis of Jochen Lang[3] [Lan01]

[3]Also advised by Dr. Dinesh Pai.

whose research addressed the robotic scanning and estimation of deformable models using the UBC Active Measurement (ACME) Facility [PLLW99] (shown in Figure 6.2).

Aside from a very brief overview of the modeling process, we illustrate how multiresolution techniques may be exploited during the GF measurement, estimation, and simulation phases. As with numerical precomputation of multiresolution models, the first stage of the process is the construction of a multiresolution surface mesh, and this is described in §6.2.1.

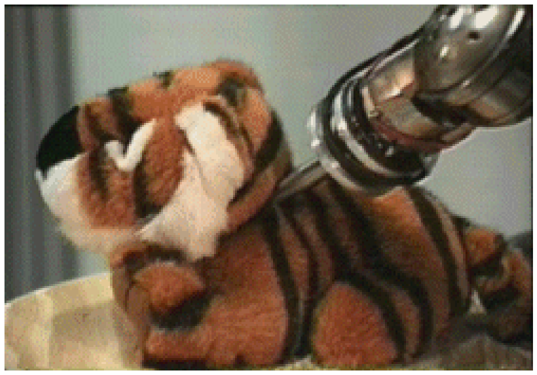


Figure 6.1: *Robotic measurement of deformable objects:* A stuffed tiger toy is being inspected by the force probe attached to ACME's Puma 260 robotic arm. The robot systematically contacts vertices of an associated multiresolution triangle mesh in order to measure displacement responses for related GFs. Stereo vision is used to measure surface movement during contact events.

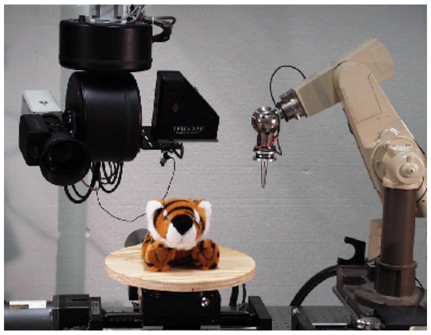


Figure 6.2: *ACME Facility Overview:* The UBC Active Measurement (ACME) Facility is a highly automated robotic measurement facility consisting of a variety of sensors and actuators, all under computer control.

Once a reality based GF model has been acquired it can be interactively simulated (with force feedback) using the previously mentioned CMA framework. Frames from an interative simulation using our ARTDEFO simulation software are shown in Figure 6.3. Point-like contact force responses are computed using vertex pressure masks (§4.2). In general the results are quite satisfactory, capturing non-local effects such as the movement of the head when the back of the tiger is poked. The model does show some of the limitations of the linear model structure for large input displacements, with somewhat exaggerated deformations. However, for moderate input displacements (approximately $<$ 15% of the tiger's diameter), the scanned model behaves quite realistically.

### 6.2.1 Multiresolution Mesh Construction and GF Measurement

Before deformation measurements are acquired, a geometric representation of the object is first scanned and later used to register GF data. It is convenient to acquire a semi-regular mesh with subdivision connectivity in order to use multiresolution techniques for measurement, data processing, and simulation. For this reason, an initial triangle mesh is first acquired using standard techniques described in [PvdDJ$^+$01], and then this mesh is reparameterized using normal mesh algorithms described in §3.2.6. For rendering purposes, extra geometric detail is mapped on to the model using a displaced subdivision surface [LMH00] approach, as described in 3.7.3. During robotic measurement, the multiresolution mesh structure is used when the robot probes surface locations corresponding to vertices of the geometric model at the desired reconstruction resolution, $l$. Multilevel GF interpolation techniques are then used to approximately predict some odd vertex GFs in $\mathcal{M}(l+1)$ for improved rendering quality (described in §6.2.3). Images of the multiresolution tiger model, and contact sampling patterns are shown in Figure 6.4. Because measured displacement fields and the applied force distributions may involve different spatial scales, this multiresolution scanning process is inherently related to hierarchical GFs and surface pressure masks.

### 6.2.2 Scattered Displacement Data Reconstruction

After measuring the displacement field components of $\Xi_{\Lambda_p^0 \Lambda_p^0}$, it is common that several matrix elements are unestimated due to missing observations of the deformed surface (shown in Figure 6.5). This problem can be minimized by obtaining more measurements but not entirely avoided. We use scattered data reconstruction to fill in elements for each column individually. One approach is to interpolate missing displacements by solving Laplace's equation over the set of unestimated vertices. The result of this interpolation process is shown in Figure 6.5.

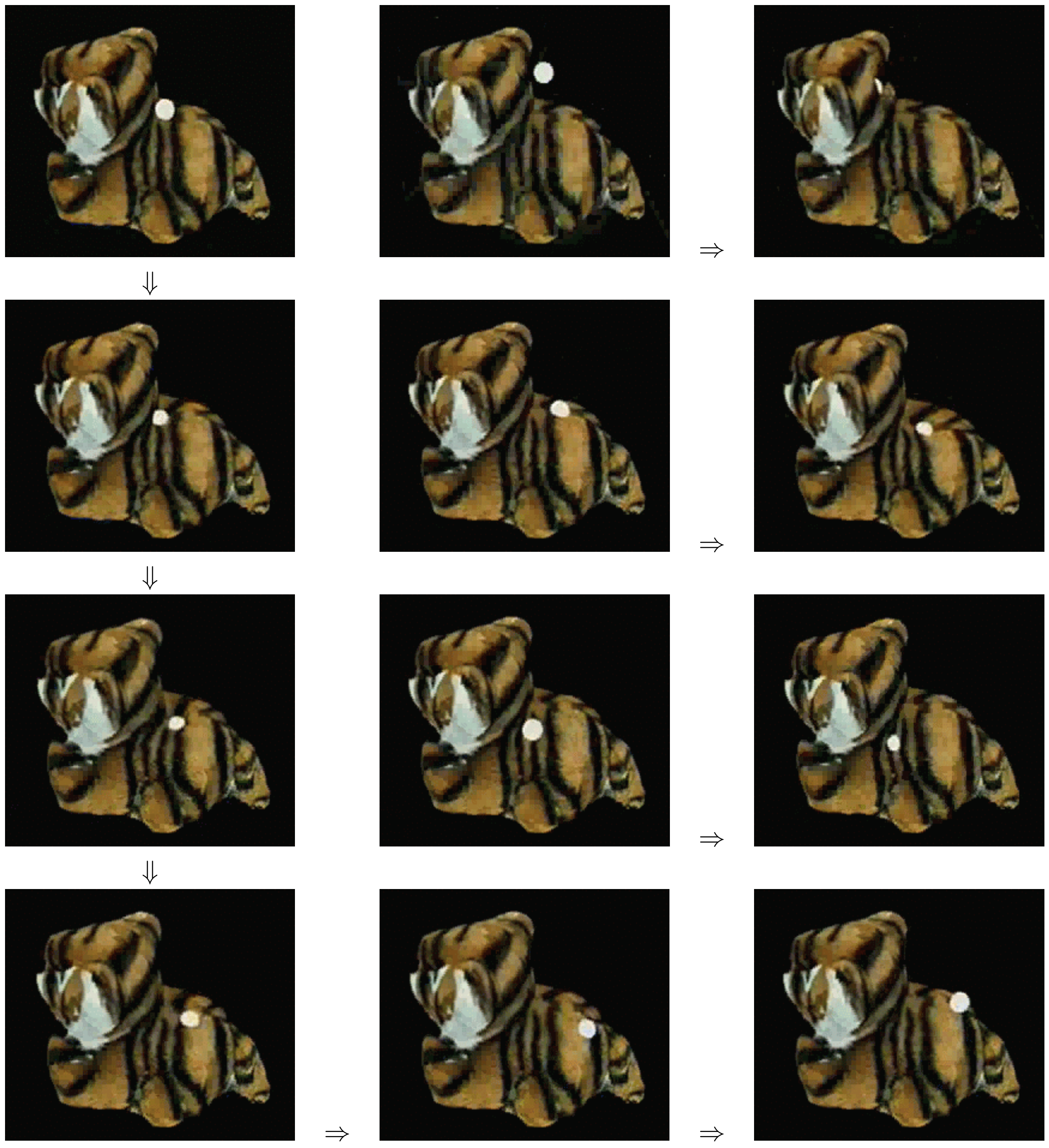


Figure 6.3: *Interactive force feedback simulation of the reality-based tiger model:* Four image sequences are shown (3 with 2 images, and one with 6) with the flow of time indicated by arrows connecting related frames. (Toy tiger model scanned by Jochen Lang [Lan01].)

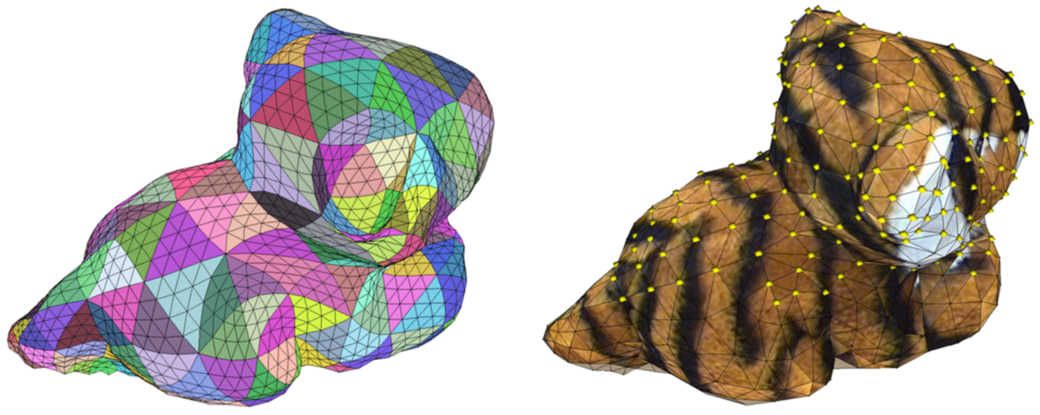


Figure 6.4: *Multiresolution Mesh and Contact Sampling Pattern:* (left) Coarse $L = 0$ parameterization of model, used for active contact measurement, displayed on finest $L = 2$ displaced subdivision surface mesh sued for simulation (see Figure 6.3); (right) yellow points drawn on the $L = 1$ resolution mark the nodes at which the system's displacement response to applied tractions was either measured (even vertices) or inferred (odd vertices).

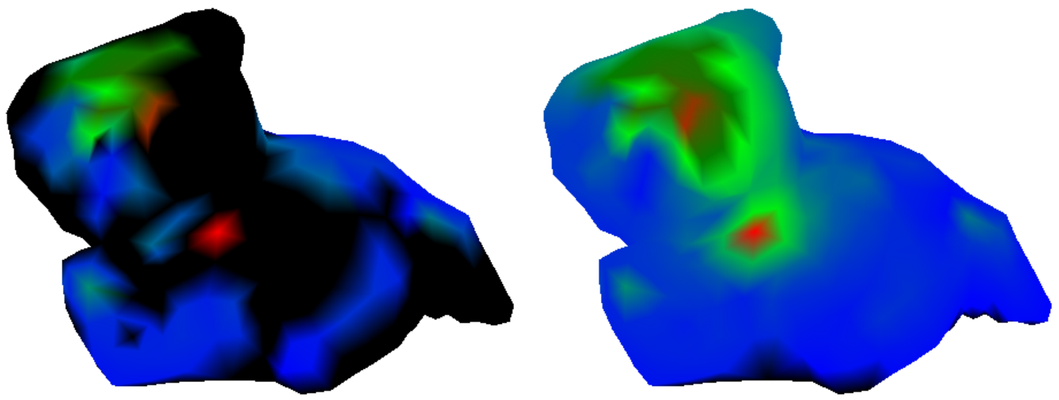


Figure 6.5: *Plots of estimated displacement responses:* (left) Missing observations on the $L = 1$ mesh result in unestimated response components (shown in black); the remaining nodes are color coded with red indicating the greatest displacement and blue the least. (right) These values are estimated by an interpolating reconstruction to obtain the final deformation responses. (Images courtesy of Jochen Lang [Lan01])

### 6.2.3 Green's Function Interpolation

In order to improve rendering quality and reduce measurement and estimation time we exploit the multiresolution mesh structure to optionally infer Green's function responses for unmeasured vertices. This is done by actively poking the model at a resolution $(l-1)$ one level coarser than the resolution $l$ used to estimate displacement fields (illustrated in Figure 6.4). The $k^{th}$ odd vertex on level $l$ has a response $\xi_k$ inferred if both even vertices $(k_1, k_2)$ of its parent edge have responses. If so, the $k^{th}$ response $\xi_k$ is linearly interpolated from the two parent responses, $(\xi_{k_1}, \xi_{k_2})$. The local responses, $\Xi_{kk}$ and $\Xi_{jk}$ when vertex $j$ is a one-ring neighbor of $k$, are handled differently.

Unlike long range displacement influences which are smoothly varying, these local values are associated with a cusp in the displacement field. Simple interpolation for these values is biased and leads to incorrect contact forces during rendering. Instead, the local values are computed as the weighted average of parent responses which have had their local parameterizations smoothly translated from even vertex $k_*$ to odd vertex $k$, e.g., $\Xi_{kk}$ is linearly interpolated from $(\Xi_{k_1k_1}$,$\Xi_{k_2k_2})$ not $(\Xi_{kk_1}$,$\Xi_{kk_2})$. This shifting of the parent's local response before averaging yields a good estimator of the local response at vertex $k$. The resulting displacement field $\Xi_{:k}$ is also linearly independent of $\Xi_{:k_1}$ and $\Xi_{:k_2}$.

Lastly, we note that this linear GF estimator could be used to construct an inter-GF wavelet transform. This would be useful for obtain compression for file storage purposes, and could be combined with wavelet GFs (in next section). Currently, we only benefit from avoiding storage of the unmeasured interpolated GFs.

### 6.2.4 Wavelet Green's Functions

Since it is possible to measure complex models and obtain GFs with high resolution displacement fields, the wavelet GFs presented in this thesis may be used to obtain compression and fast summation benefits. While this was not considered in [PvdDJ$^+$01], it is a straight-forward application of the methods from Chapter 3. Compression results for the reality based tiger model are presented in §7.7.

# Chapter 7

# Results

In addition to the images and examples already presented, several more examples and numerical results are available which confirm the effectiveness of the presented methods.

## 7.1 Note on Numerical Timings

In addition to flop counts, some timings are performed using unoptimized Java code on a single processor Intel Pentium III, 450MHz, 256MB computer with Sun's JDK 1.3 client JVM for Windows 98. Special care was taken to eliminate timing artifacts such as garbage collection, however, results still show several artifacts, e.g., cache effects. Based on hand-coded 3-by-3 blocked dense matrix-vector multiplication (see Figure 7.11), this Java computing environment is rated at 51 MFlops. By using hardware-optimized matrix libraries and current hardware, the performance of the core matrix operations can be dramatically improved[1].

## 7.2 Capacitance Matrix Algorithm Performance

The runtime performance of the basic CMA from §2.2 is shown in Table 7.1 for capacitance matrix LU factorization and substitution, and GF response summation costs. All of these times are substantially smaller than direct solution and precomputation costs (see Table 7.7). Despite their attractive timings, the limitations such as poor scaling in the capacitance matrix factorization/inversion for large $s$ and the GF vector product summation for both $s$ and $n$ are evident, and these are contrasted by improvements in following sections.

[1] While it is difficult to make a general statement regarding Java numerical performance, comparisons with Matlab 6's optimized LAPACK library calls suggest that some of our timings, e.g., of capacitance matrix LU decomposition, may be divided by a factor of approximately 5.

| # Updates, $s$ | LUD Factor (ms) | LUD Solve (ms) | $(\Xi \mathsf{E})(\mathsf{E}^{\mathsf{T}}\bar{\mathsf{v}})$ for $n = 100$ (ms) |
|---|---|---|---|
| 10 | 0.54 | 0.03 | 0.38 |
| 20 | 2.7 | 0.15 | 0.74 |
| 40 | 19 | 0.58 | 1.7 |
| 100 | 310 | 5.7 | 5.7 |

Table 7.1: *Timings of CMA Suboperations* such as LU decomposition and back-substitution of the capacitance matrix, as well as the weighted summation of $s$ GFs are shown for different constraint sizes, $s$.

## 7.3 Sequential Capacitance Matrix Inverse Updating

Large reductions in capacitance matrix inversion costs for sequential temporally coherent BVPs are possible with the sequential updating algorithm from §2.3. Tables 7.2, 7.3, and 7.4 show the typical costs required to perform various node addition and deletion operations in order to update various starting inverses to compute a particular solver. Each table shows that sequential updating is an attractive means for computing inverses in the presence of temporal coherence. The theoretical predictions for updating breakeven in the special cases of node addition (2.75,2.76) or deletion (2.104) are apparent in the numerical results. For small changes in updated node sets it is faster than computing LU decompositions as well as inverse matrices, with greatest benefits being obtained for large capacitance matrices. Using updating it is very efficient to maintain capacitance matrix inverse matrices for simulation.

We have successfully integrated the CMA with sequential updating in simulations. For example, it was used in the interactively simulated grasping task illustrated in Figure 4.1 corresponding to the LEGFM from Figure 7.18(a). While new capacitance matrices are not encountered at each frame of the simulation, when new inverses are required, the speedup obtained using sequential updating directly results in simulation frame rate speedups.

| $s_2 = 20$ | **Delete, $s_-$** | | | | |
|---|---|---|---|---|---|
| **Add, $s_+$** | 0 | 1 | 2 | 5 | 10 |
| 0 | - | 1.1 | 2.7 | (9.9) | [27] |
| 1 | 0.9 | 2.2 | (4.2) | (11) | [29] |
| 2 | 1.9 | (3.5) | (5.5) | [13] | [32] |
| 5 | (4.7) | (6.5) | (8.7) | [16] | [36] |
| 10 | (7.8) | (9.9) | [12] | [21] | [43] |

Table 7.2: *Times to update capacitance matrix inverses:* Times (in milliseconds) to compute a single inverse of size $s_2 = 20$ using different starting BVP inverses of size $s_1 = s_2 + s_- - s_+$. For comparison, times worse than the 2.7 ms required for LU factorization of the 60-by-60 matrix are shown in round brackets, whereas those 4 times worse (11 ms for LU Inverse) are shown in square brackets.

| $s_2 = 40$ | **Delete,** $s_-$ | | | | |
|---|---|---|---|---|---|
| **Add,** $s_+$ | 0 | 1 | 2 | 5 | 10 |
| 0 | - | 6.1 | 12 | (32) | [77] |
| 1 | 5.5 | 11 | 17 | (38) | [84] |
| 2 | 11 | 16 | (22) | (43) | [90] |
| 5 | (23) | (29) | (35) | (57) | [107] |
| 10 | (40) | (47) | (54) | [78] | [129] |
| 20 | (63) | (70) | (78) | [106] | [166] |

Table 7.3: *Times to update capacitance matrix inverses:* Times (in milliseconds) to compute a single inverse of size $s_2 = 40$ using different starting BVP inverses of size $s_1 = s_2 + s_- - s_+$. For comparison, times worse than the 19 ms required for LU factorization of the 120-by-120 matrix are shown in round brackets, whereas those 4 times worse (76 ms for LU Inverse) are shown in square brackets.

| $s_2 = 100$ | **Delete,** $s_-$ | | | | | |
|---|---|---|---|---|---|---|
| **Add,** $s_+$ | 0 | 1 | 2 | 5 | 10 | 20 |
| 0 | - | 37 | 70 | 174 | (374) | (865) |
| 1 | 36 | 68 | 101 | 206 | (410) | (908) |
| 2 | 67 | 98 | 131 | 243 | (440) | (941) |
| 5 | 154 | 186 | 220 | (329) | (536) | (1045) |
| 10 | 286 | (322) | (358) | (469) | (684) | (1212) |
| 20 | (520) | (555) | (595) | (712) | (940) | [1410] |

Table 7.4: *Times to update capacitance matrix inverses:* Times (in milliseconds) to compute a single inverse of size $s_2 = 100$ using different starting BVP inverses of size $s_1 = s_2 + s_- - s_+$. For comparison, times worse than the 310 ms required for LU factorization of the 300-by-300 matrix are shown in round brackets, whereas those 4 times worse (1240 ms for LU Inverse) are shown in square brackets.

## 7.4 Multiresolution Enhancements

This section describes results related to wavelet GF compression and the related fast summation speedup, as well as compressions achieved for hierarchical GFs. All multiresolution analysis is performed on the $\Lambda_p^0$ domain, and GF compression is concerned with the $\Xi_{\Lambda_p^0 \Lambda_p^0}$ GF self-effect block, since it is of greatest practical importance in simulations. As a reminder, this GF block describes surface displacments on $\Lambda_p^0$ due to tractions applied to $\Lambda_p^0$.

Several models have been analyzed and are described in Table 7.5. A fair estimate[2] of the number of tetrahedra in corresponding uniform tetrahedralizations are also stated.

| Model | Tetra | Face | Vertex,$n$ | \|Domain\| | $\|\mathcal{M}(l)\|$ | MB |
|---|---|---|---|---|---|---|
| Rabbit 2 | 872 | 320 | 162 | 133 | (9,25,99) | .64 |
| Rabbit 3 | 6903 | 1280 | 642 | 537 | (9,26,101,401) | 10 |
| Rabbit 4 | 54475 | 5120 | 2562 | 2145 | (9,26,100,404,1606) | 166 |
| Dragon 3 | 176702 | 19840 | 9920 | 7953 | (123,372,1495,5963) | 2277 |
| Finger 2 | 976 | 416 | 210 | 129 | (6,23,100) | .60 |
| Finger 3 | 7829 | 1664 | 834 | 545 | (6,23,100,416) | 11 |
| Tiger 1 | 5751 | 1176 | 590 | 509 | (126,383) | 9.3 |

Table 7.5: *Properties of models used in multiresolution experiments:* rabbit, dragon, fingertip and reality-based tiger models. Columns are provided for the number of triangles and vertices on the boundary, an estimate of the number of tetrahedra for a uniform tetrahedralization, and the size of the $\Lambda_p^0$ domain along with its partitioned level structure on which the wavelet GFs are analyzed. For comparison, the last column indicates the memory size (in MB) of the otherwise uncompressed dense $\Xi_{\Lambda_p^0\Lambda_p^0}$ matrix of 32-bit floats.

### 7.4.1 Wavelet GF Compression and Error Examples

This section shows that substantial GF compression can be obtained at the cost of introducing very practical levels of approximation error. The practical consequence is that *specifying the level of simulation error allows the speedup of our interactive simulations to be directly controlled*, and this is extremely useful for real time applications.

