Dynamic Animation of Elastic Bodies

Rogério L. W. Liesenfeld[1]
Jorge Stolfi[2]

Institute of Computing (IC)
State University of Campinas (UNICAMP)
PO Box 6065 – 13081-970 Campinas, San Paulo, Brazil
{rluis,stolfi}@dcc.unicamp.br

Abstract. We describe an animation system that simulates the dynamics of viscoelastic bodies subject to equality and inequality constraints. The equations of motion are derived from Lagrange's equation, and constraint forces are computed by the method of Lagrange multipliers. Each elastic body is modeled as a collection of tetrahedral finite elements whose deformation is restricted to affine transformations of their rest shapes. We use an original non-linear formula for the elastic forces, especially devised to prevent elements from collapsing to zero or negative volume. We also describe a general algorithm to detect violation of inequality constraints. For collisions, in particular, we use the optimization technique of Lin and Manocha to cut the detection time from quadratic to almost linear. Collisions are handled by temporary contact springs.

1 Introduction

The kinematic techniques still used in most commercial animation systems leave to the animator the task of estimating the object motions according to the laws of physics. Physically-based simulation offers a promising alternative.

We describe here an animation system that simulates the dynamic behavior of elastic bodies, according to the laws of Newtonian mechanics. In order to allow large deformations without allowing the bodies to be compressed to volume zero or negative, we adopt a model for the elastic forces that is non-linear in the deformation measures. This model, nevertheless, reduces to Hooke's linear model for small deformations.

We allow for constraints on the positions of the bodies, expressed as algebraic equalities on their coordinates. These constraints can be used to keep a body fixed, or force it to follow a predefined trajectory. Our system also detects and handles collisions between objects (or different parts of the same object), which are a necessary feature of almost any realistic animation.

1.1 Related work

Our work is mostly related to that of Terzopoulos and others [12, 11]. Alternative approaches, which restrict the deformations in order to reduce the simulation cost, have been proposed by Pentland and Williams [10], and Witkin and Welch [14].

Witkin, Gleicher and Welch [13], among others, described the handling of constraints by Lagrange multipliers. For collision detection, we rely on Lin and Manocha's optimization technique [7]. The handling of collisions by springs was already proposed by Moore and Wilhelms [9]. Newer (and still experimental) approaches are Mirtich and Canny's microimpulse model [8], and Baraff's detailed analysis of the "classical" impulse and contact force model [1, 2].

1.2 Notation

We denote a row vector by $[u_1, u_2, \ldots, u_m]$. A point u of $\mathbf{R}^m$ (an *m-vector*) is by definition a column vector.

If f is a scalar function of a vector $u \in \mathbf{R}^m$, we denote by $\partial f/\partial u$ the vector $[\partial f/\partial u_1, \ldots, \partial f/\partial u_m]^\top$; and by $\partial^2 f/\partial u^2$ the $m \times m$ matrix whose element in row i and column j is $\partial^2 f/\partial u_i \partial u_j$, for $i, j = 1, \ldots, m$.

If v is any property of the system that changes with time, we denote by $v(t)$ its value at instant t and by v' and v'' its first and second derivatives with respect to time.

2 Equations of motion

For the purposes of this section, a *dynamic system* is a collection of point-like material particles that move in space in response to internal and external forces, each according with Newton's law $F = ma$.

[1]Bolsa de mestrado FAPESP 94/4132-6.
[2]Bolsa de auxilio a pesquisa CNPq 301016/92-5.

2.1 Generalized coordinates

The *configuration* of a dynamic system at a given instant consists of the positions in $\mathbf{R}^3$ of all its material particles. The *state* of the system consists of the current positions and velocities of those particles.

In practice, we must use a simplified model of the system, where only the most important degrees of freedom in the motion of the particles are represented. For example, to model the motion of a rigid object, it is usually sufficient to keep track of the position of its center of mass, and the orientation of an orthogonal frame fixed on the body.

So, suppose we have a simplified model of the system, and $q_1, \ldots, q_n$ are n real-valued parameters whose values at any instant t completely determine the model's configuration at t. The particle velocities at t are then completely determined by the quantities $q_1, \ldots, q_n$ and their time derivatives $q'_1, \ldots, q'_n$. One says that the former are a set of *generalized coordinates* for the system, and the latter are the corresponding *generalized velocities*.

We stretch the language a bit and say that the n-vector $q = [q_1, \ldots, q_n]^\top$ *is* the configuration of the system; and the pair (q, q') *is* its state. Note that both q and q' are functions of time.

