Contents

1

Modeling Deformation of Linear Elastostatic Objects

Quasistatic deformation models have been well-known in haptic force-feedback rendering for at least a decade since their introduction by Cotin and others. They provide computationally efficient models of small-deformation response that reach equilibrium at time-scales faster than graphics rates or user interactions. In this chapter, we revisit [James and Pai 01] and show how global deformation of linear elastostatic objects can be solved efficiently using precomputed Green's functions, and fast low-rank updates based on *Capacitance Matrix Algorithms*. Capacitance matrices provide exact contact response models, allowing contact forces to be computed for haptics much faster than global deformation behavior. *Vertex pressure masks* are introduced to support the convenient abstraction of localized scale-specific point-like contact with an elastic and/or rigid surface approximated by a polyhedral mesh. Examples are presented for the CyberGlove™ and PHANToM™ haptic interfaces. Updated timings are provided, exhibiting approximately an order-of-magnitude improvement over [James and Pai 01].

1.1 Introduction

Discrete linear elastostatic models (LEMs) are important physically based elastic primitives for computer haptics because they admit a very high-degree of precomputation, or "numerical compression" [Astley and Hayward 98]. They provide cheap force response models suitable for haptic rendering of stiff elastic objects during continuous contact. The degree of useful precomputation is quite limited for most types of nonlinear and/or dynamical elastic models (although see [Barbič and James 05]), but LEMs are a well-known exception, mainly due to the precomputability of time-independent

Green's functions (GFs) and the applicability of linear superposition principles, and linear system solvers. Intuitively, GFs form a basis for describing all possible deformations of a LEM. Thus, while LEMs form a relatively simple class of elastic models in which geometric and material linearities are an ultimate limitation, the fact that the model is linear is also a crucial enabling factor. We conjecture that LEMs will remain one of the best runtime approximations of stiff elastic models for simulations requiring stable high-fidelity force feedback.

A central idea for LEMs in computer haptics is the formulation of the boundary value problem (BVP) solution in terms of suitable precomputed GFs using *Capacitance Matrix Algorithms* (CMAs). Derived from the Sherman-Morrison-Woodbury formula for low-rank updating of matrix inverses (and factorizations), CMAs have a long history in linear algebra [Press et al. 87, Hager 89], where they have been commonly used for static reanalysis [Kassim and Topping 87], to efficiently solve LEM contact mechanics problems [Ezawa and Okamoto 89, Man et al. 93] and more recently for interactive simulations and haptic rendering [Bro-Nielsen and Cotin 96, Cotin et al. 99, James and Pai 99, James and Pai 03].

For computer haptics, a fundamental reason for choosing to compute the LEM elasticity solution using a CMA formulation, is that the *capacitance matrix*[1] is the main quantity of interest: it is the *compliance matrix* which relates the force feedback response to the imposed contact displacements. Also, the precomputation of GFs effectively decouples the global deformation and force response calculations, so that the capacitance matrix can be extracted from the the GFs at no extra cost; this is the fundamental mechanism by which a haptic interface can efficiently interact with a LEM of very large complexity, such with wavelet GF models [James and Pai 03]). The user can feel no difference between the force response of the complete system and the capacitance matrix, because none exists. Lastly, CMAs are direct matrix solvers whose deterministic operation count is appealing for real-time applications.

The final part of this chapter addresses the special case of point-like haptic contact. It has long been recognized that point contact is a convenient abstraction for haptic interactions, and the PHANToM™ haptic interface is a testament to that fact. While it is possible to consider the contact area to be truly a point for rigid models, infinite contact pressures are problematic for elastic models, and tractions need to be distributed over finite surface areas. We propose to do this efficiently by introducing nodal traction distribution masks which address at least two core issues. First, having a point contact with force distributed over a finite area is somewhat contradictory, and the traction distribution is effectively an un-

[1] The term "capacitance" is due to historical convention [Hager 89].

derdetermined quantity without any inherent spatial scale. This is resolved by treating the contact as a single displacement constraint whose traction distribution enters as a user (or manipulandum) specified parameter. The distribution of force on the surface of the model can then be consistently specified in a fashion which is independent of the scale of the mesh. Second, given the model is discrete, special care must be taken to ensure a sufficiently regular force response on the surface, since irregularities are very noticeable during sliding contact motions. By suitably interpolating nodal traction distributions, displacement constraints can be imposed which are consistent with regular contact forces for numerous discretizations.

Related work on haptic rendering of elastostatics

There are several instances in the literature of real-time simulation of linear elastostatic models based on precomputed GFs methods and related techniques. These models were used because of their low runtime costs, and desirable force-feedback properties. For example, researchers at INRIA have made extensive use of a real-time elastostatic FEM variants for liver related surgical simulations [Bro-Nielsen and Cotin 96, Bro-Nielsen 96, Cotin et al. 98]. During a precomputation phase they have used condensation [Zienkiewicz 77, Bro-Nielsen and Cotin 96] as well as iterative methods [Cotin et al. 99] to compute displacement responses due to unit forces applied to vertices on the "free" boundary. At run time, a small system of equations is solved to determine the correct superposition of responses to satisfy the applied surface constraints, which may be identified as a case of the capacitance matrix approach (c.f. Lagrange multipliers [Cotin et al. 99]). Since the preprocess only exploits linearity, anisotropic (and inhomogeneous) material properties can be supported [Picinbono et al. 00]. Other groups have also used the precomputed elastostatic FEM approach of [Bro-Nielsen and Cotin 96] for surgical simulation, including the KISMET surgical simulator which incorporates precomputed models to provide high-fidelity haptic force feedback [Kühnapfel et al. 99].

One limitation of the GF precomputation strategy is that incremental runtime modifications of the model require extra runtime computations. While it may be too costly for interactive applications, this can also be efficiently performed using low-rank updating techniques such as for static reanalysis in the engineering community [Kassim and Topping 87]. For surgical simulation, a practical approach has been to use a hybrid domain decomposition approach in which a more easily modified dynamic model is used in a smaller region to be cut [Cotin et al. 98, Hansen and Larsen 98].

The authors presented a interactive animation technique in [James and Pai 99] which combined precomputed GFs of boundary element models with matrix-updating techniques for fast boundary value problem (BVP) solution. The

Green's function description provides a data-driven description that subsumes discretization issues of both [James and Pai 99] and the FEM approaches of [Bro-Nielsen and Cotin 96, Cotin et al. 99]. Precomputed stiffness matrix factorizations have also been used for interactive deformation [Berkley et al. 99], and avoid the explicit superposition of Green's function quantities, but can complicate random access for multi-point haptic contact resolution.

Astley and Hayward [Astley and Hayward 98] introduced an approximation for linear viscoelastic FEM models that also exploits linearity, in this case by precomputing multilevel Norton equivalents for the system's stiffness matrix. By doing so, haptic interaction is possible by employing an explicit multirate integration scheme wherein a model associated with the contact region is integrated at a higher rate than the remaining coarser model.