#### Measures of Error

For a given level of compression, we give two measures of the error in the reconstructed GF matrix block $\hat{\Xi}_{\Lambda_p^0\Lambda_p^0}$ relative to the exact value $\Xi_{\Lambda_p^0\Lambda_p^0}$. The first error estimate is based on the relative Frobenius (or Euclidean) norm of the error, here called the "RMS" error:

$$RMS = \frac{\|\hat{\Xi}_{\Lambda_p^0\Lambda_p^0} - \Xi_{\Lambda_p^0\Lambda_p^0}\|_F}{\|\Xi_{\Lambda_p^0\Lambda_p^0}\|_F}, \tag{7.1}$$

[2] Tetrahedra counts are based on dividing the volume of the model, $V$, by the volume of a regular tetrahedron,$V_{tet}$, with triangle face area equal to the mesh's mean face area, $a$:

$$V_{tet} = \frac{192^{\frac{1}{4}}}{9} a^{\frac{3}{2}} \quad \Rightarrow \quad \#\text{Tetrahedra} \approx \lceil \frac{V}{0.4126 a^{\frac{3}{2}}} \rceil.$$

and is a robust estimate of the average GF matrix element error. The second estimate provides a measure of the maximum relative blockwise error over all GFs, here called the "MAX" error:

$$MAX = \max_{j \in \Lambda_p^0} \frac{\|\mathsf{E}_{\Lambda_p^0}^{\mathsf{T}} \left(\hat{\xi}_j - \xi_j\right) \|_{\infty F}}{\|\mathsf{E}_{\Lambda_p^0}^{\mathsf{T}} \xi_j \|_{\infty F}} \tag{7.2}$$

where $\| \cdot \|_{\infty F}$ is defined in (3.50, p. 50).

**Rabbit Model**

Compression results for the three ($L = 2$), four ($L = 3$) and five ($L = 4$) level rabbit models are shown in Figures 7.2, 7.3 and 7.4, respectively. An image of the compressed GF matrix for the smaller $L = 2$ rabbit model was also shown earlier in Figure 3.6 (p. 58)). In general, the compression results indicate a trend toward greater compression ratios for larger models (this is characterized further in §7.4.2). In order to illustrate the performance benefit of lifting the Linear and Butterfly wavelets, results obtained using the unlifted bases are also shown for reference. To avoid clutter in our plots, the generally less effective unlifted wavelet results are plotted in a lighter color for clarity. Graphical depiction of the errors associated with GF compression are shown in Figure 7.1.

The relationship of relative RMS and MAX errors to the relative thresholding tolerance, $\varepsilon$, for various wavelets, are shown for rabbit models in Figures 7.5 ($L = 2$), 7.6 ($L = 3$) and 7.7 ($L = 4$). Interestingly, the behavior of errors for Linear and Butterfly wavelets are nearly identical for respective lifted and unlifted types. In both cases, the inverse FWT reconstruction process is stable.

**Fingerpad Model**

Compression results for the three ($L = 2$) and four ($L = 3$) level fingertip models are shown combined in Figure 7.8, where we have allowed thresholding of the base level wavelet coefficients. Dependence of the errors on threshold tolerance is shown in Figure 7.9 for the finer model. A useful degree of compression is observed in each case for modest error, e.g., 5-10% MAX error, with dramatic improvements for the $L = 3$ model.

### 7.4.2 Dependence of GF Compression on Model Complexity

To better understand how compression or fast summation speedup rates depend on the $\Lambda_p^0$ domain resolution of a model, the ratio of fast summation speedup factors for models of adjacent resolutions, $(L+1)$ and $L$, are shown in Figure 7.10 as a function of relative RMS error. Given a model with $m$ vertices in its $\Lambda_p^0$ domain, the fast summation speedup factor

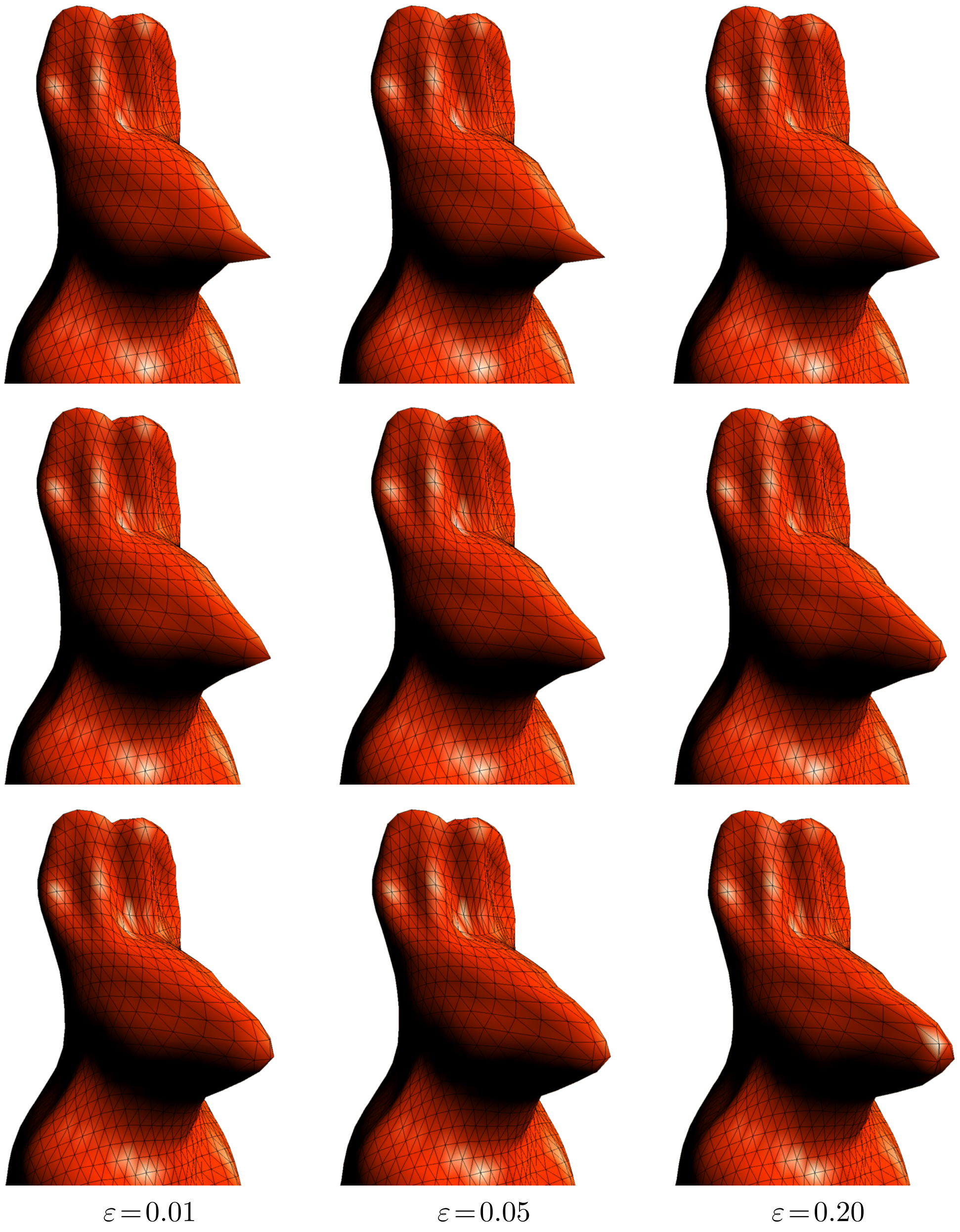
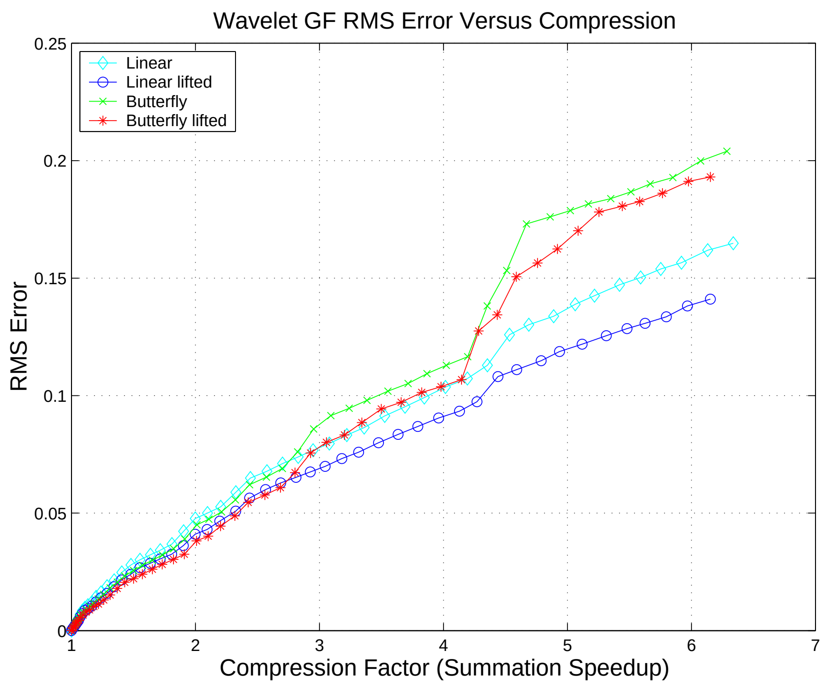


Figure 7.1: *Rabbit model ($L = 4$) approximate wavelet GF reconstructions* for lifted linear wavelets at three thresholds, $\varepsilon = (0.01, 0.05, 0.20)$, corresponding to compression factors of $(8.4, 25, 68)$. Three hierarchical GFs are shown with constraint levels 2 (top row),3 (middle row) and 4 (bottom row), and were computed using the refinement relation from fine scale (level 4) thresholded GFs. Relative errors proportional to the threshold are visible, especially in the neighbourhood of the rabbit's nose where an exaggerated normal displacement constraint has been applied to each model.



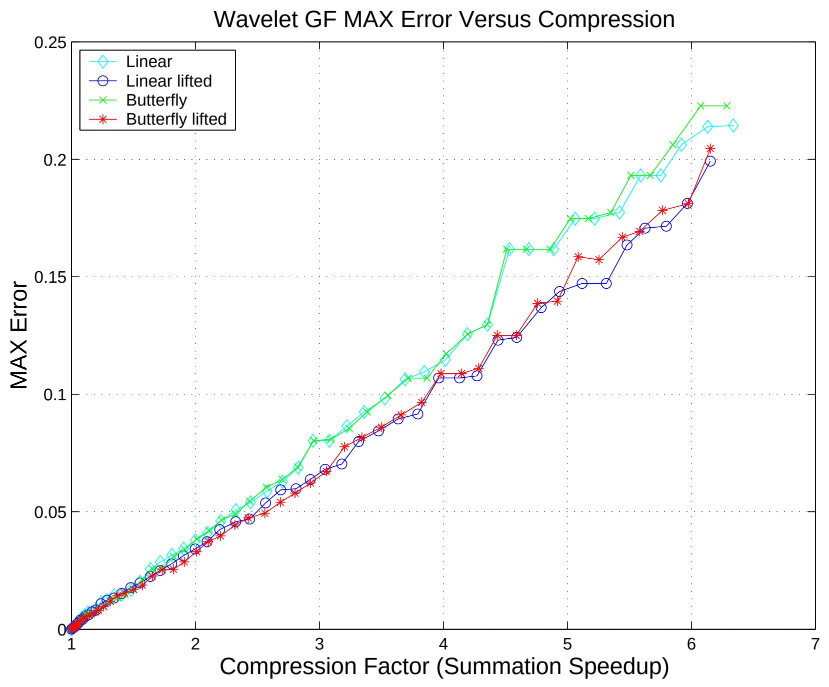
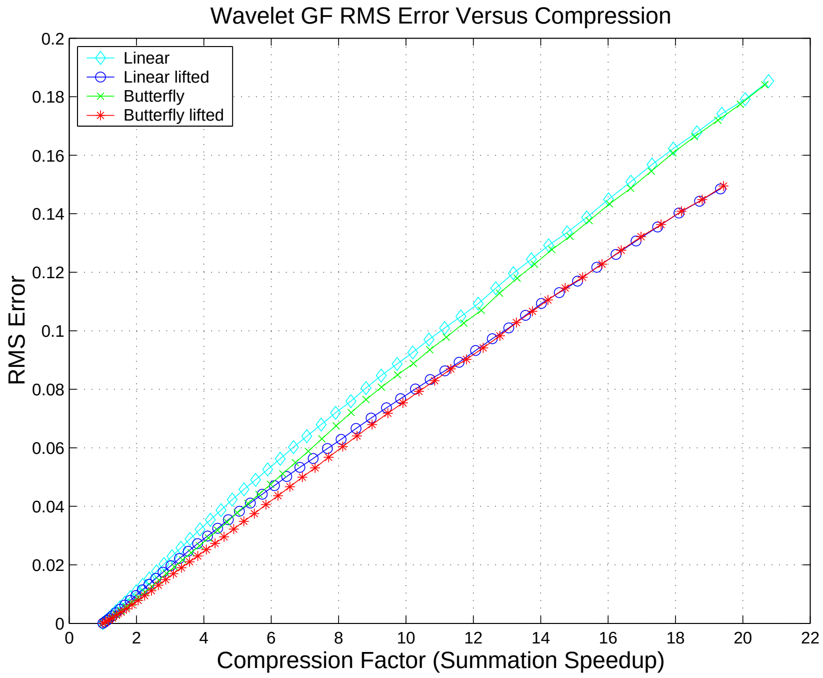


Figure 7.2: *Rabbit model ($L = 2$): Wavelet GF error versus compression*



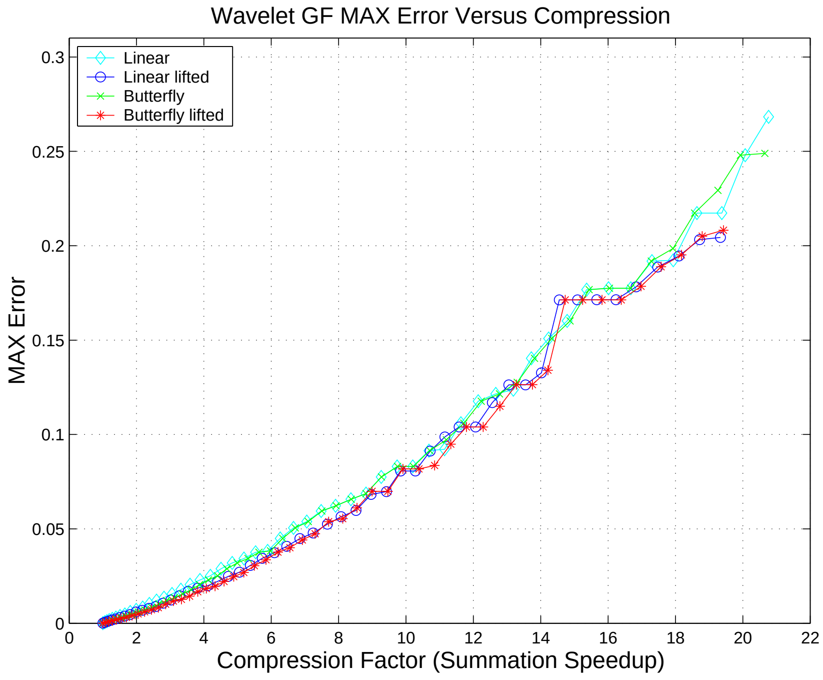
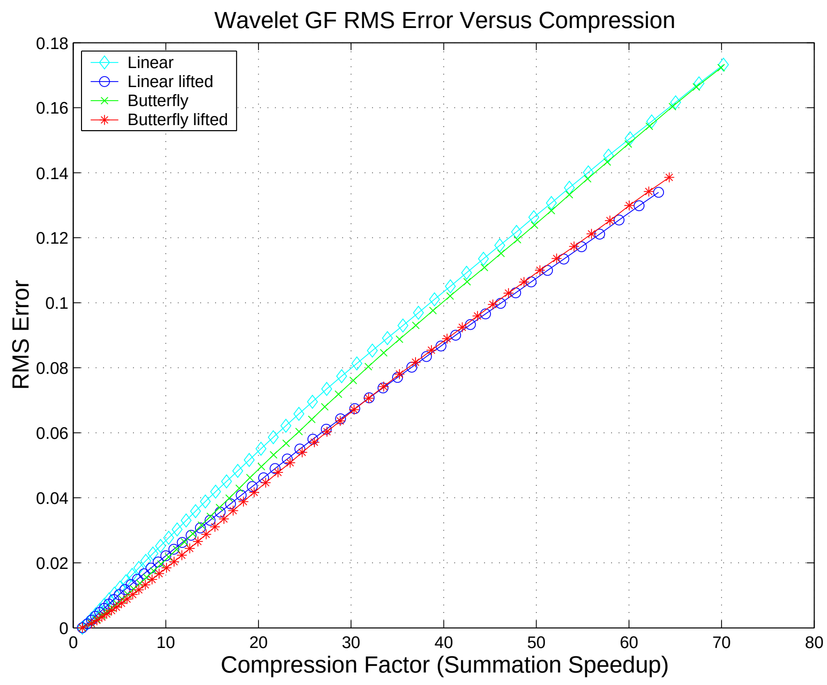


Figure 7.3: *Rabbit model ($L = 3$): Wavelet GF error versus compression*



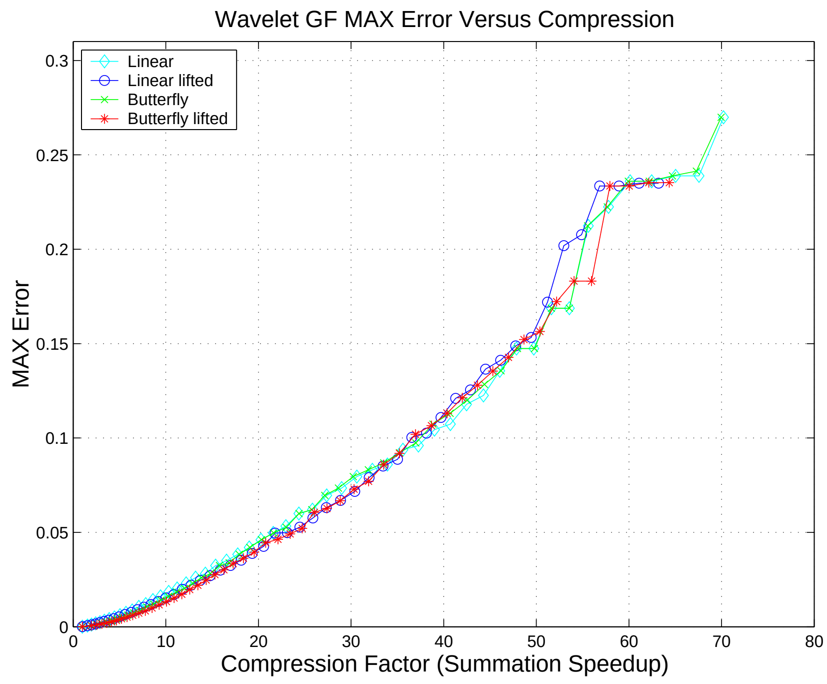
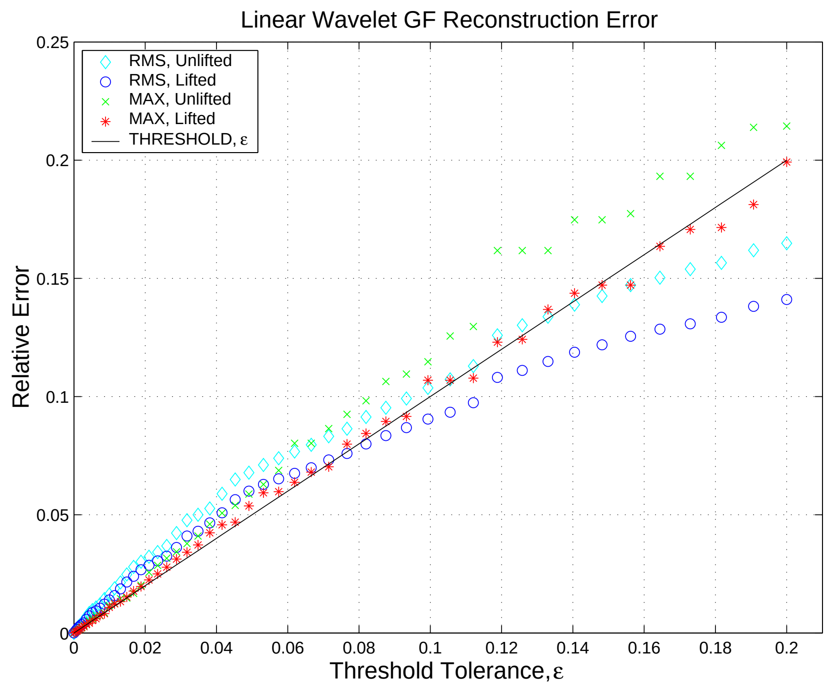


Figure 7.4: *Rabbit model ($L = 4$): Wavelet GF error versus compression*



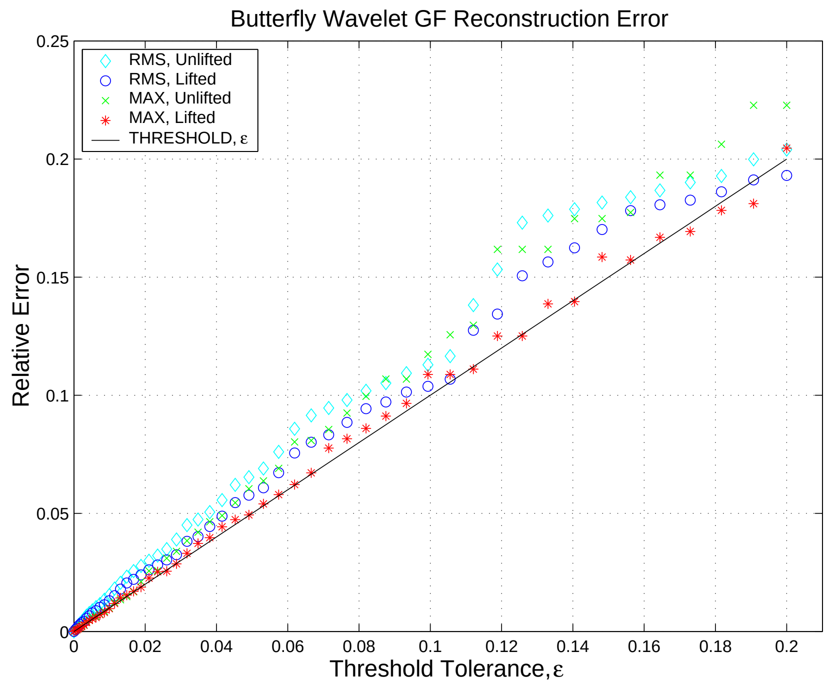
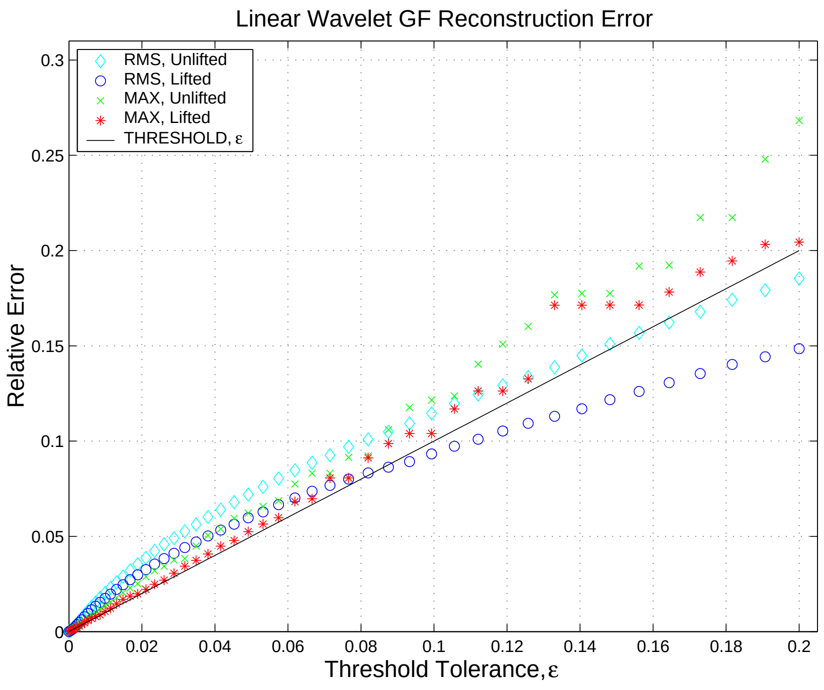


Figure 7.5: [Rabbit model ($L=2$): Wavelet GF error versus thresholding tolerance] *Rabbit model ($L = 2$): Wavelet GF error versus thresholding tolerance:* (Top) Linear wavelets; (Bottom) Butterfly wavelets.