Given a collection of forces $f_1, \ldots, f_m$ acting on the m particles of the system, one defines the corresponding *generalized force* E_i acting on each generalized coordinate q_i, such that the work done by those forces (assumed constant) when the coordinates q change by a small vector δ will be $E^\top \delta$.

2.2 Lagrange's equation

It follows from the laws of classical mechanics that the evolution of a dynamical system is completely determined by its initial state and the forces applied by the environment over time. *Lagrange's equation* [6] is a general differential formula that determines the system's evolution, in terms of an arbitrary system of generalized coordinates, from the formulas that express the energy of the system in those coordinates.

The internal energy of a dynamic system can be written as the sum of the *kinetic energy* of its particles, and a *potential energy* due to the forces between particles. The energy of the system may change due to *external forces* applied by the environment on the particles, or by *internal friction* due to forces between particles that resist their relative motion.

The potential energy of the system depends on the particle's positions alone, so it is a function P of the coordinate vector q. The kinetic energy depends only on the particle velocities, so it can be written as a function K of q and q'. The rate of energy loss due to internal friction depends on the positions and velocities of the particles, and therefore it can be expressed as a function W of q and q'.

Lagrange's equation, which is ultimately derived from Newton's law, states that the evolution of the state (q, q') satisfies

$$\frac{d}{dt}\left(\frac{\partial K}{\partial q'}(q, q')\right) + \frac{\partial W}{\partial q'}(q, q') + \frac{\partial P}{\partial q}(q) = E \quad (1)$$

where $E = [E_1, \ldots, E_n]^\top$ is the vector of generalized forces applied on the system by the environment.

Expanding the derivatives of (1) and rearranging the terms, we get the matrix form of Lagrange's equation,

$$\mathrm{M}q'' = F \quad (2)$$

where M is the *generalized mass matrix*, and F is the *generalized total force vector*, defined as

$$\mathrm{M}_{ij} = \frac{\partial^2 K}{\partial q'_i \partial q'_j} \quad (3)$$

$$F = E - \frac{\partial^2 K}{\partial q' \partial q} q'(q, q') - \frac{\partial W}{\partial q}(q, q') - \frac{\partial P}{\partial q}(q) \quad (4)$$

Equations (2–4) allow us to compute the accelerations q'' for a state (q, q'), given the external force vector E. The system's evolution can be determined by integrating the second-order differential equation $q'' = \mathrm{M}^{-1}F$, where F is computed from q, q', and E.

3 Constraints

Lagrange's formula (1) can be used only when the generalized coordinates q_i are independent and non-redundant; that is, when the set of allowed configurations for the system has dimension exactly n.

In many situations, however, this requirement is too restrictive. In practice, we generally use a *redundant* set of generalized coordinates, together with one or more *constraints* that restrict them to some lower-dimensional manifold of *valid configurations*. For example, in the case of a particle restricted to move on the unit sphere $\mathbf{S}^2$, we could use the Cartesian coordinates (x, y, z) of the particle, together with the constraint $x^2 + y^2 + z^2 - 1 = 0$.

Constraints typically arise in systems that consist of several solid bodies in contact or connected by mechanical joints, whether among themselves or to the external environment. As a rule, the only practical way to model such systems is to model each part independently, concatenate the coordinate vectors of all parts, and subject the resulting vector to the equations implied by the additional constraints.

In the context of computer animation, constraints can be used also to keep an object fixed in space, or drag it along a prescribed trajectory.

In general, suppose we have a set of constraints on the system's configuration that can be expressed by equations $\Phi_r(q,t) = 0$, for $r = 1, \ldots, k$. In order to keep these equations satisfied, the constraining processes must apply appropriate *constraint forces* on the system. The corresponding generalized forces must be added to the right-hand side of Lagrange's equation (1).

3.1 Constraints as springs

There are two main approaches for computing these forces. The simplest, and perhaps most intuitive, is to model each constraint $\Phi_r(q,t) = 0$ by a spring whose stretching energy is $K\Phi_r(q,t)^2$, for some constant $K > 0$. This approach allows the constraint to be slightly violated, but the spring will automatically provide a force that tends to restore the constraint. By increasing the spring stiffness K, the magnitude of the violations can be made as small as desired. The drawback of this method is that the presence of stiff springs makes the differential equation unstable.