Finally, local buffer models were presented by Balaniuk in [Balaniuk 00] for rendering forces computed by e.g., deformable object, simulators which can not deliver forces at fast rendering rates. An application of the technique was presented for a virtual echographic exam training simulator in [d'Aulignac et al. 00]. While we do not use the same approach here, the local buffer model concept is related to our capacitance matrix method for force computation.

1.2 Linear Elastostatic Boundary Model Preliminaries

Linear elastostatic objects are essentially three-dimensional linear springs, and as such they are useful modeling primitives for physically based simulations. The unfamiliar reader might consult a suitable background reference before continuing [Hartmann 85, Zienkiewicz 77, Brebbia et al. 84, James and Pai 99]. In this section, background material for a generic discrete GF description for a variety of precomputed linear elastostatic models is provided. Conceptually, GFs form a basis for describing all possible deformations of a LEM subject to a certain class of constraints. This is useful because it (1) provides a common language to describe all discrete LEMs, (2) subsumes extraneous discretization details by relating only physical quantities, and (3) clarifies the generality of the force feedback algorithms described later.

Another benefit of using GFs is that they provide an efficient means for exclusively simulating only boundary data (displacements and forces) if desired. While it is possible to simulate various internal volumetric quantities (see §1.2.5), simulating only boundary data involves less computation. This is sufficient since we are primarily concerned with interactive simulations that impose surface constraints and provide feedback via surface

deformation and contact forces.

1.2.1 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 possible, e.g., polyhedral, NURBS or subdivision surfaces [Schröder et al. 99].

1.2.2 Nodal Displacements and Tractions

Let the undeformed boundary be denoted by Γ. 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 (force per unit area) 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 1.1), so that the discrete displacement and traction vectors are

$$\mathsf{u} = [\mathsf{u}_1, \ldots, \mathsf{u}_n]^T \tag{1.1}$$

$$\mathsf{p} = [\mathsf{p}_1, \ldots, \mathsf{p}_n]^T, \tag{1.2}$$

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.

In order to relate traction distributions to forces, define a scalar function space, $\mathcal{L}$, on the model's boundary:

$$\mathcal{L} = \text{span}\left\{\phi_j(\mathbf{x}), \quad j = 1 \ldots n, \quad \mathbf{x} \in \Gamma\right\}, \tag{1.3}$$

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

$$\mathbf{p}(\mathbf{x}) = \sum_{j=1}^{n} \phi_j(\mathbf{x})\mathsf{p}_j, \tag{1.4}$$

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

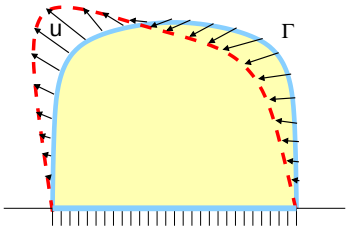


Figure 1.1. *Illustration of discrete nodal displacements* u defined at vertices on the undeformed boundary Γ (solid blue line), that result in a deformation of the surface (to dashed red line). Although harder to illustrate, a similar definition exists for the traction vector, p.

given by

$$\mathsf{f}_j = a_j \mathsf{p}_j \qquad \text{where} \qquad a_j = \int_\Gamma \phi_j(\mathbf{x}) d\Gamma_{\mathbf{x}} \tag{1.5}$$

defines the area associated with the j^{th} node.

For example, in our implementation we use linear boundary element models for which the nodes are vertices of a closed triangle mesh. The mesh is modeled as a Loop subdivision surface [Loop 87] to conveniently obtain multiresolution models for rendering as well as uniformly parameterized surfaces suitable for BEM discretization and deformation depiction. The displacement and traction fields have convenient vertex-based descriptions

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

where $\mathbf{x}_j \in \Gamma$ is the j^{th} vertex. The traction field is a piecewise linear function, and $\phi_j(\mathbf{x})$ represents a "hat function" located at the j^{th} vertex with $\phi_j(\mathbf{x}_j) = 1$. Given our implementation, we shall often refer to node and vertex interchangeably.

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

Boundary conditions arising in a force-feedback loop might consist of some 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 (1.7) yields the rendered contact forces and surface deformation.

Denote the unspecified and complementary specified nodal variables by

$$\mathsf{v}_j = \begin{cases} \mathsf{p}_j : j \in \Lambda_u \\ \mathsf{u}_j : j \in \Lambda_p \end{cases} \quad \text{and} \quad \bar{\mathsf{v}}_j = \begin{cases} \bar{\mathsf{u}}_j : j \in \Lambda_u \\ \bar{\mathsf{p}}_j : j \in \Lambda_p \end{cases}, \tag{1.6}$$

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

$$\boxed{0 = \mathsf{A}\mathsf{v} + \bar{\mathsf{A}}\bar{\mathsf{v}} = \mathsf{A}\mathsf{v} - \mathsf{z}} \tag{1.7}$$

where the BVP system matrix A and its complementary matrix $\bar{\mathsf{A}}$ are, in general, dense block n-by-n matrices [Hartmann 85]. 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 z term.

A key relationship between BVP system matrices $(\mathsf{A}, \bar{\mathsf{A}})$ of different BVP types (Λ_u, Λ_p) is that they are related by exchanges of corresponding block columns, e.g., $(\mathsf{A}_{:\mathsf{j}}, \bar{\mathsf{A}}_{:\mathsf{j}})$, and therefore small changes to the BVP type result in low-rank changes to the BVP system matrices (see §1.3.2).

While the boundary-only system matrices in (1.7) could be constructed explicitly, e.g., via condensation for FEM models [Zienkiewicz 77] or using a boundary integral formulation (see next section), it need not be in practice. The discrete integral equation in Equation 1.7 is primarily a common starting point for later definition of GFs and derivation of the CMA, while GFs may be generated with any convenient numerical method, or even robotically scanned and estimated from real objects [Pai et al. 01].

1.2.4 Example: Boundary Element Models

A simple closed-form definition of $(\mathsf{A}, \bar{\mathsf{A}})$ is possible for models discretized with the boundary element method (BEM) [Brebbia et al. 84, James and Pai 99]; BEM discretizations are possible for models with homogeneous and isotropic material properties. The surface-based nodal quantities are related by the dense linear block matrix system

$$0 = \mathsf{H}\mathsf{u} - \mathsf{G}\mathsf{p} = \sum_{j=1}^{n} \mathsf{h}_{ij}\mathsf{u}_j - \sum_{j=1}^{n} \mathsf{g}_{ij}\mathsf{p}_j \tag{1.8}$$

where G and H are n-by-n block matrices, with each matrix element, g_{ij} or h_{ij}, a 3-by-3 influence matrix with known expressions [Brebbia et al. 84].