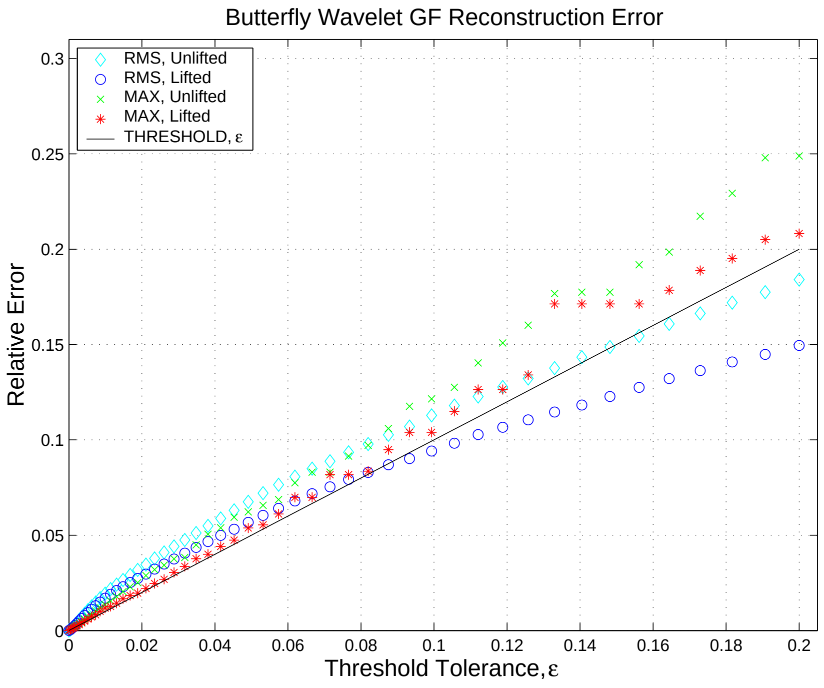
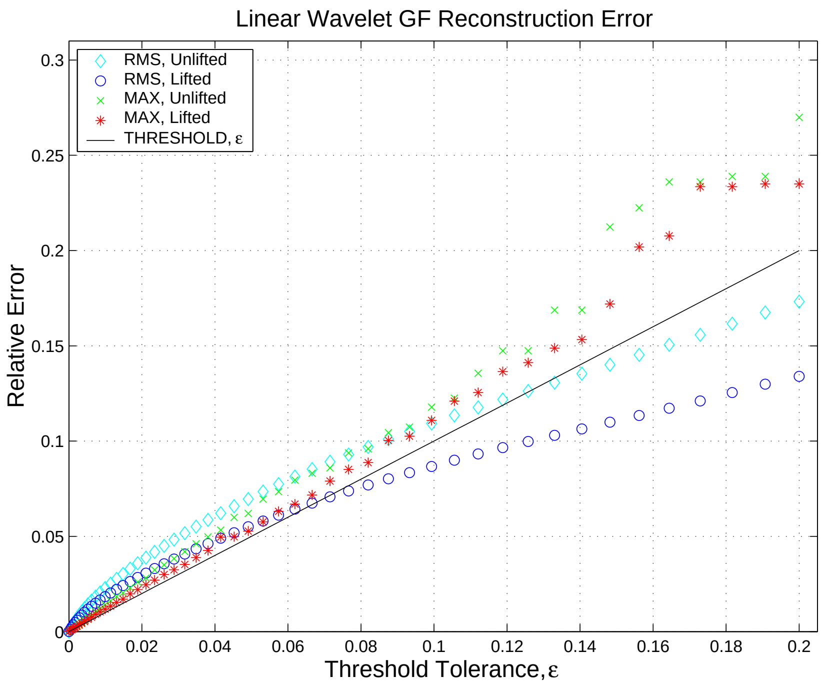


Figure 7.6: *Rabbit model ($L = 3$): Wavelet GF error versus thresholding tolerance:* (Top) Linear wavelets; (Bottom) Butterfly wavelets.



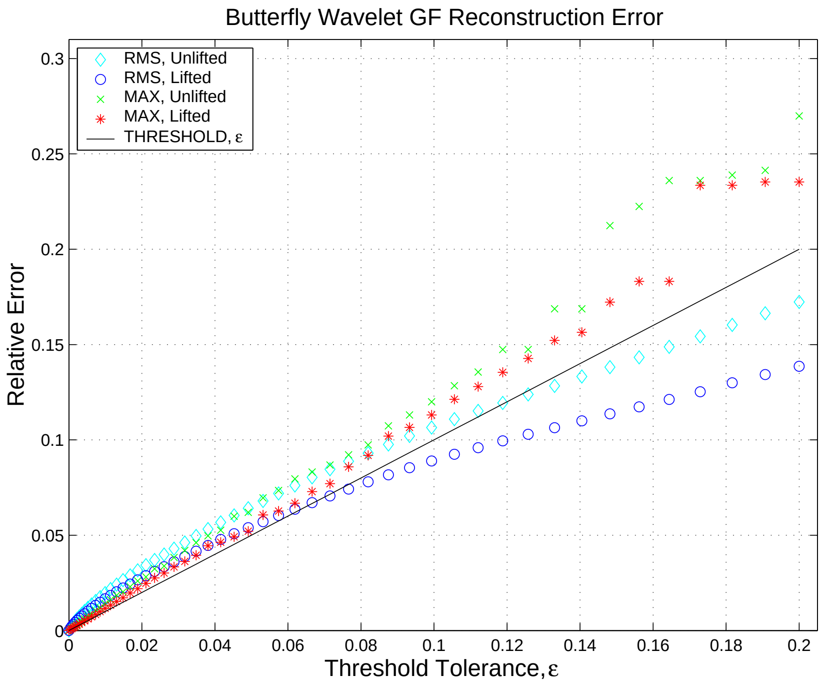
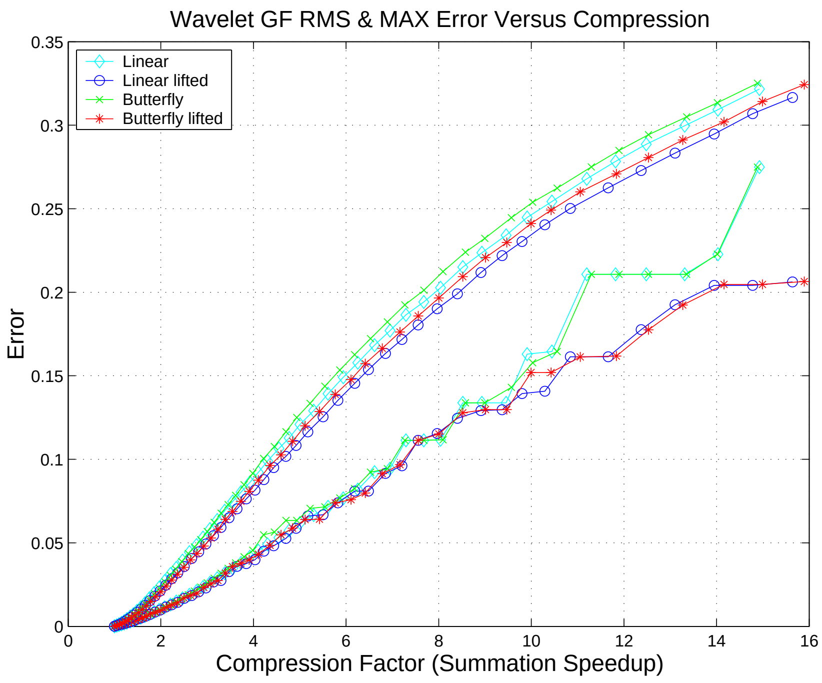


Figure 7.7: *Rabbit model ($L = 4$): Wavelet GF error versus thresholding tolerance:* (Top) Linear wavelets; (Bottom) Butterfly wavelets.



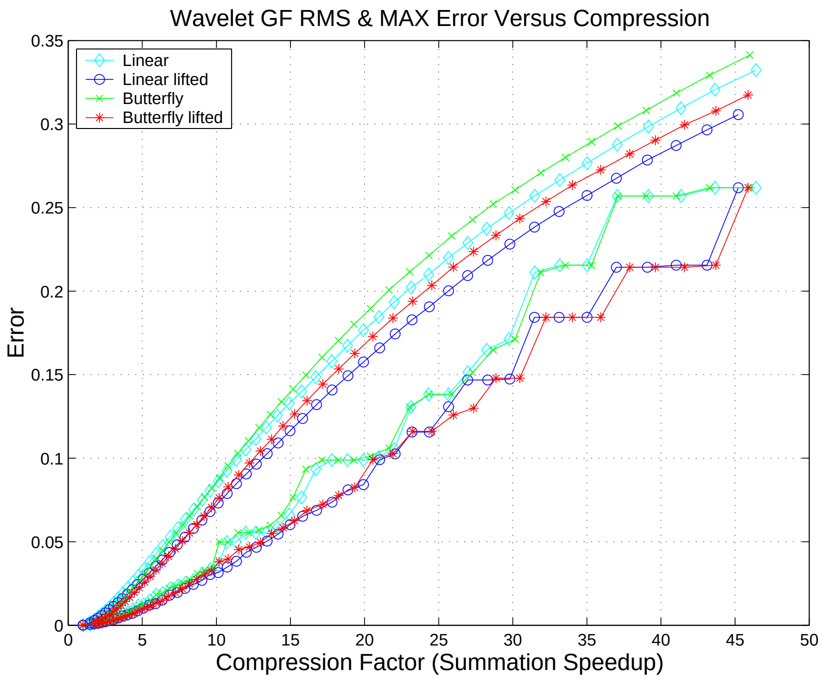
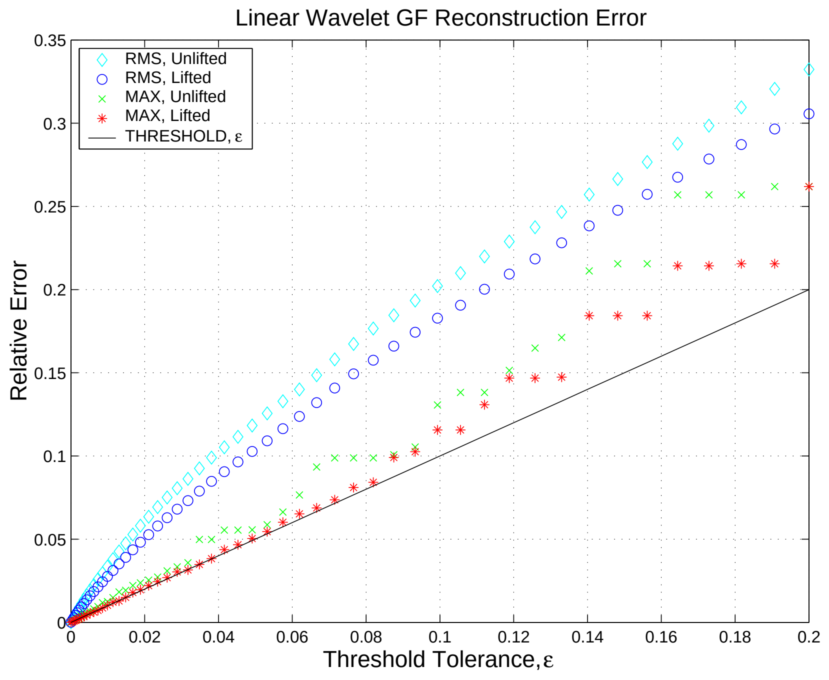


Figure 7.8: *Fingerpad models ($L = 2,3$): Wavelet GF error versus compression:* (Top) $L = 2$ model; (Bottom) $L = 3$ model. In each graph, the upper set of smoother curves are the RMS errors, while the lower set are the MAX errors.



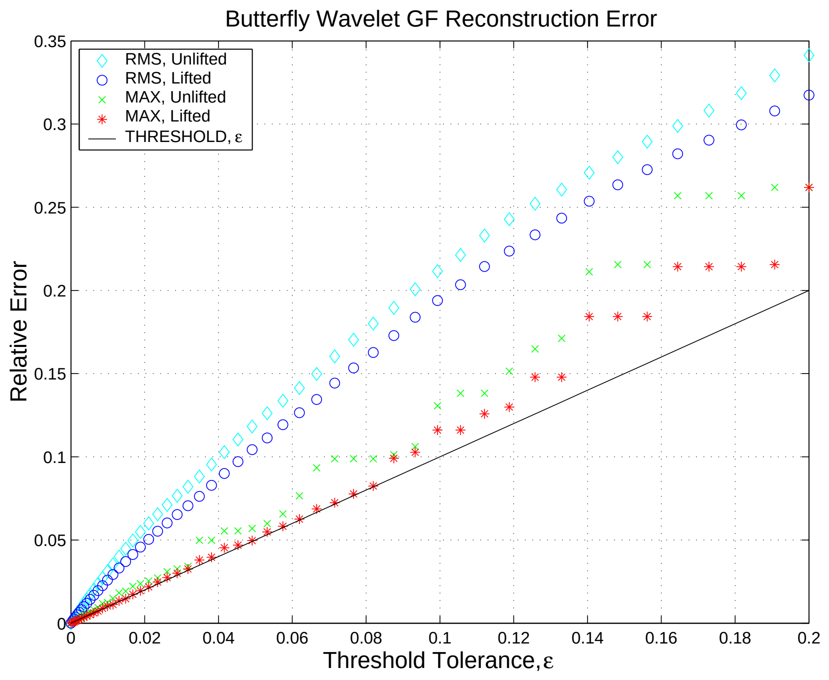


Figure 7.9: *Fingerpad model ($L = 3$): Wavelet GF error versus thresholding tolerance:* (Top) Linear wavelets; (Bottom) Butterfly wavelets.

is defined as the ratio of the number of dense GF elements, $m^2$, to the number of nonzero wavelet GF blocks, $nnz$, or

$$speedup(m) = \frac{m^2}{nnz(m,\varepsilon)}. \tag{7.3}$$

The ratio of speedup for two adjacent levels with $m$ and $4m$ vertices is therefore

$$\frac{speedup(4m)}{speedup(m)} = \frac{(4m)^2}{nnz(4m,\varepsilon)}\frac{nnz(m,\varepsilon)}{m^2} = \frac{16nnz(m,\varepsilon)}{nnz(4m,\varepsilon)}. \tag{7.4}$$

To provide intuition, linear dependence of the number of nonzeros, $nnz(m,\varepsilon)$, on $m$ would yield a ratio of 4, whereas for $nnz(m,\varepsilon) = C_\varepsilon m \log m$ one would obtain

$$\frac{speedup(4m)}{speedup(m)} = \frac{4\log(m)}{\log(4m)} < 4. \tag{7.5}$$

While the limited information in Figure 7.10 does not allow us to confidently estimate the exact dependence of $nnz$ on $m$, it does provide a very useful observation regarding the dependence of the ratio of fast summation speedups on error. It establishes that in practice there is little improvement in relative speedup between resolutions once the RMS error level has increased to a certain level.

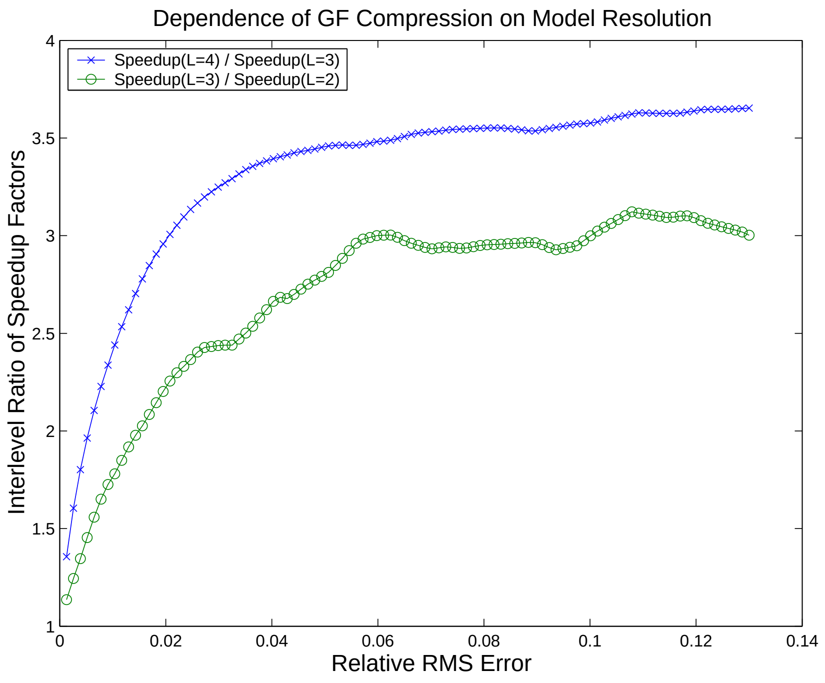


Figure 7.10: *Rabbit model: Dependence of GF compression on model resolution*

### 7.4.3 Verification of Fast Summation Speedup

Fast summation speedups are directly related to the compression achieved using wavelets. Our runtime simulations experienced close to a proportional speed-up, with the inverse lifted Linear wavelet transform being approximately as costly as an extra normal computation. We do not use Butterfly wavelets for our interactive simulation because the negligible compression benefits (if any) do not outweigh the increased cost of the inverse wavelet transform[3]. As the number of constraints increases and GF response summations dominate the graphics simulation cost, *speedups from wavelet fast summation directly translate into speedups for interactive simulations.* For example, consider the ARTDEFO force feedback simulator we have implemented (§7.5). The graphics loop simulation time per frame $t_{frame}$ is equal to the sum of our fast summation cost $t_{sum}$ and other graphics related factors $t_{other}$. For simulations of large-scale models, the simulation frame rate

$$f_{frame} = \frac{1}{t_{sum} + t_{other}} = \frac{1}{t_{sum}} + \mathcal{O}(\frac{t_{other}}{t_{sum}^2}) \tag{7.6}$$

is dominated by the rate of summation. Therefore speedup in fast summation times results in proportional speedup for frame rates. Experimental evidence for the linear dependence of fast summation speedup on GF compression is illustrated in Figure 7.11.

### 7.4.4 Timings of Selective Wavelet Reconstruction Operations, $(\mathsf{E}^{\mathsf{T}}\mathsf{W}^{-1})$

The performance of inverse FWT operations for the extraction of GF block elements (§3.3.3) are shown in Table 7.6 for an unoptimized recursive element reconstruction implementation using linear wavelets. The implementation's reconstruction of each element involves redundant calculation overhead of approximately a factor of two. Nevertheless, these pessimistic times are sufficiently fast for practical use and can be optimized further using the approaches mentioned in §3.3.3.

| # Levels | 2 | 3 | 4 |
|---|---|---|---|
| Time/block, $\mu$sec | 8 | 20 | 36 |

Table 7.6: *Pessimistic timings of selective reconstruction operations* for GF 3-by-3 block element extraction. Block extraction times are listed as a function of the number of resolution levels (# Levels) that must be adaptively reconstructed to obtain the element.

[3]Butterfly subdivision requires averaging of 4 times as many values as Linear, however it can be efficiently implemented to be only twice as costly.