3.2 Exact constraint forces

Another approach consists of directly computing the generalized constraint force vector C that is needed to exactly fulfill the constraints at each instant. Given a constraint equation $\Phi_r(q,t) = 0$, let h_r be a function of time that records the value of the left-hand side throughout the evolution of the system. Satisfying the constraint means ensuring that $h_r = 0$ at all times. If we start from a valid state, we must have $h_r = 0$ and $h_r' = 0$ at the initial moment. To maintain these conditions, we need only to ensure that the acceleration q'', at every instant, is such that h_r'' is always zero. This requirement contributes one linear equation relating q'' to known quantities, namely

$$\frac{\partial \Phi_r}{\partial q}(q,t)^\top q'' = \psi_r \tag{5}$$

where

$$\psi_r = -q'^\top \frac{\partial^2 \Phi_r}{\partial q^2}(q,t)\, q' - 2\frac{\partial^2 \Phi_r}{\partial q \partial t}(q,t)^\top q' - \frac{\partial^2 \Phi_r}{\partial t^2}(q,t) \tag{6}$$

The matrix formulation (2) is then replaced by

$$\begin{cases} \mathrm{M}q'' &= F + C \\ \mathrm{N}q'' &= \psi \end{cases} \tag{7}$$

where C is an n-vector of unknown constraint forces, N is a $k \times n$ matrix given by

$$\mathrm{N}_{ij}(q,t) = \frac{\partial \Phi_r}{\partial q_j}(q,t) \tag{8}$$

for row $r = 1, \ldots, k$ and column $j = 1, \ldots, n$, and ψ is the k-vector defined by formula (6).

3.3 Lagrange multipliers

In general, equations (6–8) do not determine the constraint forces completely. Fortunately, for most kinds of mechanical constraints (including contacts and mechanical joints), we can easily determine the direction of the associated constraint force; only the magnitude remains to be determined. For example, for a particle sliding on a fixed plane, the constraint force will be some multiple of $u - \mu v$, where u is the plane's unit normal, v is the particle's velocity, and μ is the dynamic friction coefficient.

If we know the directions $d_1, \ldots, d_k$ of the constraint force vectors for all the equations $\Phi_1, \ldots, \Phi_k$, the total force vector C is some linear combination $C = \lambda_1 d_1 + \ldots + \lambda_k d_k$. We can determine the multipliers $\lambda_1, \ldots, \lambda_k$ by solving equation (7). This is the so-called *method of Lagrange multipliers*.

Let λ be a vector of Lagrange multipliers. By writing $C = -\mathrm{G}\lambda$, for the $n \times k$ matrix G whose columns are the directions $d_1, \ldots, d_k$, we can solve (7) for λ, obtaining

$$\mathrm{N}\mathrm{M}^{-1}\mathrm{G}\lambda = \mathrm{N}\mathrm{M}^{-1}F - \psi$$

This equation allows us to compute λ, and hence C, from known quantities.

4 Continuous model for elastic bodies

In a solid body, neighboring particles remain close to each other. Therefore, we can model a solid body as a piece of a continuous medium moving through $\mathbf{R}^3$, and subject to continuous deformations.

More precisely, we model a solid body as a closed and finite region U of $\mathbf{R}^3$, its *reference configuration*. (Note that U does not have to be connected, so everything we say here about one body applies equally well to a set of two or more separate bodies.) A *configuration* of the body is then a continuous function f that maps each point $u = [u_x, u_y, u_z]^\top \in U$ of the medium to a position $f(u) = [f_x(u), f_y(u), f_z(u)]^\top$ in $\mathbf{R}^3$. Since we don't want the bodies to interpenetrate, we require that f be one-to-one when restricted to the interior U^+ of U.

4.1 Potential energy of deformation

For an elastically deformable body, a significant part of the potential energy is stored in the elastic deformation of the material. At the microscopic level, this energy is due to the displacement of each particle relative to its neighbors.

Let u be a point of U^+. The relative displacement of particles in the neighborhood of u, for a given configuration f, is determined to first order by the

Jacobian matrix of f at u,

$$\mathbf{J}\, f(u) = \begin{bmatrix} \partial f_x/\partial u_x & \partial f_x/\partial u_y & \partial f_x/\partial u_z \\ \partial f_y/\partial u_x & \partial f_y/\partial u_y & \partial f_y/\partial u_z \\ \partial f_z/\partial u_x & \partial f_z/\partial u_y & \partial f_z/\partial u_z \end{bmatrix} \quad (9)$$

Specifically, a particle that is located at $v = u + \varepsilon$ in the reference configuration, for any infinitesimal vector ε, will be located at $f(v) = f(u) + (\mathbf{J}\, f(u))\varepsilon + O(|\varepsilon|^2)$ in the configuration f. For ordinary materials, it turns out that the second-order terms have negligible effect on the elastic energy. Therefore, the density of elastic energy $\phi_f(u)$ in the neighborhood of point u can be computed from the Jacobian $\mathbf{J}\, f(u)$ alone.