In this case, the j^{th} block columns of A and $\bar{\mathsf{A}}$ may be identified as column exchanged variants of G and H:

$$\mathsf{A}_{:j} = \begin{cases} -\mathsf{G}_{:j} & : \quad j \in \Lambda_u \\ \mathsf{H}_{:j} & : \quad j \in \Lambda_p \end{cases} \tag{1.9}$$

$$\bar{\mathsf{A}}_{:j} = \begin{cases} \mathsf{H}_{:j} & : \quad j \in \Lambda_u \\ -\mathsf{G}_{:j} & : \quad j \in \Lambda_p \end{cases} \tag{1.10}$$

While we use BEM models for our implementation, we reiterate that the CMA is independent of the method used to generate the GFs.

1.2.5 Fast BVP Solution with Green's Functions

GFs of a single BVP type provide an economical means for solving (1.7) for that BVP, and when combined with the CMA (§1.3) will also be useful for solving other BVP types. From (1.7), the general solution of a BVP type (Λ_u, Λ_p) may be expressed in terms of discrete GFs as

$$\boxed{\mathsf{v} = \Xi\bar{\mathsf{v}} = \sum_{j=1}^{n} \xi_j \bar{\mathsf{v}}_j = \sum_{j \in \Lambda_u} \xi_j \bar{\mathsf{u}}_j + \sum_{j \in \Lambda_p} \xi_j \bar{\mathsf{p}}_j,} \tag{1.11}$$

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

$$\xi_j = -\left(\mathsf{A}^{-1}\bar{\mathsf{A}}\right)_{:j} \tag{1.12}$$

and

$$\boxed{\Xi = -\mathsf{A}^{-1}\bar{\mathsf{A}} = [\xi_1 \xi_2 \cdots \xi_n]\,.} \tag{1.13}$$

Equation (1.11) may be taken as the definition of the discrete GFs (and even (1.7)), since it is clear that the j^{th} GF simply describes the linear response of the system to the j^{th} node's specified boundary value, $\bar{\mathsf{v}}_j$. An illustration is given in Figure 1.2. Once the GFs have been computed for one BVP type, that class of BVPs may be solved easily using (1.11). An attractive feature for interactive applications is that the entire solution can be obtained in $18ns$ flops if only s boundary values (BV) are nonzero (or have changed since the last time step). Temporal coherence may also be exploited by considering the effect of individual changes in components of $\bar{\mathsf{v}}$ on the solution v.

1.2.6 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. This provides a dramatic speed-up for simulation by

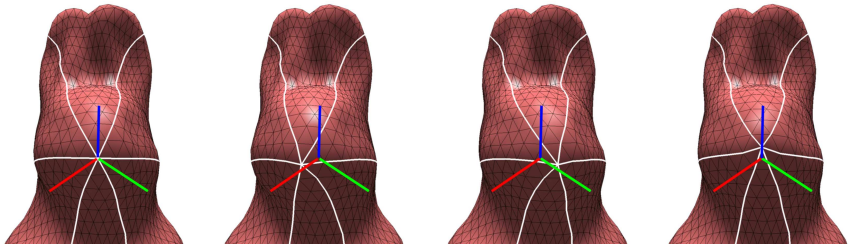


Figure 1.2. *Illustration of the j^{th} Green's function block column, $\xi_j = \Xi_{:j}$,* representing the model's response due to the three XYZ components of the j^{th} specified boundary value, $\bar{\mathsf{v}}_j$. Here the vertex belongs to the ("free") traction boundary, $j \in \Lambda_p$, and so ξ_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.

determining the deformation basis (the GFs) ahead of time. While this is not necessary a huge amount of work (see Table 1.2), the principal benefits for interactive simulations are the availability of the GF elements via cheap look-up table operations, as well as the elimination of redundant runtime computation when computing solutions, e.g., using a haptic device to grab a vertex of the model and move it around simply renders a single GF.

Once a set of GFs for a LEM 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.

1.3 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. With an improved notation and emphasis on computer haptics, this section unifies and extends the approaches presented in [James and Pai 99] exclusively for BEM models, and for FEM models in, e.g., [Bro-Nielsen and Cotin 96],

in a way that is applicable to all LEMs regardless of discretization, or origin of GFs [Pai et al. 01]. Haptic applications are considered in §1.4.

1.3.1 Reference Boundary Value Problem (RBVP) Choice

A key step in the GF precomputation process is the initial identification of a RBVP type, denoted by $(\Lambda_u^0, \Lambda_p^0)$, that is representative of the BVP types arising during simulations. For interactions with an exposed free boundary, a common choice is to have the uncontacted model attached to a rigid support as shown in Figure 1.3. The n-by-n block system matrices associated with the RBVP are identified with a subscript as A_0 and $\bar{\mathsf{A}}_0$, and the corresponding GFs are hereafter always denoted by Ξ.

Note that the user's choice of RBVP type determines which type of nodal constraints (displacement of traction) are commonly specified (in order to define Ξ), but is independent of the actual numerical boundary values $\bar{\mathsf{v}}$ used in practice. For example, there are no requirements that certain boundary values are zero, although this results in fewer summations (see (1.11)).

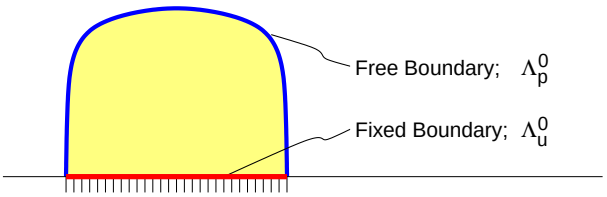


Figure 1.3. *Reference Boundary Value Problem (RBVP) example:* The RBVP associated with a model attached to a flat rigid support is shown with boundary regions having fixed (Λ_u^0) or free (Λ_p^0) nodal constraints indicated. A typical simulation would impose contacts on the free boundary via displacement constraints with the CMA.

1.3.2 Capacitance Matrix Algorithm (CMA) for BVP Solution

Precomputed GFs speed-up the solution to the RBVP, but they can also dramatically reduce the amount of work required to solve related BVP when used in conjunction with CMAs. This section describes the CMA and presents the derivation of related formulae.