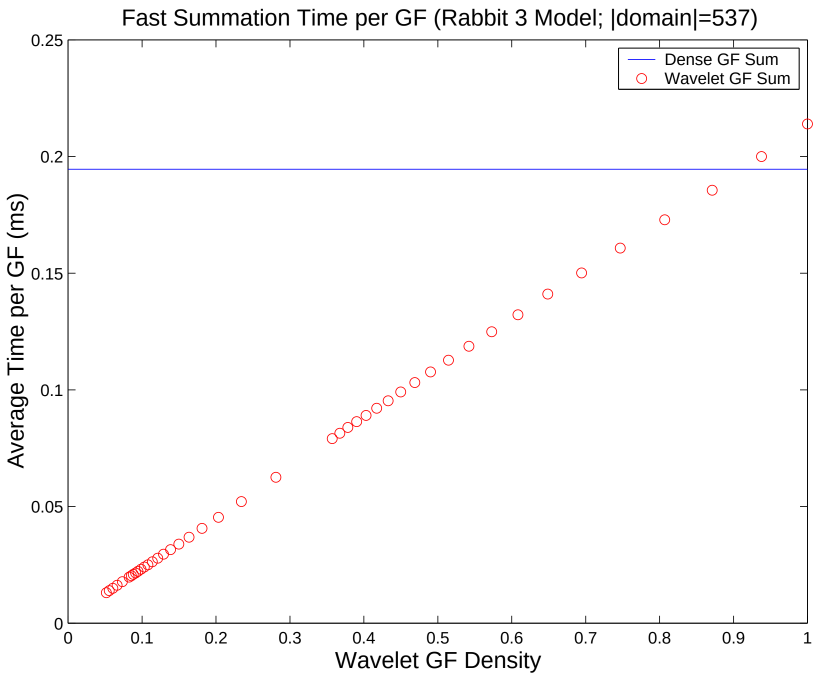


Figure 7.11: *Fast summation cost per GF summed:* Comparison of wavelet GF fast summation timings (in milliseconds) of a rabbit model ($L = 3$, 537 vertex domain) with dense GF matrix multiplication (horizontal line, time=0.19ms/GF) for full matrix multiplication. The linear dependence on nonzero GF matrix elements confirms the cost analysis of §3.3.5 (equation 3.68, p. 55): fast summation costs are directly proportional to the number of nonzero wavelet GF elements. Timings are for the lifted Linear wavelets, for which the inverse FWT requires 0.38 ms. Based on the 3-by-3 dense matrix multiplication performance (18*537 flop in .19 ms) this Java computing environment is rated at 51 MFlops.

### 7.4.5 Wavelet Compression of Hierarchical Green's Functions

#### Rabbit Model

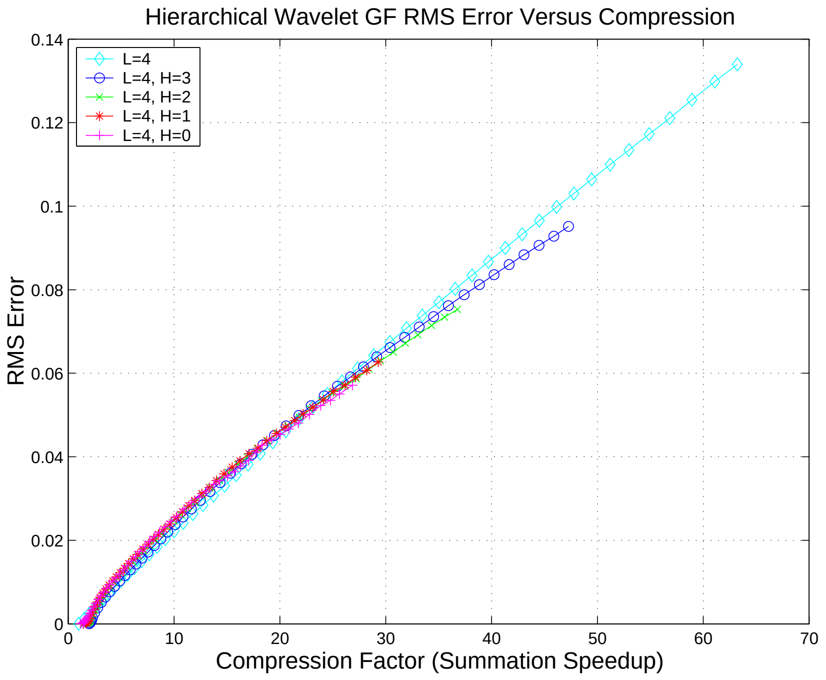
Wavelet compression results are shown in Figure 7.12 for hierarchical GFs corresponding to the rabbit model. Compression behaviors for each level of the GF hierarchy are approximately the same, although the coarser and therefore smoother GF levels result in only slightly better compression for a given RMS error, with this being more apparent for the smoother lifted Butterfly basis. Figure 7.13 displays RMS reconstruction error versus thresholding tolerance for each level of the hierarchy. The approximately equivalent compression rates for GFs across constraint scales implies that a fourfold reduction in constraints per coarsened constraint level results in approximately a fourfold speedup in fast summation for a given level of error.

#### Dragon Model

The four-level dragon ($L = 3$) is our largest model, with 19840 faces, 9920 vertices, and 7953 vertices in the $\Lambda_p^0$ domain partitioned across four levels of sizes $(123, 372, 1495, 5963)$. In order to reduce the precomputation time, we only computed hierarchical GFs at the coarsest ($l=0$) constraint scale (illustrated in Figure 3.5, p. 56), which only required 123 GFs to be precomputed instead of 7953 (discussed further in §7.6, p. 116). The deformations associated with these coarse level constraints are very smooth (shown in Figure 7.16 (p. 115), and for this reason the compression achieved for these GFs is quite good for a given RMS error; compression results are shown in Figure 7.14, and error versus threshold is shown in Figure 7.15.

## 7.5 Force feedback for Point-like Contacts

Our current force feedback implementation is based on the point-like contact approach discussed in §4.2. Forces are rendered by a 3 dof PHANToM™ haptic interface (model 1.0 Premium), on a dual Pentium III computer running Windows 2000. The haptic simulation was implemented in C++, partly using the GHOST© toolkit, and interfaced to our ArtDefo elastostatic object simulation written in Java™and rendered with Java 3D™. Collision detection and the frictional contact problem are entirely computed on the haptic servo loop running at 1 kHz, which enables very high fidelity contact force feedback even for very slow graphical simulations. The haptic loop caches state values which are used to prescribe boundary conditions for the slower graphical simulation, e.g., running at 25–80 Hz. For a point-like contact, it was only necessary to perform collision detection on the undeformed model, so this was done using the GHOST© API. A photograph of the author



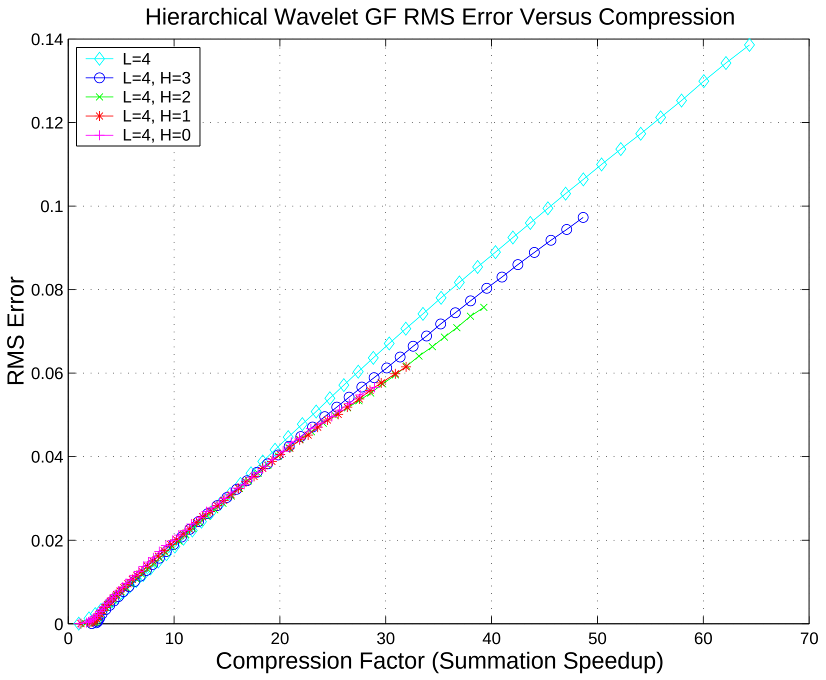
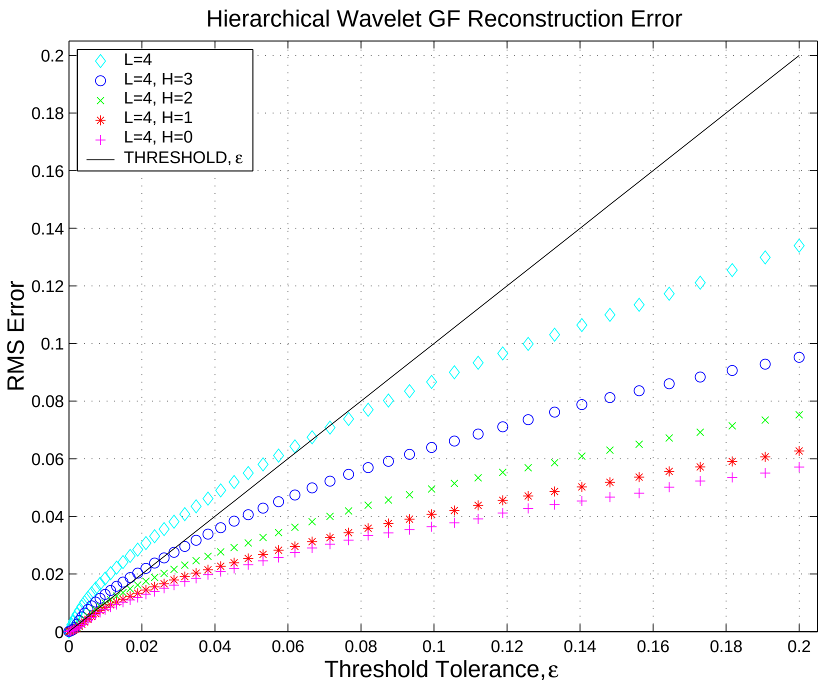


Figure 7.12: *Rabbit model ($L = 4$): Hierarchical wavelet GF error versus compression:* (Top) Lifted Linear; (Bottom) Lifted Butterfly.



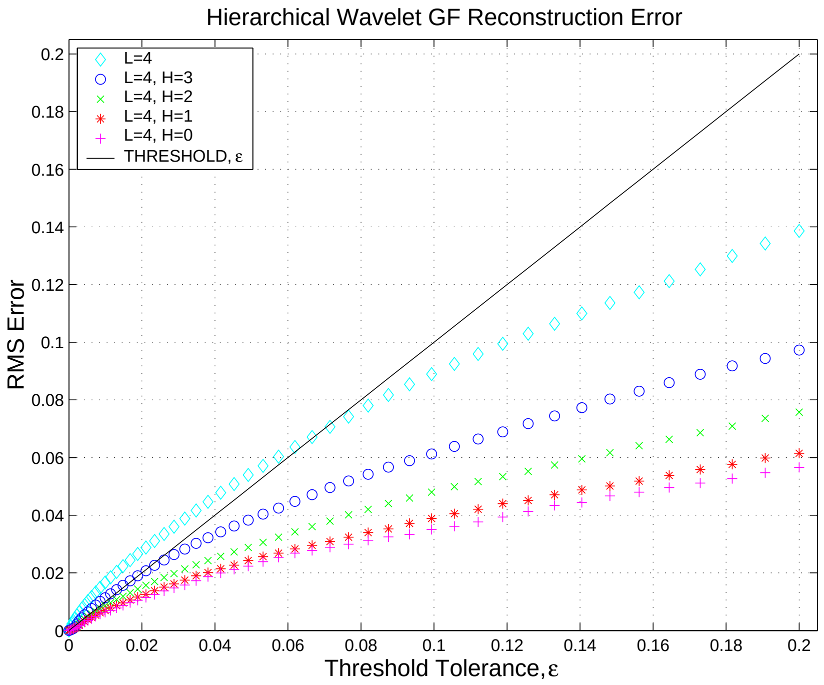
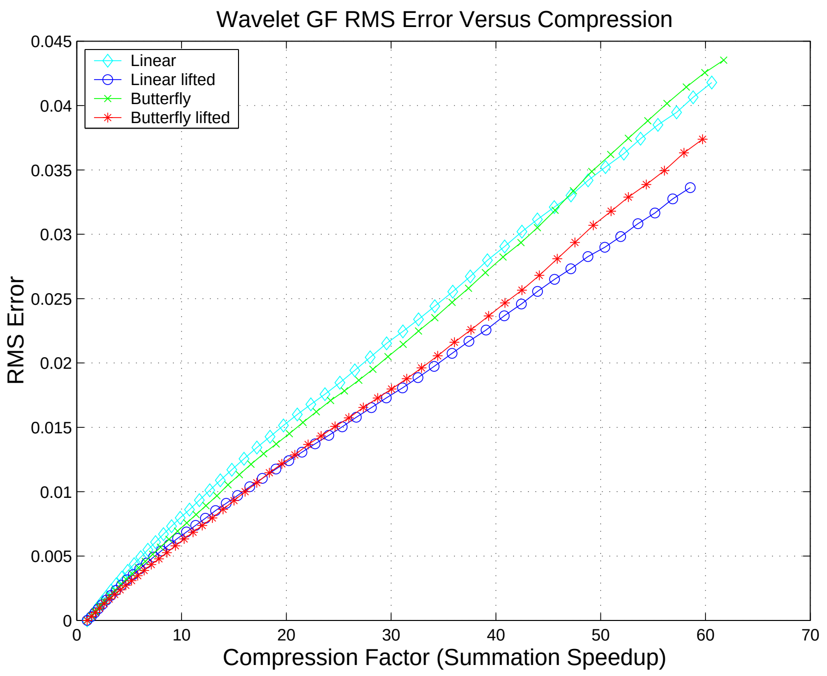


Figure 7.13: *Rabbit model ($L = 4$): Hierarchical wavelet GF error versus thresholding tolerance:* (Top) Lifted Linear; (Bottom) Lifted Butterfly.



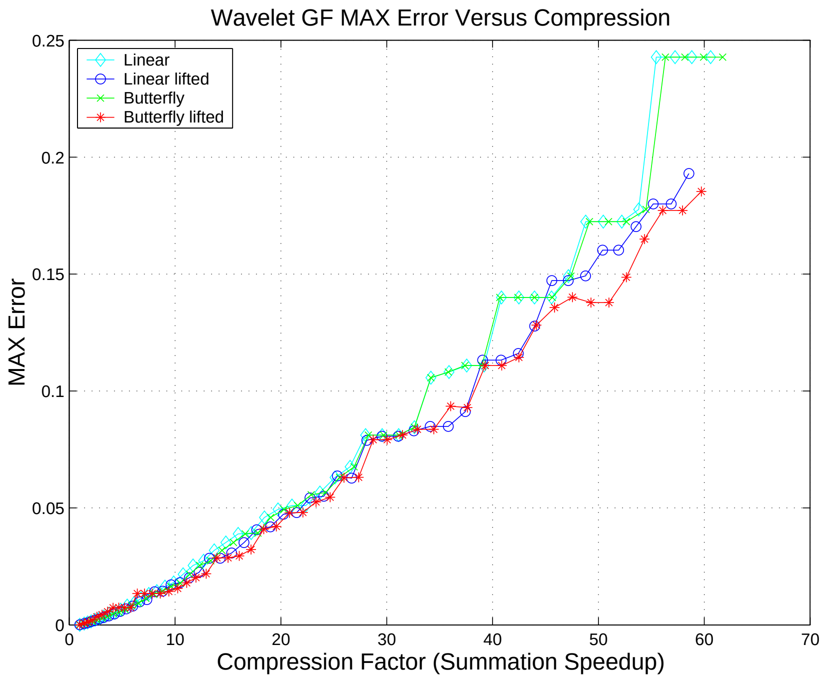
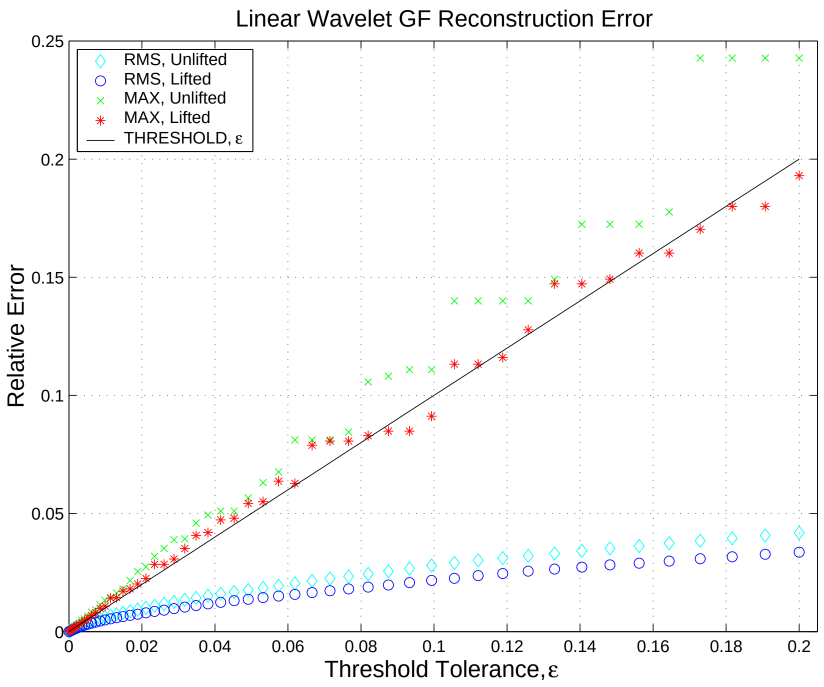


Figure 7.14: *Dragon model ($L = 3$): Hierarchical wavelet GF error versus compression:*



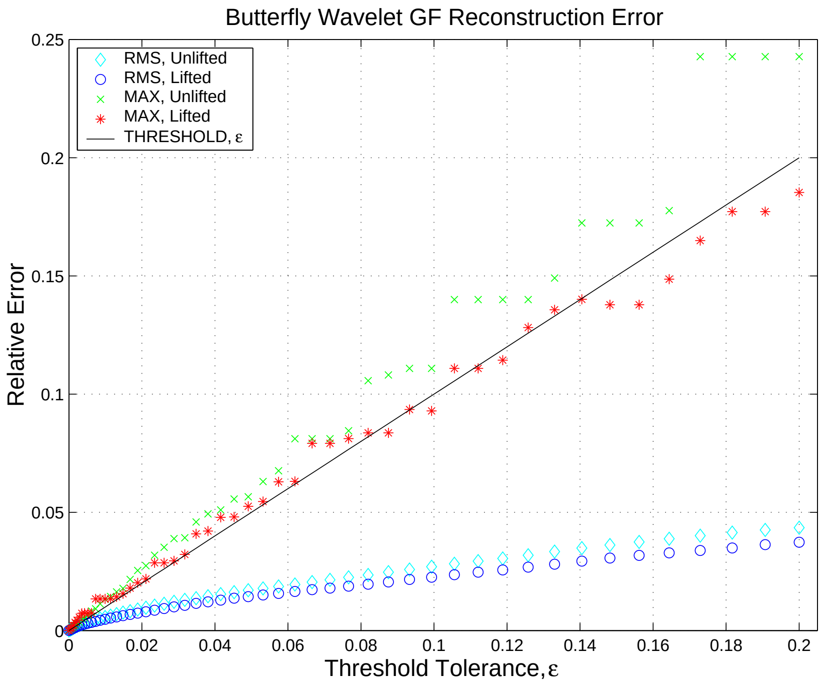


Figure 7.15: *Dragon model ($L = 3$): Hierarchical wavelet GF error versus thresholding tolerance:* (Top) Linear wavelets; (Bottom) Butterfly wavelets.

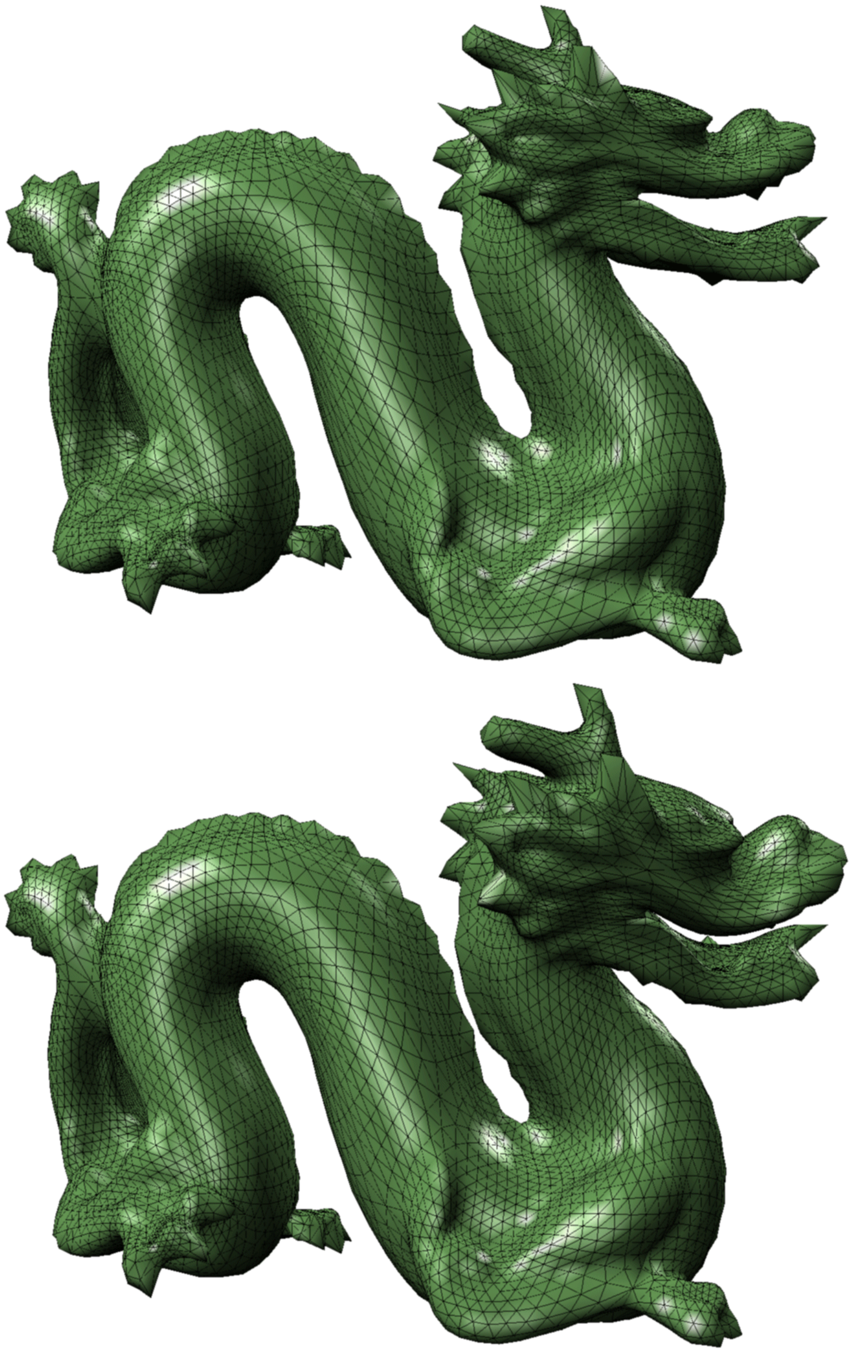


Figure 7.16: *Deformed dragon model* (*$L = 3$*)*:* (Top) Undeformed model; (Bottom) hierarchical GF model deformed due to a downward force applied to top side of head on constraint level 0. This very large model compresses extremely well (approx. factor of 100 at 5% RMS error) and is suitable for interactive force feedback simulation.

demonstrating the simulation is shown in Figure 7.17, and a number of screen shots for various models presented in [JP01] are shown in Figure 7.18.