Relevant Formulae Suppose the constraint-type changes, e.g., displacement↔traction, with respect to the RBVP at s nodes specified by the list of nodal indices $\mathsf{S} = \{\mathsf{S}_1, \mathsf{S}_2, \ldots, \mathsf{S}_s\}$. As mentioned earlier, it follows from (1.6) and (1.7) 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

$$\mathsf{A} = \mathsf{A}_0 + \left(\bar{\mathsf{A}}_0 - \mathsf{A}_0\right)\mathsf{E}\mathsf{E}^\mathsf{T} \tag{1.14}$$

$$\bar{\mathsf{A}} = \bar{\mathsf{A}}_0 + \left(\mathsf{A}_0 - \bar{\mathsf{A}}_0\right)\mathsf{E}\mathsf{E}^\mathsf{T} \tag{1.15}$$

where E is an n-by-s block matrix

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

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

Since the BVP solution is

$$\mathsf{v} = \mathsf{A}^{-1}\mathsf{z} = -\mathsf{A}^{-1}\bar{\mathsf{A}}\bar{\mathsf{v}}, \tag{1.16}$$

substituting (1.15) for $\bar{\mathsf{A}}$ and the Sherman-Morrison-Woodbury formula [Golub and Loan 96] for A^{-1} (using the GF definition $\Xi = -\mathsf{A}_0^{-1}\bar{\mathsf{A}}_0$),

$$\mathsf{A}^{-1} = \mathsf{A}_0^{-1} + (\mathsf{I} + \Xi)\mathsf{E}(-\mathsf{E}^\mathsf{T}\Xi\mathsf{E})^{-1}\mathsf{E}^\mathsf{T}\mathsf{A}_0^{-1}, \tag{1.17}$$

into (1.16), leads directly to an expression for the solution in terms of the precomputed GFs[2]. The resulting *capacitance matrix formulae* are

$$\boxed{\mathsf{v} = \underbrace{\mathsf{v}^{(0)}}_{n\times 1} + \underbrace{(\mathsf{E} + (\Xi\mathsf{E}))}_{n\times s}\underbrace{\mathsf{C}^{-1}}_{s\times s}\underbrace{\mathsf{E}^\mathsf{T}\mathsf{v}^{(0)}}_{s\times 1}} \tag{1.18}$$

where C is the s-by-s *capacitance matrix*, a negated submatrix of Ξ,

$$\boxed{\mathsf{C} = -\mathsf{E}^\mathsf{T}\Xi\mathsf{E},} \tag{1.19}$$

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

$$\boxed{\mathsf{v}^{(0)} = \mathsf{A}_0^{-1}\mathsf{z} = \left[\Xi\left(\mathsf{I} - \mathsf{E}\mathsf{E}^\mathsf{T}\right) - \mathsf{E}\mathsf{E}^\mathsf{T}\right]\bar{\mathsf{v}}.} \tag{1.20}$$

Algorithm for BVP Solution With Ξ precomputed, formulae (1.18)-(1.20) immediately suggest an algorithm given that only simple manipulations of Ξ and inversion of the smaller capacitance submatrix are required. An algorithm for computing *all* components of v is as follows:

[2]Similarly from [James and Pai 99] with $\delta\mathsf{A}_\mathsf{S} = (\bar{\mathsf{A}}_0 - \mathsf{A}_0)\mathsf{E}$.

- For each new BVP type (with a different C matrix) encountered, construct and temporarily store 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 v.

Complexity Issues Given s nonzero boundary values, each new capacitance matrix LU factorization involves at most $\frac{2}{3}s^3$ flops, after which each subsequent solve involves approximately $18ns$ flops ($s \ll n$). 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 §1.4), thus making CMAs predictable for real-time computations.

1.3.3 Selective Deformation Computation

A major benefit of the CMA direct BVP solver is that it is possible to just evaluate selected components of the solution vector at runtime, with the total computing cost proportional to the number of components desired, i.e., output-sensitive evaluation. This is a key enabling feature for force feedback where, e.g., contact forces are desired at different rates than the geometric deformations. Selective evaluation would also be useful for optimizing (self) collision detection queries, avoiding simulation of occluded or undesired portions of the model, as well as rendering adaptive level of detail representations.

In general, any subset of solution components may be determined at a smaller cost than computing v entirely. Let the solution be desired at nodes specified by the set of indices D, with the desired components of v extracted by $\mathsf{E}_\mathsf{D}^\mathsf{T}$. Using (1.18), the selected solution components may be evaluated as

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

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 in the following section.

1.3.4 Extensions

Several extensions exist to overcome various bottlenecks of the CMA algorithm. First, due to the dense nature of the GF matrix, computation and storage issues arise for large models. The $O(sn)$ cost of GF summation for surface deformation can be a bottleneck, although to some extent fast native BLAS or graphics hardware implementations (as in [Barbič and James 05]) can help. A fast summation algorithm based on fast lifted wavelet transforms was proposed in [James and Pai 03] to alleviate the summation bottleneck, and it also provides practical memory requirements for large models.

Second, the $O(s^3)$ capacitance matrix inversion/factorization step can become a bottleneck for large contact regions. By exploiting temporal coherence common in contact problems, the $O(s^3)$ cost can be reduced to $O(s^2\ \Delta s)$ where Δs is the number of changes (additions or deletions) to the contact node set. Details on the updating/downdating procedures are given in [James 01]. For temporally coherent cases where $\Delta s \ll s$, such as in grasping scenarios (see Figure 1.4), the overhead of updating the capacitance matrix inverse can often out-perform LU factorization of C. Another way to reduce contact updating complexity is to coarsen (or adapt) the contact resolution, and hierarchical (wavelet) GFs were introduced in [James and Pai 03] for this purpose. Unfortunately, coarsened contacts also limit the ability to resolve contact regions unless adaptivity is used.

1.4 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 [Balaniuk 00], we say that the capacitance matrix can be used as a local buffer model. While the capacitance matrix is used in §1.3.2 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 relevant benefit for haptics is that the local contact force response may be computed at a much faster rate than the global deformation.

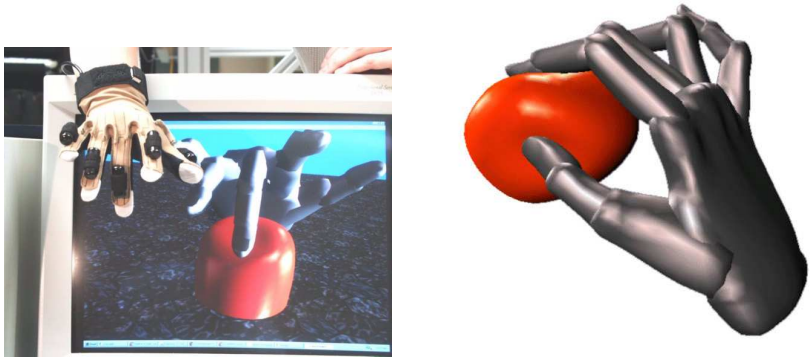


Figure 1.4. *Grasping simulation:* Using a CyberTouch data input device from Virtual Technologies Inc. (Top), a virtual hand (Bottom) was used to deform an elastostatic BEM model with approximately 900 surface degrees of freedom (dof) at graphical frame rates (>30 FPS) on a PII 450MHz computer in Java JDK 1.3. 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, the capacitance matrices computed could also have been used to render contact forces at a rate higher than that of the graphical simulation.

1.4.1 Capacitance Matrix Local Buffer Model

From (1.18), the S components of the solution v are

$$\begin{aligned}
\mathsf{E}^{\mathsf{T}}\mathsf{v} &= \mathsf{E}^{\mathsf{T}}\left[\mathsf{v}^{(0)} + (\mathsf{E} + (\Xi\mathsf{E}))\,\mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)}\right] \\
&= \mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} + \underbrace{(\mathsf{E}^{\mathsf{T}}\mathsf{E})}_{\mathsf{I}}\,\mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} + \underbrace{(\mathsf{E}^{\mathsf{T}}\Xi\mathsf{E})}_{-\mathsf{C}}\,\mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} \\
&\downarrow \qquad\qquad \text{(from (1.19))} \\
&= \mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} + \mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} - \mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)} \\
&= \mathsf{C}^{-1}\left(\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)}\right). \qquad (1.21)
\end{aligned}$$

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

$$\bar{\mathsf{v}} = \mathsf{E}\mathsf{E}^{\mathsf{T}}\bar{\mathsf{v}} \quad \Rightarrow \quad \mathsf{v}^{(0)} = -\bar{\mathsf{v}} \quad \text{(using (1.20))}. \qquad (1.22)$$

In this case, the capacitance matrix completely characterizes the local contact response, since (using (1.22) in (1.21))

$$\mathsf{E}^{\mathsf{T}}\mathsf{v} = -\mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}}\bar{\mathsf{v}}. \qquad (1.23)$$

This in turn parametrizes the global response since these components (not in S) are

$$\begin{aligned}(\mathsf{I}-\mathsf{E}\mathsf{E}^{\mathsf{T}})\mathsf{v} &= (\mathsf{I}-\mathsf{E}\mathsf{E}^{\mathsf{T}})\left[\mathsf{v}^{(0)}+(\mathsf{E}+(\Xi\mathsf{E}))\,\mathsf{C}^{-1}\mathsf{E}^{\mathsf{T}}\mathsf{v}^{(0)}\right]\\ &= (\mathsf{I}-\mathsf{E}\mathsf{E}^{\mathsf{T}})(\Xi\mathsf{E})(\mathsf{E}^{\mathsf{T}}\mathsf{v}) \qquad (1.24)\end{aligned}$$

where we have used (1.23) and the identity $(\mathsf{I}-\mathsf{E}\mathsf{E}^{\mathsf{T}})\mathsf{E}=0$. Such properties allow the capacitance matrix and Ξ to be used to derive efficient local models for surface contact.

For example, given the specified contact zone displacements

$$\mathsf{u}_{\mathsf{S}} = \mathsf{E}^{\mathsf{T}}\bar{\mathsf{v}}, \qquad (1.25)$$

the resulting tractions are

$$\mathsf{p}_{\mathsf{S}} = \mathsf{E}^{\mathsf{T}}\mathsf{v} = -\mathsf{C}^{-1}\left(\mathsf{E}^{\mathsf{T}}\bar{\mathsf{v}}\right) = -\mathsf{C}^{-1}\mathsf{u}_{\mathsf{S}}, \qquad (1.26)$$

and the rendered contact force is

$$\mathbf{f} = \mathsf{a}_{\mathsf{S}}^{\mathsf{T}}\mathsf{p}_{\mathsf{S}} = \left(-\mathsf{a}_{\mathsf{S}}^{\mathsf{T}}\mathsf{C}^{-1}\right)\mathsf{u}_{\mathsf{S}} = \mathsf{K}_{\mathsf{S}}\mathsf{u}_{\mathsf{S}}, \qquad (1.27)$$

where K_{S} is the effective stiffness of the contact zone used for force feedback rendering,

$$\mathsf{a}_{\mathsf{S}} = \left(a_{\mathsf{S}_1}, a_{\mathsf{S}_2}, \ldots, a_{\mathsf{S}_{\mathsf{s}}}\right)^{\mathsf{T}} \otimes I_3 \qquad (1.28)$$

represents nodal areas (1.5), and I_3 is the scalar 3-by-3 identity matrix. A similar expression may be obtained for torque feedback. The visual deformation corresponding to solution components outside the contact zone is then given by (1.24) using $\mathsf{p}_{\mathsf{S}}=\mathsf{E}^{\mathsf{T}}\mathsf{v}$.

1.4.2 Example: Single Displacement Constraint

A simple case, which will be discussed in much greater detail in §1.5, is that of imposing a displacement constraint on single a node k which otherwise had a traction constraint in the RBVP. This case occurs, for instance, when the tip of a haptic device comes into contact with the free surface of an object. The new BVP therefore has only a single constraint switch with respect to the 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

$$\mathsf{p}_k = -\mathsf{C}^{-1}\bar{\mathsf{u}}_k = \left(\Xi_{kk}\right)^{-1}\bar{\mathsf{u}}_k \qquad \text{or} \qquad \bar{\mathsf{u}}_k = \Xi_{kk}\mathsf{p}_k.$$

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 $\mathsf{v}=\xi_k\mathsf{p}_k$.

1.4.3 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 Figure 1.4 will depend on how the other fingers are supporting the object. For multiple contacts like this, the capacitance matrix again provides an efficient force response model for haptics. Without presenting the equations in detail, we shall just mention that the force responses for each of the contact patches can be derived from the capacitance matrix in a manner similar to equations (1.25)-(1.28).

1.5 Surface Stiffness Models for Point-like Contact

The second part of this chapter presents a simple and practical method for describing point-like contact interactions. Such interactions are commonly in the haptics literature for *rigid* surface models [Massie and Salisbury 94, Ho et al. 99]. Unlike their rigid counterparts, special care must be taken with elastic models to define *finite contact areas* for point-like interactions since point-like contacts defined only as single-vertex (§1.4.2) or nearest neighbour [Cotin et al. 99] constraints lead to mesh-related artifacts, and ambiguous interactions as the mesh is refined (see Figure 1.5). However, the benefit of point-like contacts comes from the convenience of the point-like parameterization of the contact and not because the contact is highly concentrated or "pin-like". We present an approach using *vertex pressure masks* which maintains the single contact description yet distribute 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 spatial variation on irregular meshes.

1.5.1 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.

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 $\phi_i(\mathbf{x})$.

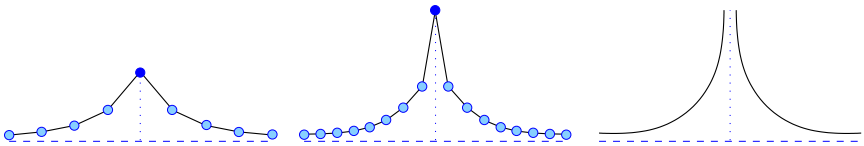


Figure 1.5. *Point contact must not be taken literally for elastic models:* This figure illustrates the development of a displacement singularity associated with a concentrated surface force as the continuum limit is approached. In the left image, an upward unit force applied to a vertex of a discrete 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 1.1). Such point-like constraints are mathematically ill-posed for linear models based on a small-strain assumption, and care must be taken to meaningfully define the interaction.