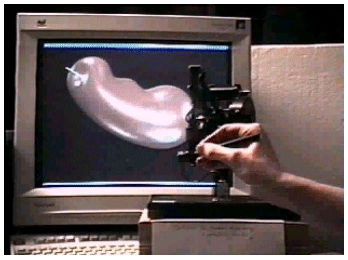


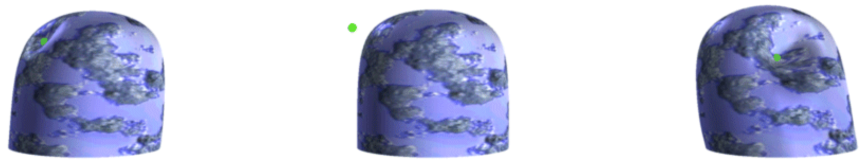
Figure 7.17: *: Photograph of force feedback simulation in use:* Users were able to push, slide and pull on the surface of the model using a point-like manipulandum. Additionally, it was possible to change the surface friction coefficient, as well as the properties of the pressure mask, with noticeable consequences. The PHANToM™ (here model 1.0 Premium) was used in all force feedback simulations, and is clearly visible in the foreground.

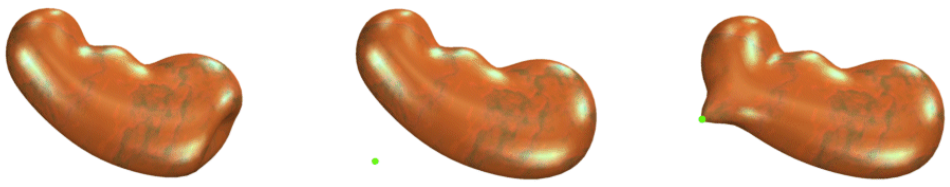
We found vertex pressure masks (from §4.2) produced noticeable improvements in the smoothness of the sliding contact force, especially when passing over regions with irregular triangulations (see Figure 4.5). We have not conducted a formal human study of the effectiveness of our simulation approach. However, the haptic simulation has been demonstrated to hundreds of users at several conferences: the $10^{th}$ Annual PRECARN-IRIS (Institute for Robotics and Intelligent Systems) Conference (Montreal, Quebec, Canada, May 2000, best poster) and in the ACM SIGGRAPH 2000 Exhibition (New Orleans, Louisiana, USA, July 2000). More recently, an invited demonstration of the multiresolution rabbit model was well received at ACM1 (invited demonstration, San Jose, March 2001), and our demonstration of the tiger reality-based model at the $11^{th}$ Annual PRECARN-IRIS Conference (Ottawa, Ontario, Canada, June 2001) won the first prize for technology demonstration. Users reported that the simulation felt realistic. In general, the precomputed LEGFM approach was found to be both stable and robust for real time simulation.

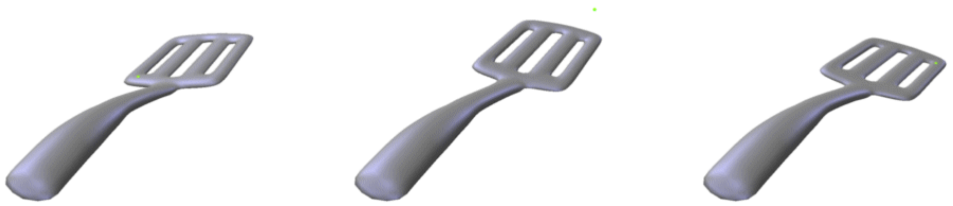
## 7.6 Precomputation Times

### 7.6.1 Direct BEM Solver

Timings for the direct isoparametric linear element BEM precomputation stage are shown in Table 7.7 for several models, including those shown in Figure 7.18. While these times are not excessive, they are several thousand times larger than the simulation times shown for comparison. Since the precomputation phase involves "pleasantly parallel" computations, our BEM solver is multithreaded and capable of computing $\mathsf{H}$ and $\mathsf{G}$ matrix elements, and performing LU back-substitution in parallel; LU factorization could be performed in parallel using block LU decomposition (which maximizes level-3 flops) [GL96], however we have not implemented this. For simplicity, all times given in Table 7.7 are for single processor calculations with the exception of "Rabbit 4" which took approximately 9 hours as an overnight job on an 8-way SMP server (8 Pentium III 450MHz CPUs, 1 GB shared


(a) A simple nodular shape with a fixed base region.


(b) A kidney-shaped model with position constrained vertices on part of the occluded side.


(c) A plastic spatula with a position constrained handle.

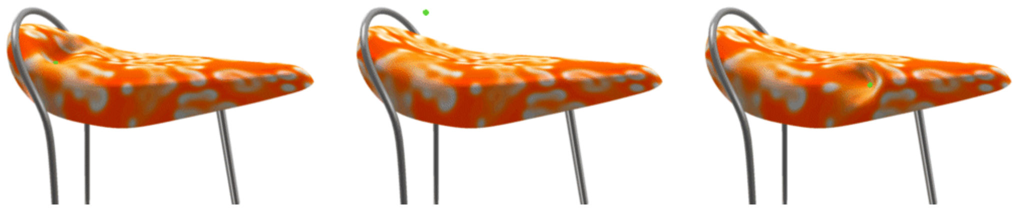

(d) A seemingly gel-filled banana bicycle seat with matching metal supports.

Figure 7.18: *Screenshots from real time haptic simulations:* A wide range of ARTDEFO models are shown subjected to various displacements using the masked point-like contacts of §4.2. For each model, the middle of the three figures is uncontacted by the user's interaction point (a small green ball).

memory, Solaris, Java JDK 1.2.2) with much of the time (about 4 hours) spent performing the uniprocessor LU factorization. Using optimized native code, it may have been possible to precompute the "Rabbit 4" model significantly faster (in approximately 2 hours assuming the speedup of 5 mentioned in footnote on p. 93).

| Model | Tetra | Face | Vertex,$n$ | \|Domain\| | Precomp | LUD % | Sim (ms) |
|---|---|---|---|---|---|---|---|
| Nodule | 820 | 256 | 130 | 89 | 1.1 min | 1 | 0.05 |
| Kidney | 2690 | 640 | 322 | 217 | 7.7 min | 3 | 0.13 |
| Spatula | 1514 | 1248 | 620 | 559 | 45 min | 6 | 0.34 |
| Banana Seat | 3013 | 1088 | 546 | 245 | 25 min | 20 | 0.15 |
| Rabbit 2 | 872 | 320 | 162 | 133 | 1.8 min | 1 | 0.07 |
| Rabbit 3 | 6903 | 1280 | 642 | 537 | 40 min | 9 | 0.33 |
| Rabbit 4 | 54475 | 5120 | 2562 | 2145 | *9 hours | N/A | 1.3 |

Table 7.7: *Green's Function precomputation and simulation times* for BEM models (some depicted in Figure 7.18). All GFs corresponding to moveable free vertices (in $\Lambda_p^{(0)}$) were computed, and the precomputation time (Precomp) of the largest model is less than an hour (see text for "Rabbit 4"). As is typical of BEM computations for models of modest size, the $\mathcal{O}(n^2)$ construction of the matrices (H and G in equation 2.13) is a significant portion of the computation, e.g., relative to the $\mathcal{O}(n^3)$ cost of performing the LU decomposition (LUD %) of the $A$ matrix. The last column indicates that (sub)millisecond graphics-loop computations (Sim) are required to determine the point-like contact deformation response of each model's free boundary for force feedback simulations.

For comparison, published LEGFM precomputation times appear in [CDA99] for FEM models of comparable complexity. A modest volumetric liver model with 6500 tetrahedra and 1400 internal nodes took approximately 8 hours to precompute on a Dec Alpha 400 MHz machine using a preconditioned conjugate gradient iterative method. In comparison, the direct BEM solver precomputation times given in Table 7.7 are quite appealing, e.g., compare "Rabbit 3," and especially since they are computed in Java. This is partly due to the fact that the number of boundary nodes $n$ is smaller than the expected number of volume nodes $\mathcal{O}(n^{\frac{3}{2}})$; for comparison, estimated tetrahedra counts are also given (see footnote on page 96 for calculation).

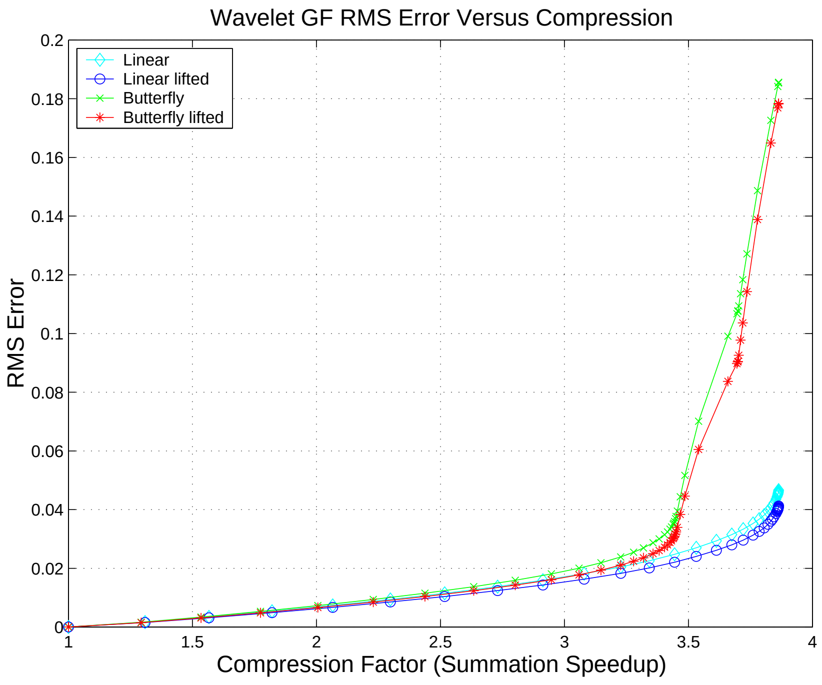
### 7.6.2 Iterative Multiresolution BEM Solver

For larger models and cases where only a few (hierarchical) GFs are desired, the memory and processing overhead associated with constructing large factorizations is onerous. Nevertheless, compared with the iterative solution alternatives, we have found direct methods to be sufficiently good when computing $\mathcal{O}(n)$ GFs, provided that the factorizations fit into main memory. However, in order to construct the $L=3$ dragon model (19840 triangles, 9920 vertices), whose dense linear BEM matrix would be 29760-by-29760 and require over 3.5 GB of RAM, the GMRES iterative solver (see §6.1.2) was used. Using BEM matrices adaptively constructed with unlifted linear wavelet transformed columns ($\varepsilon = 0.04$), and an adaptively partitioned octtree block preconditioner, each $\tilde{A}$ matrix-vector multiply and preconditioner iteration required approximately 11 seconds. The unrestarted GMRES solves used an average of 470 matrix-vector multiplies per solve (to visual tolerance), so that each solve was approximately 90 minutes long, and therefore the 369 multithreaded BVP solves required 23 days of CPU time. This process would benefit from further optimization. Using the ARTDEFO force feedback simulator, the entire 23 days of CPU time can be experienced in just a few seconds (see Figure 7.16, p. 115).

## 7.7 Multiresolution Reality-based Models

GFs estimated from real physical models can also benefit from compressed wavelet representations, although this was not considered by us in [PvdDJ$^+$01]. As an example, consider the sparse representation of the GF matrix for the scanned tiger model shown in Figure 6.4. Unlike the rabbit model, the measured tiger model's subdivision connectivity mesh has only 2 levels, with a relatively large base resolution; the measured surface domain has 509 vertices partitioned into two levels with (126,383) vertices. As a result, limited benefit can be obtained from thresholding when all base-level vertex values are retained; in the best case, only base level elements will remain, i.e., approximately 25% nonzero. This behavior is evident in the compression graph of the related Figures 7.19 and 7.20. By allowing the base resolution to be thresholded, better compression can be obtained, e.g., for a practical 10% RMS error, and this result is shown in Figures 7.21 and 7.22. In all cases, lifted Linear wavelets perform best, with Butterfly wavelets noticeably worse; a partial explanation is that this geometric model is very coarse, so (a) the compactness of the Linear interpolant is more favourable than the larger Butterfly stencil, and (b) the benefit of Butterfly's smooth interpolation can not yet be observed on this scale of the displaced subdivision mesh.

An interesting observation is that maximum relative max norm GF error for lifted Linear wavelet GFs (see Figure 7.22) is significantly less than the thresholding tolerance $\varepsilon$, whereas for BEM models, e.g., "Rabbit 4" in Figure 7.7 (p. 104), the opposite was true except for very small models, e.g., "Rabbit 2" in Figure 7.5 (p. 102). This behavior is likely related to the two level mesh, but perhaps also to the fact that the Laplacian regularized scattered data reconstruction approach (§6.2.2) has introduced GF displacement field smoothing of a less elastic nature.



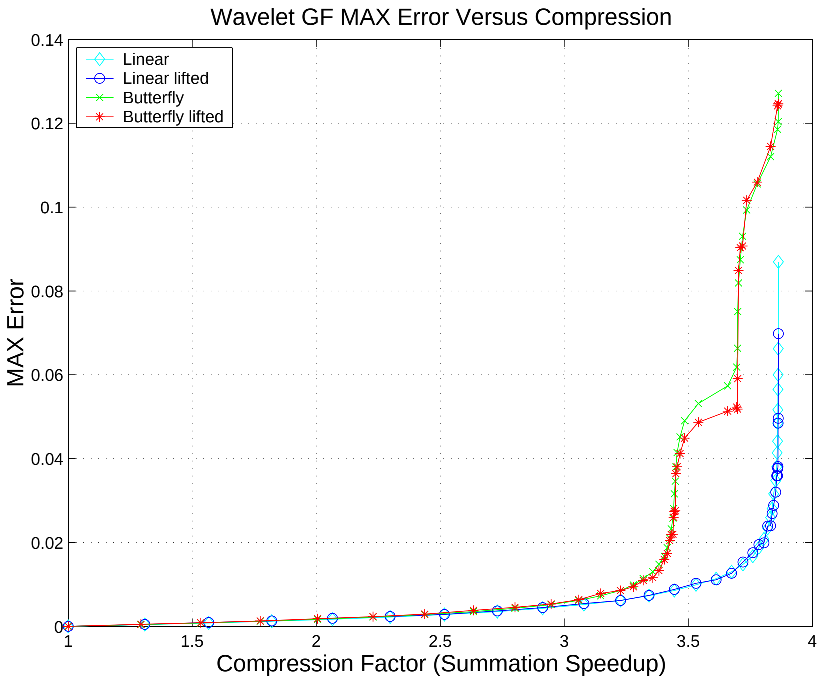
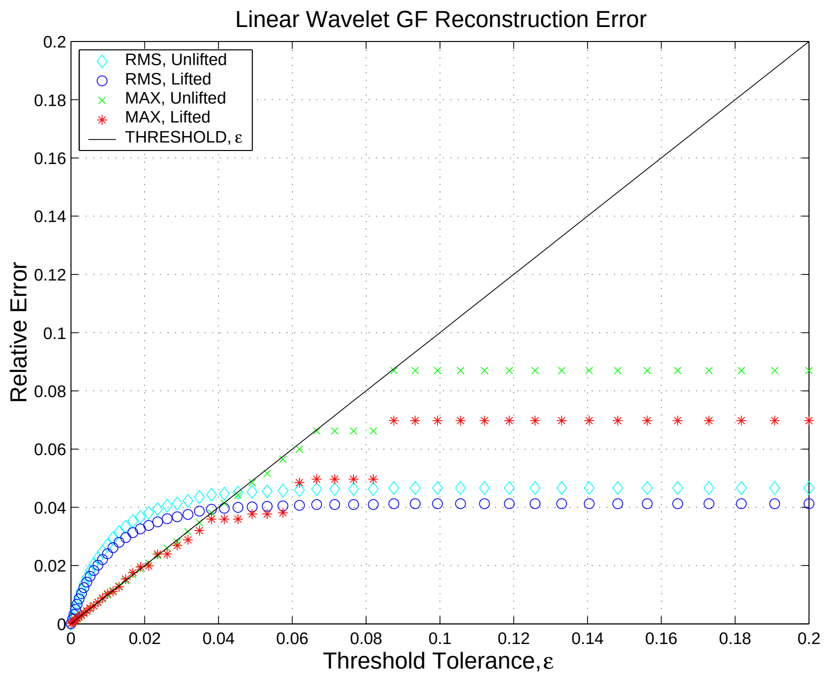


Figure 7.19: *Reality-based tiger model ($L = 1$): Wavelet compression* of two-level model with *unthresholded base resolution.* Compression is limited to below a factor of approximately four. (Top) Relative RMS error; (Bottom) Relative MAX error.



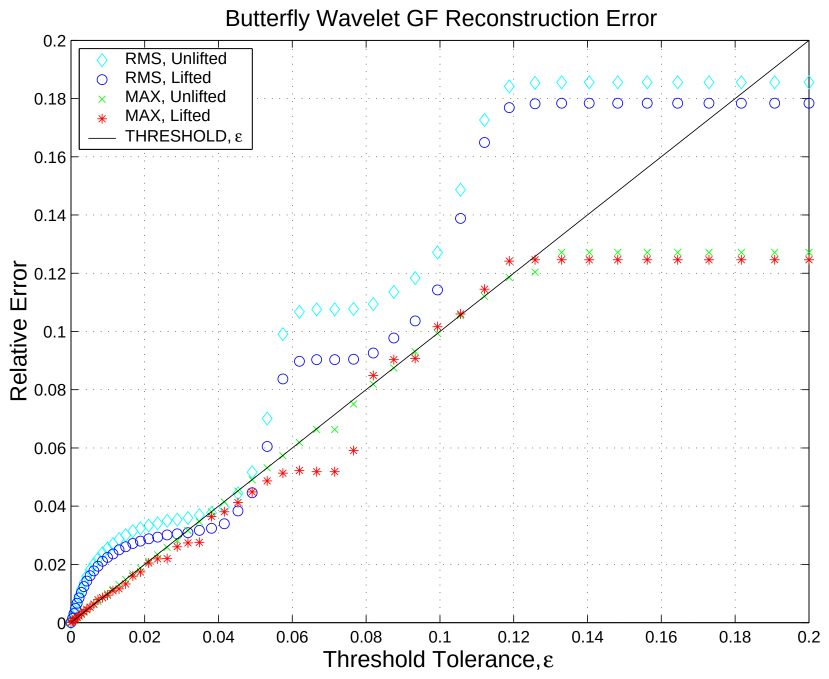
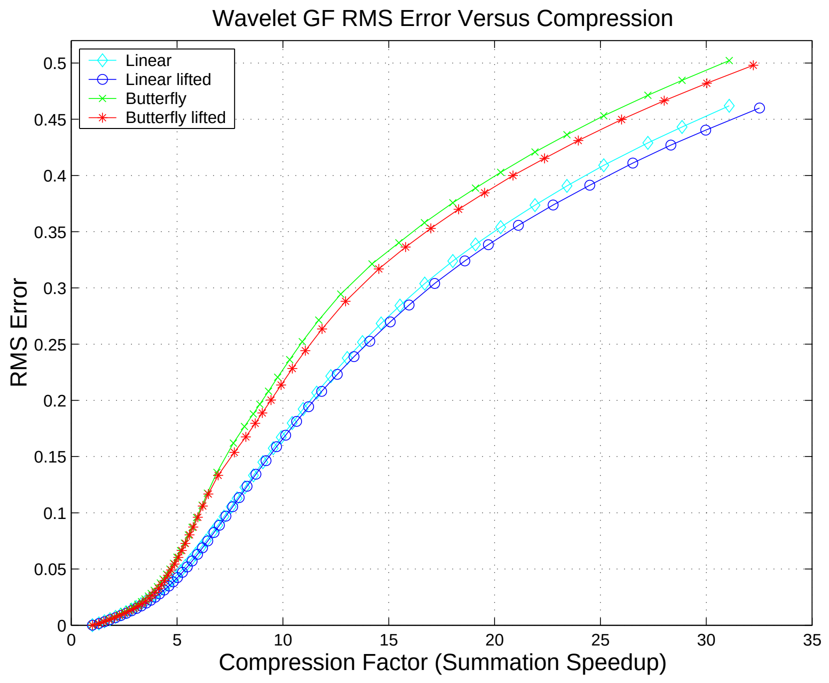


Figure 7.20: *Reality-based tiger model ($L = 1$): Wavelet GF error versus thresholding tolerance* of two-level model with *unthresholded base resolution.* (Top) Linear wavelets; (Bottom) Butterfly wavelets.