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

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

which in turn implies the normalization condition

$$\int_{\Gamma} \rho(\mathbf{x}) d\Gamma_{\mathbf{X}} = 1. \tag{1.29}$$

In the discrete setting, the piecewise linear surface density on Γ is

$$\rho(\mathbf{x}) = \sum_{j=1}^{n} \phi_j(\mathbf{x})\rho_j \in \mathcal{L}, \tag{1.30}$$

and is parameterized by the discrete scalar vertex mask vector,

$$\rho = [\rho_1, \rho_2, \ldots, \rho_n]^T.$$

Substituting (1.30) into (1.29), the discrete normalization condition satisfied becomes

$$a^T \rho = 1, \tag{1.31}$$

where a are the vertex areas from (1.5). Notice that the mask density ρ has units of $\frac{1}{\text{area}}$.

In practice, the vertex pressure mask ρ 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 ρ may then be defined using nodal collocation (see Figure 1.6),

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

followed by suitable normalization,

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

to ensure the satisfaction of (1.31).

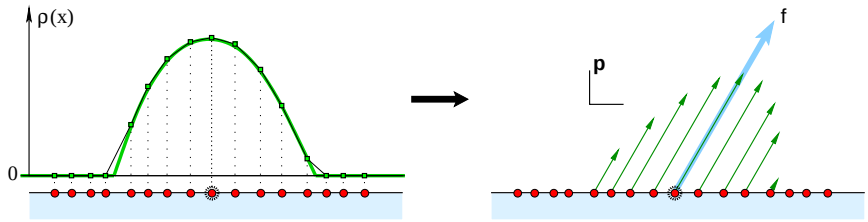


Figure 1.6. *Collocated scalar masks:* A direct means for obtaining a relative pressure amplitude distribution about each node, is to employ a user-specified scalar functional of the desired spatial scale. The scalar pressure mask is then given by nodal collocation (left), after which the vector traction 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).

In the following, denote the density mask for the i^{th} vertex by the n-vector ρ^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 1.7 and 1.8) had linear radial dependence,

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

where r specifies the radial scale, and is representative of the haptic probe's tip. The effect of changing r is shown in Figure 1.7.

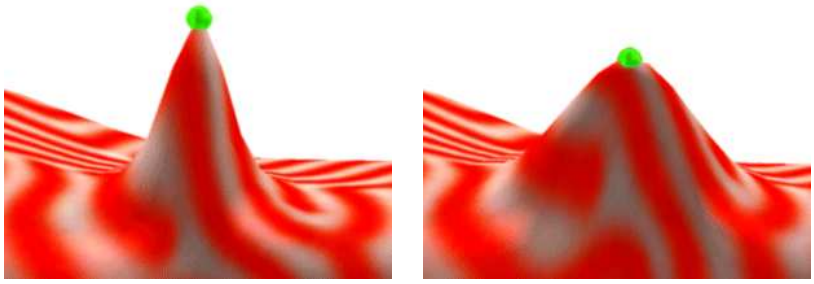
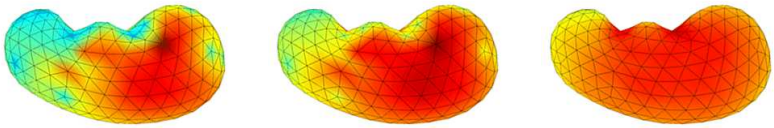


Figure 1.7. *Illustration of changing mask scale:* An exaggerated pulling deformation illustrates different spatial scales in two underlying traction distributions. In each case, pressure masks were generated using the linear spherical mask functional (see §1.5.1) for different values of the radius parameter, r.



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

Figure 1.8. *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 §1.5.1) of radius twice the mesh's mean edge length) and better approximates the regular force response expected of such a model. Masks essentially provide anti-aliasing for stiffnesses defined with discrete traction distributions, and help avoid "soft spots."

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

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

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

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

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

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

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

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

therefore the effective elastic stiffness of the masked vertex is

$$\boxed{\mathsf{K}_i = \mathsf{K}_i^{\mathsf{E}} = \left(\sum_{j \in \mathcal{M}_i} \rho_j^i \Xi_{ij} \right)^{-1}, \quad i \in \Lambda_p^0.} \tag{1.32}$$

Some examples are provided in Table 1.1 and Figure 1.8.

Therefore, in the simple case of a single masked vertex displacement constraint 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

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

where ζ_i is the convolution of the GFs with the mask ρ, 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 $\zeta_i = \xi_i / a_i$.

Rigid Vertex Stiffness, K^{R} For rigid surfaces a finite force response may be defined using an isotropic stiffness matrix,

$$\mathsf{K}^{\mathsf{R}} = k^{\text{Rigid}} I_3 \in \mathbb{R}^{3\times 3}, \qquad k^{\text{Rigid}} > 0.$$

Mesh Level	Vertices	$\|\mathsf{K}\|_F$, Single	$\|\mathsf{K}\|_F$, Masked
1	34	7.3	13.3
2	130	2.8	11.8
3	514	1.1	11.2

Table 1.1. *Vertex stiffness dependence on mesh resolution:* This table shows vertex stiffness (Frobenius) norms (in arbitrary units) at the top center vertex of the BEM model in Figure 1.11(a), as geometrically modeled using Loop subdivision meshes for three different levels of resolution. 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 §1.5.1) with radius equal to the coarsest (level 1) mesh's mean edge length) has substantially less mesh dependence, and quickly approaches a nonzero value.

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

$$\mathsf{K}_i = \mathsf{K}^{\mathsf{R}}, \quad i \in \Lambda_u^0, \tag{1.33}$$

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 §1.5.3 that defining stiffness responses at these nodes is important for determining contact responses on neighbouring triangles which are not rigid.

1.5.3 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

$$\mathsf{K}(\mathbf{x}) = \sum_{i=1}^{n} \phi_i(\mathbf{x})\mathsf{K}_i, \quad \mathbf{x} \in \Gamma, \tag{1.34}$$

so that $(\mathsf{K}(\mathbf{x}))_{kl} \in \mathcal{L}$. Note that there are no more than three nonzero terms in the sum of (1.34), 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^{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 $\bar{\mathsf{u}}$ applied at $\mathbf{x} \in \Gamma$ on a triangle having vertex indices $\{i_1, i_2, i_3\}$, the corresponding global solution is

$$\mathsf{v} = \sum_{i \in \{i_1,i_2,i_3\} \cap \Lambda_p^0} \zeta_i \phi_i(\mathbf{x})\mathbf{f}. \tag{1.35}$$

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.

1.5.4 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) have forces displayed as softer materials during continuous contact. This can be achieved using a *haptic vertex stiffness*, $\mathsf{K}_i^{\mathsf{H}}$, which is proportional to the elastic vertex stiffness, $\mathsf{K}_i^{\mathsf{E}}$. 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

$$\mathsf{K}_i^{\mathsf{H}} = \eta_i \mathsf{K}_i^{\mathsf{E}} \quad \text{where} \quad \eta_i = \min\left(1, \frac{\|\mathsf{K}^{\mathsf{R}}\|}{\|\mathsf{K}_i^{\mathsf{E}}\|}\right),$$

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 using (1.34), so that $\|\mathsf{K}^{\mathsf{H}}(\mathbf{x})\| \leq \|\mathsf{K}^{\mathsf{R}}\|, \forall \mathbf{x} \in \Gamma$.

In accordance with force-reflecting contact, the deformed elastic state corresponds to the haptic force applied at the contact location $\mathbf{x}^{\mathsf{C}}$. This produces geometric contact configurations similar to that shown in Figure 1.9, where the haptic displacement u^{H} can differ from the elastic displacement u^{E}. The geometric deformation is determined from the applied force $\mathbf{f}$ and equation (1.35). 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" [Zilles and Salisbury 94] or "surface contact point" [Sensable Technologies, Inc.] is defined as the parametric image of $\mathbf{x}^{\mathsf{C}}$ on the deformed surface.

1.6 Results

GFs were precomputed using the boundary element method (BEM) with piecewise linear boundary elements. Table 1.2 provides timings for the BEM precomputation stages as well as the submillisecond cost of simulating point-like deformations using GFs. Further timings of CMA suboperations are shown in Table 1.3, and reflect interactive performance for modest numbers of constraint type changes, s. All timings were performed using the same unoptimized Java code as in the original paper [James and Pai 01], however were re-run on a single core of a Intel Core Duo (T2700 2.33GHz), with 2GB RAM, and Sun's Java 1.6.0 server JVM (for Windows); these

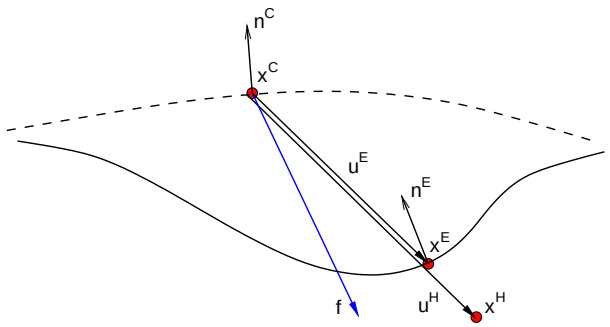


Figure 1.9. *Geometry of point-like contact:* The surface of the static/undeformed geometry (curved dashed line) and that of the deformed elastic model (curved solid line) are shown along with: applied force ($\mathbf{f}$), static contact location (x^C), deformed elastic model contact location (x^E), haptic probe-tip location (x^H), haptic contact displacement ($u^H = x^H - x^C$), elastic contact displacement ($u^E = x^E - x^C$), static contact normal (n^C) and elastic contact normal (n^E). Once the contact is initiated by the collision detector, the sliding frictional contact can be tracked in surface coordinates at force feedback rates.

timings are roughly an order-of-magnitude faster than in the original paper. Obviously model complex models and contact scenarios are now possible. These times can be reduced further by using optimized matrix libraries.

An application of the CMA for multiple distributed contacts with unilateral contact constraints was the grasping task illustrated in Figure 1.4 using the LEM from Figure 1.11(a).

Model	# Vertices, n	# Faces	Precomp	LUD %	Simulate
Nodule	130v (89 free)	256f	0.052 min	16%	10 μsec
Kidney	322v (217 free)	640f	0.43 min	16%	25 μsec
Spatula	620v (559 free)	1248f	2.7 min	12%	64 μsec
Banana Seat	546v (245 free)	1088f	1.4 min	23%	28 μsec

Table 1.2. *GF precomputation and simulation times* for the BEM models depicted in Figure 1.11. All GFs corresponding to moveable free vertices (in Λ_p^0) were computed, and the precomputation time (Precomp) of the largest model is less than an hour. As is typical of BEM computations for models of modest size ($n < 1000$), the $\mathcal{O}(n^2)$ construction of the matrices (H and G in equation 1.8) 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 submillisecond graphics-loop computations (Simulate) are required to determine the point-contact deformation response of each model's free boundary–primarily a rank-9 summation.

A force-feedback implementation of the point-like contact approach dis-

# Updates, s	LU Factor	LU Solve	$(\Xi\mathsf{E})(\mathsf{E}^\mathsf{T}\bar{\mathsf{v}})$ for $n=100$
10	0.08 ms	3 μsec	37 μsec
20	0.43 ms	11 μsec	77 μsec
40	2.59 ms	42 μsec	152 μsec
100	40.0 ms	230 μsec	382 μsec

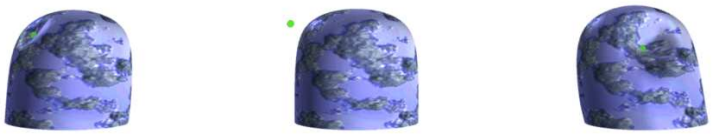
Table 1.3. *Timings of CMA suboperations* such as LU decomposition (LU Factor) and back-substitution (LU Solve) of the capacitance matrix, as well as the weighted summation of s GFs (per 100 nodes) are shown for different sizes of updated nodal constraints, s.

cussed in the previous section was built. Forces were rendered by a 3-DoF PHANToM™ haptic interface (model 1.0 Premium), on a dual Pentium II computer running Windows NT. 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™. The frictional point-contact problem was computed by the haptic servo loop at 1 kHz, which then prescribes boundary conditions for the slower graphical simulation 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 authors demonstrating the simulation is shown in Figure 1.10, and a number of screen shots for various models are shown in Figure 1.11.

Figure 1.10. *: Photograph of simulation in use:* Users were able to push, slide and pull on the surface of the model using a point-like manipulandum.

We observed that the vertex masks were successful in producing noticeable improvements in the smoothness of the sliding contact force, especially when passing over regions with irregular triangulations (see Figure 1.8). 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 two conferences: the 10^{th} Annual PRECARN-IRIS (Institute for Robotics and Intelligent Systems) Conference (Montreal, Quebec, Canada, May 2000) and in the ACM SIGGRAPH 2000 Exhibition (New Orleans, Louisiana, USA, July 2000). Users reported that the simulation felt realistic. In general, the precomputed LEM approach was found to be both stable and robust.



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

(b) A kidney-shaped model with position-constrained vertices on 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 1.11. *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 §1.5. For each model, the middle of the three figures is uncontacted by the user's interaction point (a small green ball).