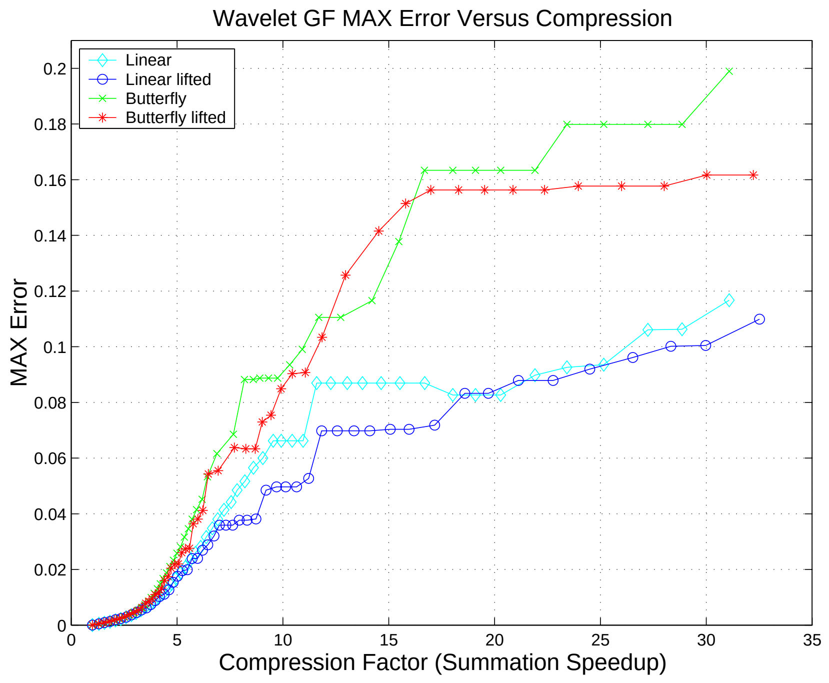
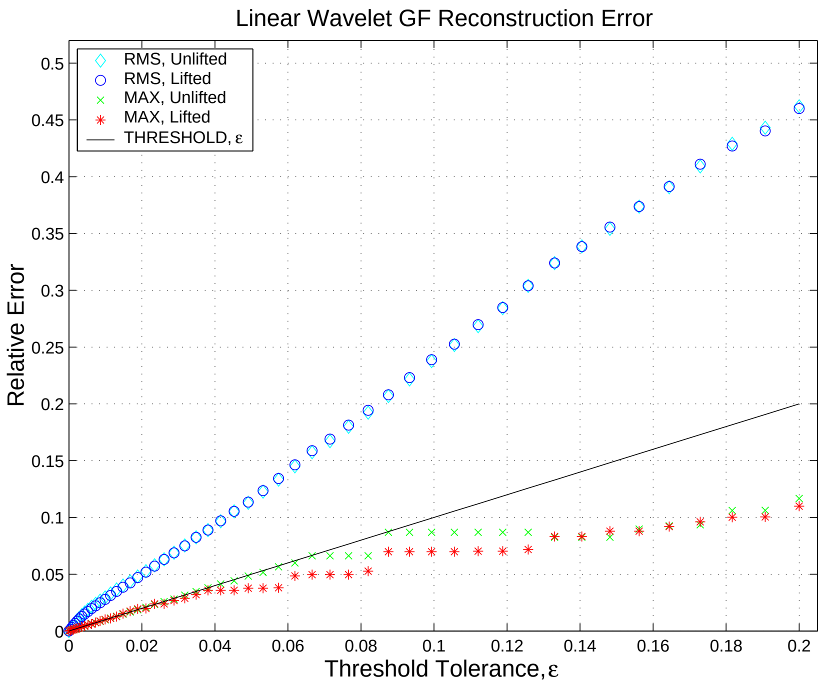


Figure 7.21: *Reality-based tiger model* ($L = 1$)*: Wavelet compression* of two-level model with *thresholded base resolution.* Compression can exceed a factor of four with moderate error. (Top) Relative RMS error; (Bottom) Relative MAX error.



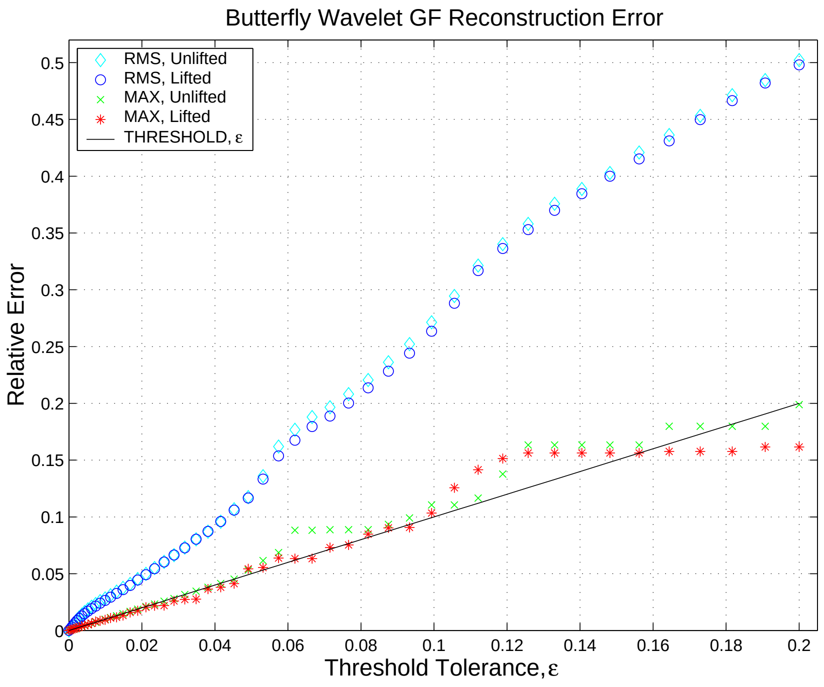


Figure 7.22: *Reality-based tiger model ($L = 1$): Wavelet GF error versus thresholding tolerance* of two-level model with *thresholded base resolution.* (Top) Linear wavelets; (Bottom) Butterfly wavelets.

# Chapter 8

# Conclusion

## 8.1 Summary and Conclusions

This thesis has outlined a framework for interactive simulation of large scale Green's function (GF) based physical models which is a significant improvement over previously known approaches. The Capacitance Matrix Algorithm (CMA) formalism for efficiently simulating precomputed GF models allows arbitrary discretized approximations of linear elliptic partial differential equations to be simulated using a common framework, and this is useful conceptually and also from a software perspective. The easy access to capacitance matrix compliance models has proven highly effective for supporting haptic and force feedback interaction using ordinary personal computers and our ARTDEFO interactive simulator written in Java™. Efficient methods introduced for sequential updating cached capacitance matrix inverses were highly effective and provided significant speedups during our ARTDEFO simulations of unilateral contact. Numerous multiresolution enhancements were presented (hierarchical wavelet GFs, fast summation CMA, multiresolution constraints) and shown to offer dramatic improvements in CMA effectiveness. These multiresolution improvements greatly extend the complexity of models that can be interactively simulated and also precomputed. In particular, the fast summation simulation enhancements which exploit wavelet GF compression were highly successful; we have shown how to achieve hundredfold reductions in interactive simulation costs for geometrically complex elastic models (dragon $L = 3$) at acceptable levels of error (5% RMS). We also showed that similar compression rates could in fact be achieved using a wide range of second-generation lifted wavelet schemes. New approaches for hierarchically precomputing large scale models, and also acquiring multiresolution GF models using reality based robotic measurement techniques turned out to be highly effective. Several suggestions have been made to address the limitations associated with linear strain and material properties: the interpretation of the CMA as a sensitivity analysis method extends it to nonlinear elastostatics; multizone domain decomposition methods can also provide for nonlinear simulation by using hybrid nonlinear models, as well as more general kinematic relationships which can support large relative strains; precomputed nonlinear reanalysis can also be used to simulate nonlinear material properties. With these improvements in efficiency and extensions for large scale simulation, interactive Green's function based models will be of much greater use in interactive computer graphics, computer haptics, and related applications such as virtual prototyping, video games, character animation, surgical simulation, and interactive assembly planning.

## 8.2 Future Work

There are a number of promising directions for future research with perhaps the largest area being the simulation and visualization of other physical systems not directly related to elastostatics, e.g., thermostatics, hydrostatics, electrostatics, etc. The benefit of other wavelet schemes should be studied for compression improvements and practical concerns such as the accommodation of common discretizations and definitions on irregular (base) meshes. Other issues related to smoothness of discretization spaces, larger models, and high accuracy tolerances should also be considered. Recently it has been shown that smooth wavelets based on Loop subdivision achieve excellent compression for complicated geometry [KSS00] at the cost of an expensive (but here affordable) forward transform, and these schemes have desireable properties for compression, summation and visual smoothness; this is an avenue for future research, however we note that our results do not initially suggest that such wavelets will provide *significant* speedup for GF compression. The compression of GF matrix blocks other than the free surface self-effect (illustrated in Figure 3.4, p. 47), should be investigated; preliminary studies indicate that this is also effective. Algorithms for adaptive multiresolution approximations of contact constraints for real time simulation are needed, e.g., to avoid "popping" artifacts. A careful study of nonsmooth contact mechanics should be done to determine approximate yet plausible (frictional) contact models suitable for dynamic or purely kinematic interactive applications. Incorporating LEGFMs into a dynamic simulator, possibly with modeled contact sounds, would of course be a natural extension. Methods for multizone kinematic elastostatic models appears to be very promising for interactive simulation of constrained multibody flexible structures, and should be investigated further. Several of the nonlinear material and strain extensions briefly mentioned hold promise, but will require significant study to determine their utility in different settings. Collision detection for deformable objects can be optimized for LEGFMs given the efficient random access to state values. Effective strategies for precomputation of very large models can also be further improved; investigation of block iterative methods for solving systems with multiple right hand sides [SG96], and the effects of subdivision mesh grading on iterative solves should both be considered. Issues related to the stable simulation of models which contain errors need to be better understood; this is centrally related to the simulation of wavelet compressed models and also models acquired with physical measurement. Lastly, the methods presented here are suitable for hard real time simulation environments and could be further studied in such a context.

# Bibliography

[AB93] M. H. Aliabadi and C. A. Brebbia, editors. *Computational Methods in Contact Mechanics*. Computational Mechanics Publications and Elsevier Applied Science, 1993.

[ABCR93] B. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin. Wavelet-like bases for the fast solution of second-kind integral equations. *SIAM Journal on Scientific Computing*, 14(1):159–184, January 1993.

[AGH01] Mehmet A. Akgün, John H. Garcelon, and Raphael T. Haftka. Fast exact linear and non-linear structural reanalysis and the Sherman-Morrison-Woodbury formulas. *International Journal for Numerical Methods in Engineering*, 50:1587–1606, 2001.

[AH98] Oliver Astley and Vincent Hayward. Multirate haptic simulation achieved by coupling finite element meshes through norton equivalents. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1998.

[AP98] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, Philadelphia, 1998.

[APC97] U. Ascher, D. K. Pai, and B. Cloutier. Forward dynamics, elimination methods, and formulation stiffness in robot simulation. *International Journal of Robotics Research*, 16(6):749–758, December 1997.

[Aro76] J.S. Arora. Survey of structural reanalysis techniques. *Journal of the Structural Division, ACSE*, 102:783–802, 1976.

[ARW95] Uri Ascher, Steve Ruuth, and Brian Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal of Numerical Analysis*, 32:797–823, 1995.

[Bal00] Remis Balaniuk. Building a haptic interface based on a buffer model. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[Bar92] James R. Barber. *Elasticity*. Kluwer Press, Dordrecht, first edition, 1992.

[Bar96] David Baraff. Linear-Time simulation using lagrange multipliers. In *SIGGRAPH 96 Conference Proceedings*, pages 137–146, 1996.

[Bas01] C. Basdogan. Real-time Simulation of Dynamically Deformable Finite Element Models Using Modal Analysis and Spectral Lanczos Decomposition Methods. In *Proceedings of the Medicine Meets Virtual Reality (MMVR'2001) Conference*, pages 46–52, 2001.

[BC96] Morten Bro-Nielsen and Stephane Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum*, 15(3):57–66, August 1996.

[BC00] J. R. Barber and M. Ciavarella. Contact mechanics. *International Journal of Solids and Structures*, 37:29–43, 2000.

[BCR91a] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms. *Comm. in Pure and Applied Math.*, 44:141–183, 1991.

[BCR91b] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Communications on Pure and Applied Mathematics*, XLIV:141–183, 1991.

[Bey92] G. Beylkin. On the representation of operators in bases of compactly supported wavelets. *SIAM Journal on Numerical Analysis*, 29(6):1716–1740, December 1992.

[BG66] J. M. Bennett and D. R. Green. Updating the inverse or triangular factors of a modified matrix. Technical Report 42, Computing Department, University of Sidney, 1966.

[BGS70] R. H. Bartels, G. H. Golub, and M. A. Saunders. Numerical techniques in mathematical programming. In J. B. Rosen, O. L. Mangasarian, and K. Ritter, editors, *Symposium on Nonlinear Programming (1st: 1970: Madison, WI, USA)*, Publication of the Mathematics Research Center, University of Wisconsin, Madison, pages 123–176. Academic Press, Boston, MA, USA, 1970.

[BH93] J-FM Barthelemy and RT Haftka. Approximation concepts for optimum design–a review. *Structural Optimization*, 5:129–144, 1993.

[BN96] Morten Bro-Nielsen. Surgery simulation using fast finite elements. *Lecture Notes in Computer Science*, 1131:529–, 1996.

[BTW84] C. A. Brebbia, J. C. F. Telles, and L. C. Wrobel. *Boundary Element Techniques: Theory and Applications in Engineering*. Springer-Verlag, New York, second edition, 1984.

[BW92] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH 92 Proceedings)*, volume 26, pages 303–308, July 1992.

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 43–54. ACM SIGGRAPH, Addison Wesley, July 1998.

[CDA99] S. Cotin, H. Delingette, and N. Ayache. Realtime elastic deformations of soft tissues for surgery simulation. *IEEE Transactions On Visualization and Computer Graphics*, 5(1):62–73, 1999.

[CDF92] A. Cohen, I. Daubechies, and J. Feauveau. Bi-orthogonal bases of compactly supported wavelets. *Comm. Pure and Appl. Math.*, 45:485–560, 1992.

[Che87] M. Chen. On the solution of circulant linear systems. *SIAM Journal on Numerical Analysis*, 24:668–683, 1987.

[CM92] T.P. Caudell and D.W. Mizell. Augmented Reality: An Application of Heads-Up Display Technology to Manual Manufacturing Processes. In *Proceedings of Hawaii Intl. Conf. on System Sciences*, volume 2, pages 659–669, January 1992.

[CPD$^+$96] Andrew Certain, Jovan Popović, Tony DeRose, Tom Duchamp, David Salesin, and Werner Stuetzle. Interactive multiresolution surface viewing. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 91–98. ACM SIGGRAPH, Addison Wesley, August 1996.

[CS85] Tony F. Chan and Faisal Saied. A comparison of elliptic solvers for general two-dimensional regions. *SIAM Journal on Scientific and Statistical Computing*, 6(3):742–760, July 1985.

[cT00] Murak Cenk Çavuşoğlu and Frank Tendick. Multirate simulation for high fidelity haptic interaction with deformable objects in virtual environments. In

*Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, USA, 2000.

[Cyb] Cyberware, http://www.cyberware.com.

[Dah96] Wolfgang Dahmen. Stability of multiscale transformations. *J. Fourier Anal. Appl.*, 4:341–362, 1996.

[Dau92] Ingrid Daubechies. *Ten Lectures on Wavelets*, volume 61 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[dBL00] Diego d'Aulignac, Remis Balaniuk, and Christian Laugier. A haptic interface for a virtual exam of a human thigh. In *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, USA, 2000.

[DDBC01] Gilles Debunne, Mathieu Desbrun, Alan Barr, and Marie-Paule Cani. Dynamic real-time deformations using space and time adaptive sampling. In *Computer Graphics (SIGGRAPH 2001 Proceedings)*, 2001.

[Del98] H. Delingette. Towards realistic soft tissue modeling in medical simulation. In *Proceedings of the IEEE : Special Issue on Surgery Simulation*, pages 512–523, April 1998.

[DJL92] R.A. DeVore, B. Jawerth, and B.J. Lucier. Image compression through wavelet transform coding. *IEEE Trans. Information Th.*, 38:719–746, 1992.

[DKT98] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 85–94. ACM SIGGRAPH, Addison Wesley, July 1998.

[DLG90] Nira Dyn, David Levin, and John A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, April 1990.

[Dry83] M. Dryja. A finite element - capacitance matrix method for the elliptic problem. *SIAM Journal on Numerical Analysis*, 20(4):671–680, August 1983.

[DS96] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. Technical report, Bell Laboratories, Lucent Technologies, 1996.

[DSB99] Mathieu Desbrun, Peter Schröder, and Alan Barr. Interactive animation of structured deformable objects. In *Graphics Interface*, pages 1–8, June 1999.

[EDD$^+$95] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. *Computer Graphics*, 29(Annual Conference Series):173–182, 1995.

[EEH00] Bernhard Eberhardt, Olaf Etzmuss, and Michael Hauth. Implicit-explicit schemes for fast animation with particle systems. In *Eurographics Computer Animation and Simulation Workshop 2000*, 2000.

[EO89] Y. Ezawa and N. Okamoto. High-speed boundary element contact stress analysis using a super computer. In *Proc. of the* $4^{th}$ *International Conference on Boundary Element Technology*, pages 405–416, 1989.

[Fea87] Roy Featherstone. *Robot dynamics algorithms*. Kluwer, 1987.

[FKR$^+$98] Yuhong Fu, Kenneth J. Klimkowski, Gregory J. Rodin, Emery Berger, James C. Browne, Jürgen K. Singer, Robert A. Van De Geijn, and Kumar S. Vemaganti. A fast solution method for three-dimensional many-particle problems of linear elasticity. *International Journal for Numerical Methods in Engineering*, 42:1215–1229, August 1998.

[GGMS74] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, 28(126):505–535, April 1974.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1995.

[GL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore and London, third edition, 1996.

[GM97] S. F. Gibson and B. Mirtich. A survey of deformable models in computer graphics. Technical Report TR-97-19, Mitsubishi Electric Research Laboratories, Cambridge, MA, November 1997.

[GMW91] P. E. Gill, W. Murray, and M. H. Wright. *Numerical Linear Algebra and Optimization, Vol. 1*. Addison–Wesley, Reading, MA, 1991.

[GR87] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, August 1987.

[GS85] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. on Graphics*, 4:74–123, 1985.

[GSCH93] Steven J. Gortler, Peter Schröder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 221–230, 1993.

[GVSS00] Igor Guskov, Kiril Vidimce, Wim Sweldens, and Peter Schröder. Normal meshes. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, Annual Conference Series, pages 95–102. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[GW01] Igor Guskov and Zoe Wood. Topological Noise Removal. In *Graphics Interface*, June 2001.

[Hac85] Wolfgang Hackbusch. *Multi-Grid Methods and Applications*. Springer, Berlin, 1985.

[Hag89] William W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, June 1989.

[Har85] Friedel Hartmann. *The mathematical foundation of structural mechanics*. Springer-Verlag, New York, 1985.

[Har89] Friedel Hartmann. *Introduction to Boundary Elements: Theory and Applications*. Springer-Verlag, Berlin, 1989.

[HBS99] C.-H. Ho, C. Basdogan, and M. A. Srinivasan. Efficient point-based rendering techniques for haptic display of virtual objects. *Presence*, 8(5):477–491, 1999.

[HDD$^+$94] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise Smooth Surface Reconstruction. In *SIGGRAPH 94 Conference Proceedings*, pages 295–302, 1994.

[HE01] M. Hauth and O. Etzmuß. A high performance solver for the animation of deformable objects using advanced numerical methods. In *Proceedings of Eurographics 2001*, 2001. to appear.

[HK98] Koichi Hirota and Toyohisa Kaneko. Representation of Soft Objects in Virtual Environments. In *Proceedings of the 2nd International Conference on Artificial Reality and Tele-Existence*, December 21–23 1998.

[HL98] K. V. Hansen and O. V. Larsen. Using region-of-interest based finite element modeling for brain-surgery simulation. *Lecture Notes in Computer Science*, 1496:305–, 1998.

[HN89] W. Hackbusch and Z. P. Nowak. On the fast matrix multiplication in the boundary element method by panel clustering. *Numerische Mathematik*, 54(4):463–491, 1989.

[Hub95] Philip M. Hubbard. *Collision Detection for Interative Graphics Applications*. PhD thesis, Dept. of Comp. Science, Brown University, Box 1910, Providence, RI 02912, May 1995.

[Imm] Immersion Corporation. VirtualHand SDK, http://www.immersion.com.

[Joh85] K. L. Johnson. *Contact Mechanics*. Cambridge University Press, Cambridge, 1985.

[JP99a] Doug L. James and Dinesh K. Pai. ARTDEFO: Accurate Real Time Deformable Objects. In *Computer Graphics Proceedings (SIGGRAPH 99)*, pages 65–72. ACM Siggraph, 1999.

[JP99b] Doug L. James and Dinesh K. Pai. ARTDEFO: Accurate Real Time Deformable Objects. In *SIGGRAPH 99 Conference Proceedings Video Tape*, Annual Conference Series. ACM SIGGRAPH, 1999.

[JP01] Doug L. James and Dinesh K. Pai. A Unified Treatment of Elastostatic Contact Simulation for Real Time Haptics. *Haptics-e, The Electronic Journal of Haptics Research (www.haptics-e.org)*, 2(1), September 2001.

[JP02] Doug L. James and Dinesh K. Pai. Real Time Simulation of Multizone Elastokinematic Models. In *ICRA2002: IEEE International Conference on Robotics and Automation*, 2002. (Submitted).

[JS77] M. A. Jaswon and G. T. Symm. *Integral equation methods in potential theory and elastostatics*. Academic Press, New York, 1977.

[KCC$^+$00] Y.-M. Kang, J.-H. Choi, H.-G. Cho, D.-H. Lee, and C.-J. Park. Real-time animation technique for flexible and thin objects. In *Proceedings of WSCG*, pages 322–329, February 2000.

[KcM99] U. Kühnapfel, H.K. Çakmak, and H. Maaß. 3d modeling for endoscopic surgery. In *Proceedings of IEEE Symposium on Simulation*, pages 22–32, Delft University, Delft, NL, October 1999.

[Kel29] O. D. Kellogg. *Foundations of potential theory*. Springer, Berlin, 1929.

[KKP91] J. H. Kane, D. E. Keyes, and K. Guru Prasad. Iterative solution techniques in boundary element analysis. *International Journal for Numerical Methods in Engineering*, 31:1511–1536, 1991.

[KL96] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. *Computer Graphics*, 30(Annual Conference Series):313–324, 1996.

[KL97] K. Kolarov and W. Lynch. Compression of functions defined on the surface of 3d objects. In J. Storer and M. Cohn, editors, *Proc. of Data Compression Conference*, pages 281–291. IEEE Computer Society Press, 1997.

[KLM01] P. Krysl, S. Lall, and J. E. Marsden. Dimensional model reduction in non-linear finite element dynamics of solids and structures. *International Journal for Numerical Methods in Engineering*, 51:479–504, 2001.

[KROM99] C. Kane, E. A. Repettto, M. Ortiz, and J. E. Marsden. Finite element analysis of nonsmooth contact. *Computer Methods in Applied Mechanics and Engineering*, 180, 1999.

[KSS00] Andrei Khodakovsky, Peter Schröder, and Wim Sweldens. Progressive geometry compression. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, Annual Conference Series, pages 271–278. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[KT87] A. M. Abu Kassim and B. H. V. Topping. Static reanalysis: a review. *Journal of Structural Engineering*, 113:1029–1045, 1987.

[Lag97] Christian Lage. The application of object oriented methods to boundary elements. *J. Comp. Math. Appl. Mech. Eng*, 1997.

[Lan01] Jochen Lang. *Deformable Model Acquisition and Verification*. PhD thesis, Department of Computer Science, University of British Columbia, 2001.

[LDW97] Michael Lounsbery, Tony D. DeRose, and Joe Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 16(1):34–73, January 1997. ISSN 0730-0301.

[LMH00] Aaron Lee, Henry Moreton, and Hugues Hoppe. Displaced subdivision surfaces. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, Annual Conference Series, pages 85–94. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[LNPE92] C. Lubich, U. Nowak, U. Pohle, and Ch. Engstler. Mexx – numerical software for the integration of constrained mechanical multibody systems. Tech. report SC92-12, Konrad-Zuse-Zentrum für Informationstechnik, Berlin., 1992.

[Loo87] Charles Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[Lov27] A. E. H. Love. *A Treatise on the Mathematical Theory of Elasticity*. Dover Publications, New York, fourth edition, 1927.

[LSS$^+$98] A. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. Maps: Multiresolution adaptive parameterization of surfaces, 1998.

[LV98] Shang-Hong Lai and Baba C. Vemuri. Generalized capacitance matrix theorems and algorithm for solving linear systems. *SIAM Journal on Scientific Computing*, 19(3):1024–1045, May 1998.

[MAR93] K. W. Man, M. H. Aliabadi, and D. P. Rooke. Analysis of Contact Friction using the Boundary Element Method. In M. H. Aliabadi and C. A. Brebbia, editors, *Computational Methods in Contact Mechanics*, chapter 1, pages 1–60. Computational Mechanics Publications and Elsevier Applied Science, 1993.

[MP78] D. E. Muller and F. Preparata. Finding the intersection of two convex polyhedra. *Theoret. Comput. Sci.*, 7:217–236, 1978.

[MS94] T. H. Massie and J. K. Salisbury. The phantom haptic interface: A device for probing virtual objects. In *ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Chicago, IL, Nov. 1994.

[MS96] Hugh B. Morgenbesser and Mandayam A. Srinivasan. Force shading for haptic shape perception. In *Proceedings of the ASME Dynamics Systems and Control Division*, volume 58, 1996.

[NH97] T. Nishida and K. Hayami. Application of the fast multipole method to the 3-D BEM analysis of electron guns. In M. Marchetti, C. A. Brebbia, and M. H. Aliabadi, editors, *International Conference on Boundary Element Methods (19th: 1997: Rome, Italy)*, volume 19, pages 613–624, Southampton, UK, 1997. Computational Mechanics.

[NKLW94] K. Nabors, F. T. Korsmeyer, F. T. Leighton, and J. White. Preconditioned, adaptive, multipole-accelerated iterative methods for three-dimensional first-kind integral equations of potential theory. *SIAM Journal on Scientific Computing*, 15(3):713–735, May 1994. Iterative methods in numerical linear algebra (Copper Mountain Resort, CO, 1992).

[OW79] Dianne P. O'Leary and Olof Widlund. Capacitance matrix methods for the Helmholtz equation on general three-dimensional regions. *Mathematics of Computation*, 33(147):849–879, July 1979.

[Par] Paraform, http://www.paraform.com.

[PDA01] G. Picinbono, H. Delingette, and N. Ayache. Non-linear and anisotropic elastic soft tissue models for medical simulation. In *ICRA2001: IEEE International Conference on Robotics and Automation*, Seoul Korea, May 2001.

[PFTV87] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*, chapter Sherman-Morrison and Woodbury, pages 66–70. Cambridge University Press, Cambridge, 1987.

[PLLW99] D. K. Pai, J. Lang, J. E. Lloyd, and R. J. Woodham. ACME, A Telerobotic Active Measurement Facility. In *Proceedings of the Sixth Intl. Symp. on Experimental Robotics*, 1999.

[PN95] A. P. Peirce and J. A. L. Napier. A spectral multipole method for efficient solution of large-scale boundary element models in elastostatics. *International Journal for Numerical Methods in Engineering*, 38(23):4009–4034, 1995.

[PV94] W. Proskurowski and P. S. Vassilevski. Preconditioning capacitance matrix problems in domain imbedding. *SIAM Journal on Scientific Computing*, 15(1):77–88, January 1994.

[PV95] W. Proskurowski and P. S. Vassilevski. Preconditioning nonsymmetric and indefinite capacitance matrix problems in domain imbedding. *SIAM Journal on Scientific Computing*, 16(2):414–430, March 1995.

[PvdDJ$^+$01] Dinesh K. Pai, Kees van den Doel, Doug L. James, Jochen Lang, John E. Lloyd, Joshua L. Richmond, and Som H. Yau. Scanning Physical Interaction Behavior of 3D Objects. In *Computer Graphics Proceedings (SIGGRAPH 2001)*. ACM Siggraph, 2001.

[PW80] W. Proskurowski and O. Widlund. A finite element-capacitance matrix method for the Neumann problem for Laplace's equation. *SIAM Journal on Scientific and Statistical Computing*, 1(4):410–425, December 1980.

[PW89] Alex Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics (Proceedings of SIGGRAPH 89)*, 23(3):215–222, July 1989. Held in Boston, Massachusetts.

[Rai] Raindrop Geomagic, Inc., http://www.geomagic.com.

[Rea] Reachin, http://www.reachin.se.

[Rie92] Kurt S. Riedel. A Sherman-Morrison-Woodbury identity for rank augmenting matrices with application to centering. *SIAM Journal on Matrix Analysis and Applications*, 13(2):659–662, April 1992.

[SBD$^+$00] G. Székely, Ch. Brechbühler, J. Dual, R. Enzler, J. Hug, R. Hutter, N. Ironmonger, M. Kauer, V. Meier, P. Niederer, A. Romberg, P. Schmid, G. Schweitzer, M. Thaler, V. Vuskovic, G. Tröster, U. Haller, and M. Bajka. Virtual Reality-Based Simulation of Endoscopic Surgery. *Presence*, 9(3):310–333, June 2000.

[Sen] Sensable Technologies, Inc. GHOST SDK, http://www.sensable.com.

[SG96] V. Simoncini and E. Gallopoulos. A hybrid block GMRES method for nonsymmetric systems with multiple right-hand sides. *J. Comput. Appl. Math.*, 66:457–469, 1996.

[Sha93] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 41(12):3445–3462, 1993.

[She53] J. Sherman. Computations relating to inverse matrices. *Nat. Bur. Standards Appl. Math. Ser.*, 29:13–124, 1953.

[SM50] J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Annals of Mathematical Statistics*, 21:124–127, 1950.

[SP96] Amir Said and William A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, June 1996.

[SS86] Youcef Saad and Martin H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[SS95a] Peter Schröder and Wim Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. In *Computer Graphics Proceedings (SIGGRAPH 95)*, pages 161–172. ACM Siggraph, 1995.

[SS95b] Peter Schröder and Wim Sweldens. Spherical Wavelets: Texture Processing. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 252–263, New York, NY, 1995. Springer-Verlag.

[SS96] Wim Sweldens and Peter Schröder. Building your own wavelets at home. In "Wavelets in Computer Graphics", ACM SIGGRAPH Course Notes, 1996.

[Sta96] Jos Stam. Stochastic dynamics: Simulating the effects of turbulence on flexible structures. Technical report, Inria, 1996.

[Ste79] G. W. Stewart. The effects of rounding error on an algorithm for downdating a Cholesky factorization. *J. Inst. Math. Applic.*, 23:203–13, 1979.

[Sub00] Subdivision for Modeling and Animation. Course Notes of SIGGRAPH 2000, ACM SIGGRAPH, July 2000.

[Swe98] Wim Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM Journal on Mathematical Analysis*, 29(2):511–546, March 1998.

[Tay91] V. Taylor. *Application-specific architectures for large finite-element applications*. PhD thesis, Dept. El. Eng. and Comp. Sci., Univ. of California, Berkeley, CA, 1991. Unpublished doctoral dissertation.

[TF88] D. Terzopoulos and Kurt Fleischer. Deformable models. *The Visual Computer*, 4:306–331, 1988.

[TKN99] T. Takahashi, S. Kobayashi, and N. Nishimura. Fast multipole BEM simulation of overcoring in an improved conical-end borehole strain measurement method. In *Mechanics and Engineering in Honor of Professor Qinghua Du's 80th Anniversary*, pages 120–127, Beijing, 1999. Tsinghua University Press.

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, pages 205–214, July 1987.

[vdDP98] Kees van den Doel and Dinesh K. Pai. The sounds of physical shapes. *Presence*, 7(4):382–395, 1998.

[War95] Joe Warren. Subdivision methods for geometry design. unpublished manuscript, 1995.

[WDGT01] X. Wu, M.S. Downes, T. Goktekin, and F. Tendick. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In A. Chalmers and T.-M. Rhyne, editors, *Eurographics 2001*. Eurographics, 2001.

[Wil89] John Williams. Physically-based modeling: Past, present, and future. SIGGRAPH 89 Panel, 1989.

[Woo50] M. A. Woodbury. Inverting modified matrices. Memorandum Report 42, Statistical Research Group, Princeton, NJ, 1950.

[WW98] Henrik Weimer and Joe Warren. Subdivision schemes for thin plate splines. *Computer Graphics Forum*, 17(3):303–314, 1998. ISSN 1067-7055.

[WW99] Henrik Weimer and Joe Warren. Subdivision schemes for fluid flow. In Alyn Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 111–120, Los Angeles, 1999. Addison Wesley Longman.

[WW00] Joe Warren and Henrik Weimer. Non-stationary subdivision for inhomogeneous operator differential equations. In Larry L. Schumaker, Pierre-Jean Laurent, and Paul Sablonniere, editors, *Curve and Surface Fitting: Saint-Malo 99*. Vanderbilt University Press, 2000.

[XESV97] Julie C. Xia, Jihad El-Sana, and Amitabh Varshney. Adaptive real-time level-of-detail-based rendering for polygonal models. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):171–183, 1997.

[Yip86] E. L. Yip. A note on the stability of solving a rank-$p$ modification of a linear system by the Sherman-Morrison-Woodbury formula. *SIAM Journal on Scientific and Statistical Computing*, 7(2):507–513, April 1986.

[YNK01] Kenichi Yoshida, Naoshi Nishimura, and Shoichi Kobayashi. Application of fast multipole Galerkin boundary integral equation method to elastostatic crack problems in 3D. *International Journal for Numerical Methods in Engineering*, 50:525–547, January 2001.

[Yse86] H. Yserentant. On the multilevel splitting of finite element spaces. *Numer. Math.*, 49:379–412, 1986.

[ZC00] Yan Zhuang and John Canny. Haptic interaction with global deformations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[Zie77] O. C. Zienkiewicz. *The Finite Element Method*. McGraw-Hill Book Company (UK) Limited, Maidenhead, Berkshire, England, 1977.

[ZS94] C. B. Zilles and J. K. Salisbury. A constraint-based god-object method for haptic display. In *ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, volume 1, pages 149–150, Chicago, IL (US), 1994.

[ZSS96] Denis Zorin, Peter Schröder, and Wim Sweldens. Interpolating subdivision for meshes with arbitrary topology. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 189–192. ACM SIGGRAPH, Addison Wesley, August 1996.

[ZSS97] Denis Zorin, Peter Schröder, and Wim Sweldens. Interactive multiresolution mesh editing. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 259–268. ACM SIGGRAPH, Addison Wesley, August 1997.

# Appendix A

# Boundary Integral Formulation of Navier's Equation

## A.1 Navier's Equation

Linear elastostatic objects with isotropic and homogeneous material properties, have displacements satisfying the well-known Navier's equation on $\Omega$,

$$G\sum_{k=1}^{3}\left(\frac{\partial^2 u_i}{\partial x_k^2}+\frac{1}{1-2\nu}\frac{\partial^2 u_k}{\partial x_k \partial x_i}\right)+b_i=0, \tag{A.1}$$

which is conveniently written in a vector operator form as

$$\left(\mathbf{N}\mathbf{u}\right)(\mathbf{x})+\mathbf{b}(\mathbf{x})=0, \qquad \mathbf{x}\in\Omega. \tag{A.2}$$

Here $\nu$ is Poisson's ratio and $G$ is the shear modulus. These are material properties which can be found in handbooks for many materials. Suitable values for Poisson's ratio are $0<\nu\leq\frac{1}{2}$, with $\nu=\frac{1}{2}$ corresponding to an incompressible material. The shear modulus $G$ is positive, with larger values resulting in larger forces accompanying a given deformation. Figure A.2 shows the effect of changing $\nu$; see also Figure A.1.

The traction at a point on the surface is

$$p_i=p_i(\mathbf{x})=G\sum_{j=1}^{3}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)n_j+\frac{2G\nu}{1-2\nu}n_i\left(\sum_{k=1}^{3}\frac{\partial u_k}{\partial x_k}\right) \tag{A.3}$$

where $n_i$ are the direction cosines of the outward normal. In a vector operator notation this becomes

$$\mathbf{p}(\mathbf{x})=\left(\mathbf{P}\mathbf{u}\right)(\mathbf{x}), \qquad \mathbf{x}\in\Gamma. \tag{A.4}$$

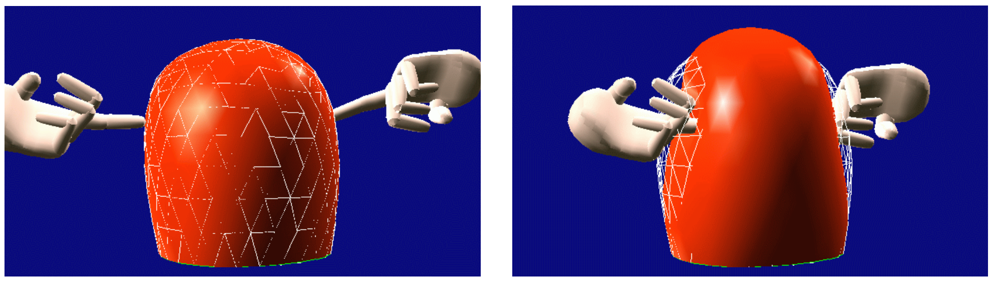

(a) (b)

Figure A.1: A test nodule is pinched between two fingers. The nodule is a Loop subdivision surface, whose control polyhedron is an octahedron. The boundary of one face of the octahedron is tagged as "sharp" [HDD$^+$94] and leads to a sharp edge around the bottom face of the object. We impose a zero-displacement boundary condition on this face, and zero traction everywhere else, except for the two finger contacts.

## A.2 Boundary Integral Formulation

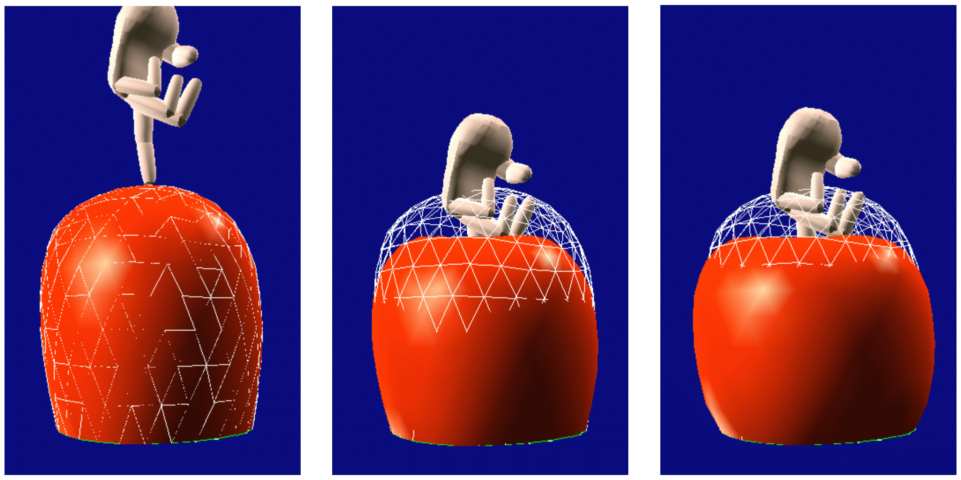
Similar to Laplace's equation, Navier's equation on a domain may be converted to an integral equation defined on the boundary of that domain. At the heart of the derivation is integration by parts, which produces boundary integrals from volume integrals. The end result is that Navier's equations (A.1) on, for example, a bounded domain may be converted into a set of integral equations. The direct boundary integral equation formulation yields the vector integral equation

$$\begin{aligned} &\mathbf{c}(\mathbf{x})\mathbf{u}(\mathbf{x}) + \int_\Gamma \mathbf{p}^*(\mathbf{x},\mathbf{y})\mathbf{u}(\mathbf{y})\, d\Gamma_{\mathbf{y}} \\ &= \int_\Gamma \mathbf{u}^*(\mathbf{x},\mathbf{y})\mathbf{p}(\mathbf{y})\, d\Gamma_{\mathbf{y}} + \int_\Omega \mathbf{u}^*(\mathbf{x},\mathbf{y})\mathbf{b}(\mathbf{y})\, d\Omega_{\mathbf{y}} \end{aligned} \tag{A.5}$$

valid at a point $\mathbf{x}$ on the boundary $\Gamma$ [BTW84]. Three matrix functions occur in this equation:

$$\begin{aligned} \mathbf{c} &= \mathbf{c}(\mathbf{x}) &&= [c_{ij}], \\ \mathbf{u}^* &= \mathbf{u}^*(\mathbf{x},\mathbf{y}) &&= [u^*_{ij}], \\ \mathbf{p}^* &= \mathbf{p}^*(\mathbf{x},\mathbf{y}) &&= [p^*_{ij}]. \end{aligned}$$

The integral kernel functions $u^*_{ij}(\mathbf{x},\mathbf{y})$ and $p^*_{ij}(\mathbf{x},\mathbf{y})$ are known fundamental solutions and tractions, respectively, and are provided in the next section. The coefficient $c_{ij}(\mathbf{x})$ depends on the smoothness properties of the boundary at $\mathbf{x}$, but is not needed explicitly (see Appendix B.2.2).



(a) (b) (c)

Figure A.2: Poisson's ratio, $\nu$, provides an easy way to describe the compressibility of a material. Figure (a) shows the example nodule in its rest state. A coarse reference mesh is also shown in white on the surface. In figure (b), $\nu = 0.01$, making the material very compressible; the nodule exhibits a sponge-like behavior, deforming mainly in the vicinity of the contact. In Figure (c) $\nu = 0.5$, making the material incompressible; the sides of the nodule bulge to conserve volume.

### A.2.1 Fundamental Solutions

The fundamental solutions of Navier's equation, $u^*_{ij}(\mathbf{x}, \mathbf{y})$, correspond to the displacement in the $j^{th}$ direction at a field point, $\mathbf{y}$, as produced by a unit point load applied in each of the $i$ directions at a specified load point, $\mathbf{x}$, in an infinite linear elastic medium. This corresponds to the fundamental solution due to Kelvin [Lov27]. Conceptually, this point load fundamental solution plays an analogous role in elasticity as the familiar $\frac{1}{r}$ Coulomb potential solution accompanying a point charge in electrostatics. In both cases, the fundamental solutions are highly localized and decay very quickly, e.g., the fundamental displacements have a typical $\frac{1}{r}$ character while the fundamental tractions behave like $\frac{1}{r^2}$. Mathematically, $u^*_{ij}(\mathbf{x}, \mathbf{y})$ is the $j^{th}$ component of the displacement solution to

$$(\mathbf{Nu})\,(\mathbf{y}) + \delta(\mathbf{x} - \mathbf{y})\hat{\mathbf{e}}_i = 0, \tag{A.6}$$

where the vector operator notation from (A.2) has been used. The fundamental tractions are related to the fundamental displacements via (A.4), that is

$$\mathbf{p}^* = \mathbf{P}\mathbf{u}^*. \tag{A.7}$$

Expressions for the fundamental solutions are [BTW84]

$$u^*_{ij}(\mathbf{x}, \mathbf{y}) = \frac{1}{16\pi(1-\nu)G}\left\{\frac{(3-4\nu)\,\delta_{ij}}{r} + \frac{r_i r_j}{r^3}\right\}$$

$$p^*_{ij}(\mathbf{x},\mathbf{y}) = \frac{(1-2\nu)}{8\pi(1-\nu)}\left[\frac{(r_i n_j - r_j n_i)}{r^3} - \left\{\frac{\delta_{ij}}{r^2} + \frac{3 r_i r_j}{(1-2\nu)\, r^4}\right\}\frac{\partial r}{\partial \mathbf{n}(\mathbf{y})}\right]$$

where

$$\begin{aligned} \mathbf{r} &= \mathbf{y}-\mathbf{x} & r &= |\mathbf{r}| \\ r_i &= (\mathbf{r})_i & \frac{\partial r}{\partial \mathbf{n}(\mathbf{y})} &= \frac{\mathbf{r}\cdot\mathbf{n}(\mathbf{y})}{r} \end{aligned} \qquad \text{(A.8)}$$

and $\mathbf{n}(\mathbf{y})$ is the outward unit normal at $\mathbf{y} \in \Gamma$.

### A.2.2 Internal Body Forces

Any user-specified body forces mildly complicate the boundary-only character of the integral equations, as they introduce a volume integral term in (A.5). However, for certain classes of functions, e.g., polynomials, it is possible to analytically convert the volume integral into a boundary integral using essentially repeated integration by parts via the Multiple Reciprocity Method [BTW84]. For example, a constant gravitational force, $\mathbf{b} = \rho\mathbf{g}$, may be evaluated as a boundary integral. More simply, concentrated force loads, $\mathbf{b} = \mathbf{b_0}\delta(\mathbf{x}-\mathbf{x}_0)$, are trivial to integrate, and are useful for introducing internal body articulation. Due to space limitations, the body force term will not be mentioned hereafter.

# Appendix B

# Boundary Element Method (BEM) Tutorial

## B.1 The Boundary Element Method

The Boundary Element Method (BEM) is a straight-forward approach to discretizing integral equations defined on the boundary via a collocation method. There are three main steps when implementing the BEM in 3D:

1. Discretize the boundary $\Gamma$ into a set of $N$ non-overlapping elements which represent the displacements and tractions by functions which are piecewise interpolated between the element's nodal points.

2. Apply the integral equation(s) at each of the $n$ boundary nodes, and perform the resulting integrals over each boundary element in order to generate an undetermined system of $3n$ equations involving the $3n$ nodal displacements and $3n$ nodal tractions.

3. Apply the boundary conditions of the desired boundary value problem, fixing $n$ nodal values (either displacement or traction) per direction. The remaining linear system of $3n$ equations is determined and may be solved to obtain the unknown nodal boundary values.