1.7 Summary

We have summarized an approach for real-time solution of boundary value problems for discrete linear elastostatic models (LEM), regardless of discretization, using precomputed GFs in conjunction with capacitance matrix algorithms (CMAs). The data-driven CMA formulation highlights the special role of the capacitance matrix in computer haptics as a contact compliance useful for generating contact force and stiffness models, and provides a framework for extending the capabilities of these models.

Additionally, the important special case of point-like contact was addressed with special attention given to the consistent definition of contact forces for haptics. While this topic has been discussed before, we have introduced vertex masks to specify the distribution of contact forces in a way which leads to physically consistent force-feedback models which avoid the numerical artifacts which lead to nonsmooth rendering of contact forces on discrete models, as wells as ill-defined contacts in the continuum limit.

Epilogue: Green's function models are particularly effective for linear elastostatic models, however their use is limited for large-deformation models (although see [James and Pai 02] for articulated models). At the time of this writing, we have been investigating alternative basis-superposition methods for haptic rendering that are based on dimensional model reduction and precomputed large-deformation modal models. We refer the reader to on-going work for 6-DoF haptic rendering of multi-point contact between geometrically complex models [Barbič and James 07].

Bibliography

[Astley and Hayward 98] 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.

[Balaniuk 00] Remis Balaniuk. "Building a Haptic Interface based on a Buffer Model." In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[Barbič and James 05] Jernej Barbič and Doug James. "Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models." *ACM Transactions on Graphics* 24:3 (2005), 982–990.

[Barbič and James 07] Jernej Barbič and Doug L. James. "Time-critical distributed contact for 6-DoF haptic rendering of adaptively sampled reduced deformable models." In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2007. (in press).

[Berkley et al. 99] J. Berkley, S. Weghorst, H. Gladstone, G. Raugi, D. Berg, and M. Ganter. "Fast Finite Element Modeling for Surgical Simulation." In *Proceedings of Medicine Meets Virtual Reality*, pp. 55–61, 1999.

[Brebbia et al. 84] C. A. Brebbia, J. C. F. Telles, and L. C. Wrobel. *Boundary Element Techniques: Theory and Applications in Engineering*, Second edition. New York: Springer-Verlag, 1984.

[Bro-Nielsen and Cotin 96] Morten Bro-Nielsen and Stephane Cotin. "Real-time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation." *Computer Graphics Forum* 15:3 (1996), 57–66.

[Bro-Nielsen 96] Morten Bro-Nielsen. "Surgery Simulation Using Fast Finite Elements." *Lecture Notes in Computer Science* 1131 (1996), 529–.

[Cotin et al. 98] Stephane Cotin, Herve Delingette, and Nicholas Ayache. "Efficient Linear Elastic Models of Soft Tissues for Real-time Surgery Simulation." Technical Report RR-3510, Inria, Institut National de Recherche en Informatique et en Automatique, 1998.

[Cotin et al. 99] 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 (1999), 62–73.

[d'Aulignac et al. 00] 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.

[Ezawa and Okamoto 89] 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*, pp. 405–416, 1989.

[Golub and Loan 96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*, Third edition. Baltimore and London: Johns Hopkins University Press, 1996.

[Hager 89] William W. Hager. "Updating the inverse of a matrix." *SIAM Review* 31:2 (1989), 221–239.

[Hansen and Larsen 98] 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 (1998), 305–.

[Hartmann 85] Friedel Hartmann. *The mathematical foundation of structural mechanics*. New York: Springer-Verlag, 1985.

[Ho et al. 99] C.-H. Ho, C. Basdogan, and M. A. Srinivasan. "Efficient point-based rendering techniques for haptic display of virtual objects." *Presence* 8:5 (1999), 477–491.

[James and Pai 99] Doug L. James and Dinesh K. Pai. "ArtDefo: Accurate Real Time Deformable Objects." *Computer Graphics* 33:Annual Conference Series (1999), 65–72.

[James and Pai 01] Doug L. James and Dinesh K. Pai. "A Unified Treatment of Elastostatic and Rigid Contact for Real-Time Haptics." *Haptics-e, The Electronic Journal of Haptics Research (www.haptics-e.org)* 2:1.

[James and Pai 02] Doug L. James and Dinesk K. Pai. "Real-Time Simulation of Multizone Elastokinematic Models." In *2002 IEEE Intl. Conference on Robotics and Automation. Washington DC*, 2002.

[James and Pai 03] Doug L. James and Dinesh K. Pai. "Multiresolution Green's Function Methods for Interactive Simulation of Large-scale Elastostatic Objects." *ACM Transactions on Graphics* 22:1 (2003), 47–82.

[James 01] Doug L. James. "Multiresolution Green's Function Methods for Interactive Simulation of Large-scale Elastostatic Objects and Other Physical Systems in Equilibrium." Ph.D. thesis, University of British Columbia, Vancouver, British Columbica, Canada, 2001.

[Kassim and Topping 87] A. M. Abu Kassim and B. H. V. Topping. "Static Reanalysis: a review." *Journal of Structural Engineering* 113 (1987), 1029–1045.

[Kühnapfel et al. 99] U. Kühnapfel, H.K. Çakmak, and H. Maaß. "3D Modeling for Endoscopic Surgery." In *Proceedings of IEEE Symposium on Simulation*, pp. 22–32. Delft University, Delft, NL, 1999.

[Loop 87] Charles Loop. "Smooth Subdivision Surfaces Based on Triangles." Master's thesis, University of Utah, Department of Mathematics, 1987.

[Man et al. 93] K. W. Man, M. H. Aliabadi, and D. P. Rooke. "Analysis of Contact Friction using the Boundary Element Method." In *Computational Methods in Contact Mechanics*, edited by M. H. Aliabadi and C. A. Brebbia, Chapter 1, pp. 1–60. Computational Mechanics Publications and Elsevier Applied Science, 1993.

[Massie and Salisbury 94] 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, 1994.

[Pai et al. 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.

[Picinbono et al. 00] G. Picinbono, J. C. Lombardo, H. Delingette, and N. Ayache. "Anisotropic Elasticity and Force Extrapolation to Improve Realism of Surgery Simulation." In *Proceedings of IEEE International Conference on Robotics and Automation*. San Francisco, USA, 2000.

[Press et al. 87] 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, pp. 66–70. Cambridge: Cambridge University Press, 1987.

[Schröder et al. 99] P. Schröder, D. Zorin, T. DeRose, D. R. Forsey, L. Kobbelt, M. Lounsbery, and J. Peters. "Subdivision for Modeling and Animation." SIGGRAPH 99 Course Notes, 1999.

[Sensable Technologies, Inc.] Sensable Technologies, Inc. "GHOST SDK, http://www.sensable.com."

[Zienkiewicz 77] O. C. Zienkiewicz. *The Finite Element Method*. Maidenhead, Berkshire, England: McGraw-Hill Book Company (UK) Limited, 1977.

[Zilles and Salisbury 94] 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*, 1, 1, pp. 149–150. Chicago, IL (US), 1994.