Drawing on the notation from [BTW84], the discretization of (A.5), dropping the body force, may be summarized as follows. The piecewise interpolated displacement and traction functions evaluated at the point $\mathbf{x}$ may be written as

$$\begin{aligned} \mathbf{u} &= \mathbf{u}(\mathbf{x}) = (u_1, u_2, u_3)^T = \mathbf{\Phi}(\mathbf{x})\mathsf{u} \\ \mathbf{p} &= \mathbf{p}(\mathbf{x}) = (p_1, p_2, p_3)^T = \mathbf{\Phi}(\mathbf{x})\mathsf{p} \end{aligned} \tag{B.1}$$

where $\mathbf{\Phi}(\mathbf{x})$ is an interpolation matrix and $\mathsf{u}$ and $\mathsf{p}$ are $n$-vectors of the nodal displacement and traction 3-vectors, respectively, e.g., $\mathsf{u} = [\mathsf{u}_1, \ldots, \mathsf{u}_n]^T$.

The displacement, traction and $c$ vectors at the $i^{th}$ node, $\mathbf{x}_i$, will be denoted by

$$\mathsf{u}_i = \mathbf{u}(\mathbf{x}_i), \qquad \mathsf{p}_i = \mathbf{p}(\mathbf{x}_i), \qquad \mathsf{c}_i = \mathbf{c}(\mathbf{x}_i),$$

respectively. Substituting (B.1) into the elasticity integral equation (A.5) applied at $\mathbf{x}_i$ and converting the surface integrals into sums of integrals over each boundary element, one obtains

$$\begin{aligned} \mathsf{c}_i\mathsf{u}_i + \sum_{j=1}^{N}\left(\int_{\Gamma_j} \mathbf{p}^*(\mathbf{x}_i,\mathbf{y})\,\mathbf{\Phi}(\mathbf{y})\,d\Gamma_{\mathbf{y}}\right)\mathsf{u} \\ = \sum_{j=1}^{N}\left(\int_{\Gamma_j} \mathbf{u}^*(\mathbf{x}_i,\mathbf{y})\,\mathbf{\Phi}(\mathbf{y})\,d\Gamma_{\mathbf{y}}\right)\mathsf{p} \end{aligned}$$

which, in an obvious notation, may be written as

$$\mathsf{c}_i\mathsf{u}_i + \sum_{j=1}^{n} \hat{\mathsf{h}}_{ij}\mathsf{u}_j = \sum_{j=1}^{n} \mathsf{g}_{ij}\mathsf{p}_j. \tag{B.2}$$

For convenience, define off-diagonal $\mathsf{h}_{ij}$ as $\hat{\mathsf{h}}_{ij}$, but let

$$\mathsf{h}_{ii} = \mathsf{c}_i + \hat{\mathsf{h}}_{ii}. \tag{B.3}$$

Assembling the equations at all nodes into a block matrix system yields

$$\sum_{j=1}^{n} \mathsf{h}_{ij}\mathsf{u}_j = \sum_{j=1}^{n} \mathsf{g}_{ij}\mathsf{p}_j \quad \text{or} \quad \mathsf{Hu} = \mathsf{Gp}. \tag{B.4}$$

The final step is to specify the boundary conditions at each of the $n$ nodes, then bring the unknowns to the left-hand side, and the knowns to the right-hand side to obtain the final linear system

$$\mathsf{Av} = \mathsf{z}, \tag{B.5}$$

which may be solved for the unknown nodal quantities, $\mathsf{v}$.

All that remains is to determine the integrals for the matrix entries of $\mathsf{H}$ and $\mathsf{G}$. Indeed, this is the part of the BEM which takes the majority of a computation. Complete formulae for constant boundary elements, are provided in the Appendix for those who are interested in constructing their own elasticity solver. It is the simplest element for the reader to implement and understand. Formulae for linear elements may be found in [Har89][1]

[1]Unfortunately there is a string of misprint corrections in the literature here. Hartmann [Har89] corrects a misprint in the source, but accidentally introduces another. Fortunately, the false correction is noticeable upon comparison.

### B.1.1 Constant Element Case

Analogous to the midpoint rule for integrating a univariate function, integration of a triangular constant element is accomplished using data located at the centroid. This corresponds to a centroid collocation scheme, as the $j^{th}$ node, $\mathbf{x}_j$, is identified as the centroid of the $j^{th}$ element, and is where the elastic state is represented accurately. In this case $n = N$. Since the collocation node lies in the element's interior, it is called a *nonconforming* element. This happens to make the element particular easy to implement, as connectivity is not required. It also has the convenient casual property that special care need not be taken to accommodate corners or sharp edges [BTW84].

## B.2 Constant Element Influence Coefficients

When implementing boundary elements, there are always a number of singular integrals which the user must acquire or spend some time calculating. In the relatively simple case of triangular constant elements with centroid collocation, there are only a few integrals, and they are presented here for completeness.

### B.2.1 Inter-element Effects

These are integrals corresponding to interactions where the load point lies at the centroid of a triangle other than the one being integrated over, i.e., $\mathbf{x}_i \notin \Gamma_j$. This includes the elements of the $3 \times 3$ matrices $\mathbf{g}_{ij}$ and $\hat{\mathbf{h}}_{ij}$, for $i \neq j$, and therefore corresponds to the majority of the integrals. Since $r = |\mathbf{x}_i - \mathbf{y}|$ is never zero, these are nonsingular integrals which may easily be calculated using standard numerical quadrature (see Brebbia [BTW84]).

### B.2.2 Self-effects

Self-effects correspond to the integrals in the diagonal terms of equation B.4 such as $\mathbf{g}_{ii}$ and $\hat{\mathbf{h}}_{ii}$. Since the load point lies in the center of the triangle being integrated over, these are singular integrals, as the fundamental solutions are unbounded as $r \to 0$. The first integral is only weakly singular, while the second integral is strongly singular and only exists in a Cauchy principal value sense.

#### Calculation of $\mathbf{h}_{ii}$

Despite $\hat{\mathbf{h}}_{ii}$ being strongly singular, it is easy to calculate indirectly using rigid body translations by considering a bounded object with all nodes subjected to any arbitrary constant displacement boundary conditions, $\mathbf{u}_j = \bar{\mathbf{u}}, \forall j$. Since this body necessarily experiences no

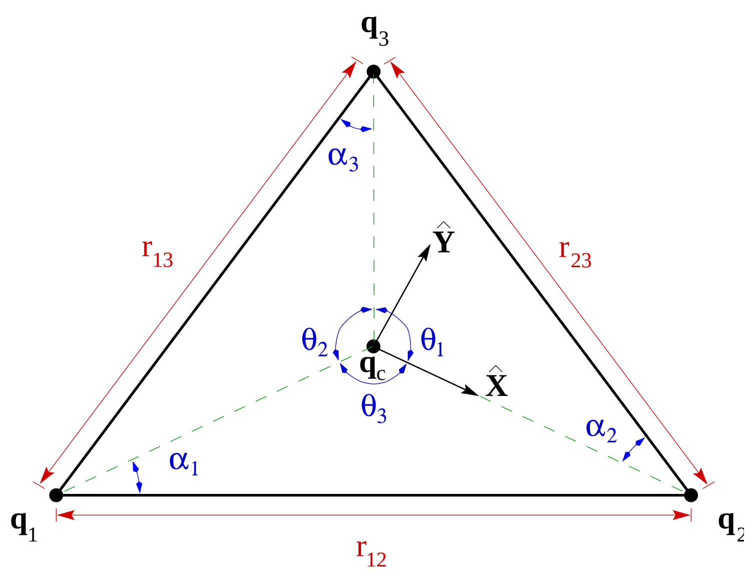


Figure B.1: Notation

induced surface tractions, $\mathbf{p}_j = 0,\ \forall j$. It follows from (B.4) that

$$\mathbf{h}_{ii} = -\sum_{j\neq i} \mathbf{h}_{ij} \tag{B.6}$$

and therefore *neither* $\hat{\mathbf{h}}_{ii}$ *nor* $\mathbf{c}_i$ *need be calculated explicitly.* Note that (B.6) implies that $H$ is a singular matrix.

## Calculation of $\mathbf{g}_{ii}$

The elements of

$$(\mathbf{g}_{ii})_{kl} = \frac{1}{16\pi(1-\nu)G}\int_{\Gamma_i}\left(\frac{(3-4\nu)\,\delta_{kl}}{r} + \frac{r_k r_l}{r^3}\right) d\Gamma_{\mathbf{y}}$$

will be expressed for a triangle $\Delta$, using the notation in figure B.1, i.e., with vertices at $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$, centroid at $\mathbf{q}_c$, area $A$, and an outward unit normal $\hat{\mathbf{n}}$. Omitting any constant factors, the first integral is $J_1^\Delta$, and the second is

$$\hat{\mathbf{X}}_k\hat{\mathbf{X}}_l J_1^\Delta + \frac{1}{2}\left(\hat{\mathbf{X}}_k\hat{\mathbf{Y}}_l + \hat{\mathbf{X}}_l\hat{\mathbf{Y}}_k\right) J_2^\Delta + \left(\hat{\mathbf{Y}}_k\hat{\mathbf{Y}}_l - \hat{\mathbf{X}}_k\hat{\mathbf{X}}_l\right) J_3^\Delta.$$

with

$$\hat{\mathbf{X}} = \frac{\mathbf{q}_2 - \mathbf{q}_c}{|\mathbf{q}_2 - \mathbf{q}_c|}, \quad \hat{\mathbf{Y}} = \hat{\mathbf{n}} \times \hat{\mathbf{X}}, \tag{B.7}$$

and

$$J_m^\Delta = \frac{2A}{3}\left[\frac{J_m(\theta_1,\alpha_2,0)}{r_{23}} + \frac{J_m(\theta_2,\alpha_3,0)}{r_{31}} + \frac{J_m(\theta_3,\alpha_1,\theta_1+\theta_2)}{r_{12}}\right] \tag{B.8}$$

where

$$J_1\left(\Delta\theta, \alpha, \theta_{\min}\right) = \ln\left[\frac{\tan\left(\frac{\Delta\theta+\alpha}{2}\right)}{\tan\left(\frac{\alpha}{2}\right)}\right]$$

$$J_2\left(\Delta\theta, \alpha, \theta_{\min}\right) = \sin\left(\frac{\theta_{\min}-\alpha}{2}\right)\ln\left[\frac{\tan\left(\frac{\Delta\theta+\alpha}{4}\right)}{\tan\left(\frac{\alpha}{4}\right)}\right] + \cos\left(\frac{\theta_{\min}-\alpha}{2}\right)\ln\left[\frac{\left[1-\tan\left(\frac{\alpha}{4}\right)\right]\left[1+\tan\left(\frac{\Delta\theta+\alpha}{4}\right)\right]}{\left[1+\tan\left(\frac{\alpha}{4}\right)\right]\left[1-\tan\left(\frac{\Delta\theta+\alpha}{4}\right)\right]}\right]$$

$$J_3\left(\Delta\theta, \alpha, \theta_{\min}\right) = 2\sin\left(2\theta_{\min}-\alpha+\frac{\Delta\theta}{2}\right)\sin\left(\frac{\Delta\theta}{2}\right) - \sin^2\left(\theta_{\min}-\alpha\right)\ln\left[\frac{\tan\left(\frac{\alpha}{2}\right)}{\tan\left(\frac{\Delta\theta+\alpha}{2}\right)}\right].$$

# Appendix C

# Justification of Interpolated Traction Distributions for Point Contact

This section derives the nodal boundary conditions associated with a localized point contact at an arbitrary mesh location. The practical consequence is that the discrete traction distribution may be conveniently interpolated from suitable nearby nodal distributions or masks.

Given a continuous surface traction distribution, $\mathbf{p}(\mathbf{x})$, a corresponding discrete distribution $\Phi(\mathbf{x})\mathsf{p}$ may be determined by a suitable projection into $\mathcal{L}$ of each Cartesian component of $\mathbf{p}(\mathbf{x})$. For example, consider the projection of a scalar function on $\Gamma$ defined as the minimizer of the scalar functional $\mathcal{E} : \mathbb{R}^{3n} \mapsto \mathbb{R}$,

$$\mathcal{E}(\mathsf{p}) = \int_\Gamma \left[ \|\mathbf{p}(\mathbf{x}) - \Phi(\mathbf{x})\mathsf{p}\|_2^2 + \|B\Phi(\mathbf{x})\mathsf{p}\|_2^2 \right] d\Gamma_{\mathbf{x}}, \tag{C.1}$$

where $B : \mathcal{L} \mapsto \mathbb{R}$ is some linear operator that can be used, e.g., to penalize nonsmooth functions, and $\mathbf{\Phi}(\mathbf{x}) : \mathbb{R}^{3n} \mapsto \mathbb{R}^3$ is a nodal interpolation matrix defined on the surface,

$$\mathbf{\Phi}(\mathbf{x}) = [\Phi_1(\mathbf{x})\Phi_2(\mathbf{x}) \cdots \Phi_n(\mathbf{x})] = [\phi_1(\mathbf{x})\phi_2(\mathbf{x}) \cdots \phi_n(\mathbf{x})] \otimes I_3, \quad \mathbf{x} \in \Gamma, \tag{C.2}$$

with $\Phi_j(\mathbf{x}) = \phi_j(\mathbf{x}) I_3$ and $I_3$ the 3-by-3 identity matrix. The Euler-Lagrange equations for this minimization are

$$\sum_{j=1}^{n} \left( \int_\Gamma \left[ \phi_i(\mathbf{x})\phi_j(\mathbf{x}) + (B\phi_i(\mathbf{x}))\,(B\phi_j(\mathbf{x})) \right] d\Gamma_{\mathbf{x}} \right) \mathsf{p}_j = \int_\Gamma \phi_i(\mathbf{x})\mathbf{p}(\mathbf{x}) d\Gamma_{\mathbf{x}}, \tag{C.3}$$

$i = 1, 2, \ldots, n$, which, in an obvious notation, is written as the linear matrix problem

$$\mathcal{A}\mathsf{p} = \mathsf{f} \tag{C.4}$$

to be solved for the nodal traction values $\mathsf{p}$. Note that $\mathcal{A}$ has units of area.

The relevant traction distribution for point-like contact is a scale-independent concentrated point load

$$\mathbf{p}(\mathbf{x}) = \mathbf{p}^{\delta}(\mathbf{x}) = \mathbf{f}^{\delta}\delta(\mathbf{x} - \mathbf{x}^{\delta}) \tag{C.5}$$

which models a force $\mathbf{f}^{\delta}$ delivered at $\mathbf{x}^{\delta} \in \Gamma$. The force $n$-vector in equation (C.4) has components

$$\mathsf{f}_i = \phi_i(\mathbf{x}^{\delta})\mathbf{f}^{\delta} \tag{C.6}$$

and the corresponding pressure distribution's nodal values are

$$\mathsf{p} = \mathcal{A}^{-1}\mathsf{f}. \tag{C.7}$$

For compactly supported basis functions, $\phi_i(\mathbf{x})$, $\mathsf{f}$ has only a small number of nonzero components for any given $\mathbf{x}$. Hence $\phi_i(\mathbf{x}^{\delta})$ are the interpolation weights describing the contribution of the nearby nodal pressure distributions, here specified by the columns of $\mathcal{A}^{-1}$.

As an example, consider the important case where $\mathcal{L}$ is a continuous piecewise linear function space with $\phi_i(\mathbf{x}_j) = \delta_{ij}$. This was the space used in our implementation. In this case, at most only three components of $\mathsf{f}$ are nonzero, given by the indices $\{i_1, i_2, i_3\}$ which correspond to vertices of the contacted triangle $\tau^{\delta}$, i.e., for which $\mathbf{x}^{\delta} \in \tau^{\delta}$. The values $\phi_i(\mathbf{x}^{\delta})$ are the barycentric coordinates of $\mathbf{x}^{\delta}$ in $\tau^{\delta}$. The pressure distribution's nodal values are then

$$\mathsf{p} = \mathcal{A}^{-1}\mathsf{f} = \sum_{k=1}^{3} \left(\mathcal{A}^{-1}\right)_{:i_k} \mathsf{f}_{i_k} = \sum_{k=1}^{3} \phi_{i_k}(\mathbf{x}^{\delta}) \left[\left(\mathcal{A}^{-1}\right)_{:i_k} \mathbf{f}^{\delta}\right] = \sum_{k=1}^{3} \phi_{i_k}(\mathbf{x}^{\delta})\mathsf{p}^{(i_k)}, \tag{C.8}$$

where $\mathsf{p}^{(i_k)}$ is the pressure distribution corresponding to the application of the load directly to vertex $i_k$. Therefore the *piecewise linear pressure distribution for a point load applied at a barycentric location on a triangle is equal to the barycentric average of the pressure distributions associated with the point load applied at each of the triangle's vertices.* This may be recognized as an elastic generalization of *force shading* [MS96] for rigid models.

Note that the $j^{th}$ column of $\mathcal{A}^{-1}$ is a vertex mask that describes the nodal distribution of the load applied to the $j^{th}$ vertex. By modifying the penalty operator $B$ it would be possible to engineer masks that exhibit varying degrees of smoothness and spatial localization.

# Appendix D

# Software System Overview

This research project also involved the development of several software packages for the precomputation and simulation of LEGFMs. These systems were implemented in Java and graphically rendered with Java3D, with the exception of native interfaces to C++ code for the support of haptic devices. Using high-level object-oriented languages helped reduce the complexity and amount of code required. Nevertheless, the total size of the entire research code base is approximately 125000 lines of Java code, and a few thousand lines of C++ code. Code is grouped into several Java packages, which aid in code reuse and reduce intermingling of unrelated code. Within each package, strong use of object-oriented design (OOD) patterns [GHJV95] was made to increase the general usability of data and algorithm implementations.

The final design is considerate of various Java performance issues. Because of the often high cost of garbage collection for creational patterns in Java, new objects are created sparingly in numerical computations, and every attempt is made to preallocate objects where possible, especially for "real time" simulation. In practice, it was possible to avoid many of the performance pitfalls commonly associated with naive Java programming.

## D.1 Subdivision Geometric Modeling and Wavelets

The complexity of implementing multiresolution algorithms has been eased by designing a geometric modeling package based on geometric subdivision [Sub00], with other members of our Interactive Simulation group. A half-edge data structure [MP78, GS85] is used to represent all polyhedral mesh models as a set of connected faces, edges, and vertices; each simplex is implemented as a separate Java object. Geometric subdivision algorithms are then easily implemented as operations on these primitives. For example, support is provided for traversing the multiresolution subdivision mesh hierarchy, and rendering of subdivided deforming meshes can be performed "in place" for interactive applications. Simplicies may be indexed to support random access to these primitives, e.g., vertices, and ordering of associated data. Collision detection and proximity queries are supported, including a sphere tree approach [Hub95] and a standard octtree space partitioning. The subdivision package also provides support for subdivision connectivity mesh reparameterization and displacement mapping based on displaced subdivision surfaces [GVSS00, LMH00]; the normal piercing phase is accelerated using the octtree spatial hierarchy.

Multiresolution analysis of data defined on the surface geometry is handled by a second package. Most notably it includes lifted fast wavelet transform [Swe98] implementations for processing various surface function data types, as well as methods for accessing multiresolution/hierarchical basis functions and their associated refinement relations.

## D.2 Green's Function Precomputation

Precomputation of GFs is implemented for boundary integral equations, e.g., associated with Laplace and Navier's equations. Discretization is handled by a very general object-oriented Petrov-Galerkin framework, and we refer the reader to [Lag97] for similar design details. Collocation integral equation discretization schemes, such as the BEM, are a subclass of schemes described by this approach, and implementations exist for constant (centroid-based) and isoparametric linear (vertex-based) elements. GF matrix solvers for both direct and iterative methods implement a common GF iterator interface which allows for easy parallel/distributed GF precomputation; for multiresolution models, GFs can be sequentially transformed and thresholded as they become available to avoid memory bottlenecks. Multithreading (for machines with SMP support) is supported in the matrix element computation, and both the direct and iterative GF solution stages. In order to handle very large problems, care is taken to never unnecessarily store or copy large matrices.

## D.3 Interactive Simulation

### D.3.1 Simulation of Green's Function Models

Interfaces are used to describe the various GF data objects, and provide seamless element access and (fast-)summation support for data associated with dense, wavelet and hierarchical GF matrices of assorted floating point formats. The interfaced GF object is used by a BVP solver object to efficiently implement the CMA, seamlessly taking advantage of fast-summation and parallel processing if available. BVPs are specified using a rank-update BVP object which internally compares the BVP to the RBVP. A common interface is implemented by BVP solver objects with different capacitance matrix inversion and caching capabilities. An interface is also used to encapsulate GF models of different underlying representations for use in simulations. For example, the finger model can be represented as a single GF model even though it is actually composed of three connected GF models. No special support for simulating reality-based models is required as this is implicit in the GF model description, however extensive software for acquiring these models was developed in a separate project [PvdDJ$^+$01, Lan01].

### D.3.2 ArtDefo Simulator

Our simulator, named ArtDefo [JP99a], provides support for contact interactions using point-like and convex rigid manipulandums whose motion may be specified by mouse and haptic interfaces, as well as more general means. A contact mediator object resolves contacts and applies appropriate nodal boundary conditions to the deformable object(s) in contact. Convex rigid manipulandums help simplify node collision detection issues, and contact states are currently determined using a “trial and error” approach. At present friction is only implemented for point-like contacts, so that nodes of deformable objects do not slide on rigid objects when in contact. Contact states can be monitored by other objects using a contact observer interface; for example, this is used to implement (unilateral) compliance in the mouse and CyberGlove user interfaces so that large nonphysical forces can not be exerted on the deformable object.

### D.3.3 Haptic Interfaces

Access to native C++ software APIs for haptic interfaces is provided by corresponding wrapper classes which encapsulate Java Native Interface (JNI) calls to the C++ API layer. For example, CyberGlove grasping applications employ a wrapper object which uses JNI to communicate with the VirtualHand SDK [Imm] to instantiate and access a C++ virtual hand object. During simulation, the wrapper object provides read access to finger joint angles and transforms. The accessed hand state is then used to control a second hand with compliant joint kinematics which is resisted by the deformable object. Each 3D finger link’s mesh is associated with a convex rigid manipulandum in the ArtDefo contact simulator.

PHANToM force feedback applications use a larger C++ layer consisting of several objects involved in the simulation of point-like frictional contact. The GHOST SDK’s [Sen] gstForceField class is entended to render contact forces computed by our pressure mask formalism. The collision detection, contact sliding and contact force computation are all performed at approximately 1 kHz by the C++ layer running on a separate thread from the graphics simulation. One-way communication of the contact state information is achieved by writing to a thread-safe object accessible by the Java graphics simulation. The Java layer is responsible for computation and graphical rendering of the surface deformation corresponding to the point contact. In this way, even objects which are too large to be rendered at interactive rates, can still be felt with the force feedback interface due to the loose coupling of force feedback and graphical simulation threads.