Let g be a configuration of the body for which the density $\phi_g(u)$ is zero. (We say that g is *locally relaxed* at u.) Then $\phi_f(u)$ depends on the local deformation determined by f, relative to that determined by g. That is, ϕ_f must be expressible as $\Phi(\mathrm{C})$, a function that depends on the material, where

$$\mathrm{C} = \mathbf{J}\,(f \circ g^{-1})(u) = (\mathbf{J}\, f(u))(\mathbf{J}\, g(u))^{-1} \quad (10)$$

is the *strain tensor* of configuration f at u.

Moreover, the elastic energy should not be affected by rotating the body as a rigid whole. In that case, it can be shown that $\Phi(\mathrm{C})$ must be a function of $\mathrm{D} = \mathrm{C}^\top \mathrm{C}$, a symmetric matrix called the *metric tensor* of f at u.

Also, if the material is isotropic (meaning that its mechanical properties are the same in all directions), the energy should remain unaffected by rotation of the locally relaxed configuration around point $g(u)$. In that case, it can be shown that $\Phi(\mathrm{C})$ must depend only on the coefficients d_1, d_2, d_3 of the characteristic polynomial of D,

$$\chi(\lambda) = \det(\mathrm{D} - \lambda \mathrm{I}) = 1 + d_1\lambda + d_2\lambda^2 + d_3\lambda^3 \quad (11)$$

It can be checked that

$$d_1 = \textstyle\sum_i \mathrm{D}_{ii} = \sum_{ij} \mathrm{C}_{ij}^2 \quad (12)$$

$$d_2 = \textstyle\sum_i \mathrm{D}^*_{ii} = \sum_{ij} (\mathrm{C}^*_{ij})^2 \quad (13)$$

$$d_3 = \det \mathrm{D} = (\det \mathrm{C})^2 \quad (14)$$

where M^* denotes the matrix of 2×2 cofactors of M.

In conclusion, the energy function $\Phi(\mathrm{C})$ is a function of the coefficients d_1, d_2, d_3 that is minimum (zero) when D is the identity matrix, i.e. when $d_1 = d_2 = 3$, and $d_3 = 1$. These conditions still allow infinitely many functions Φ. Partly for reasons of computational efficiency, we have chosen

$$\phi = \frac{\alpha}{2}\left(\sqrt{d_3} + \frac{1}{\sqrt{d_3}} - 2\right) + \frac{\beta}{2}\left(\frac{d_1^2}{3} - d_2\right) \quad (15)$$

For small deformations, the coefficients α and β that appear in formula (15) are precisely the two *elastic moduli* of the material, that express its resistance to changes in volume and in shape, respectively.

One advantage of formula (15), compared to the simpler quadratic forms that have been used in other works, is that it tends to infinity as the volume approaches zero. Therefore, the elastic forces computed by this formula will automatically prevent the tetrahedra from turning "inside out".

4.2 Viscosity losses

A real deformable body generally shows viscous friction, a loss of kinetic energy due to forces that tend to oppose the relative motion of neighboring particles inside the body.

The function that gives the current velocity of a particle in terms of its current position is $h = f' \circ f^{-1}$. The relative motion of particles in a small neighborhood is therefore summarized, to first order, by the spatial derivatives of this function; namely by the matrix $\mathrm{V} = \mathbf{J}\, h = \mathbf{J}\,(f' \circ f^{-1}) = (\mathbf{J}\, f')(\mathbf{J}\, f)^{-1})$.

By a reasoning similar to that used in the derivation of the elastic energy formula, we conclude that, in an homogeneous medium, the power loss ω per unit volume is some function of the coefficients of the symmetric part of V. We eventually settled for the following formula,

$$\omega = \frac{\eta_1}{2}\left(\sum_i \mathrm{V}_i^2\right) + \frac{\eta_2}{2}\left(\sum_{i<j} (\mathrm{V}_{i,j} + \mathrm{V}_{j,i})^2\right) \quad (16)$$

where η_1 and η_2 are viscosity coefficients related to the rates of change of volume and shape, respectively.

5 Finite element model

In order to obtain a finite description of the configuration of an elastic body, we use a simple finite element model. Namely, we approximate its shape by the union of tetrahedra (*elements*), with pairwise disjoint interiors, glued by their faces. Note that the kinetic energy, elastic energy, and dissipated power of the model are simply the sums of the corresponding terms for each element.

We restrict the deformations of the body so that, within each element, the configuration function is always an affine map of $\mathbf{R}^3$ to $\mathbf{R}^3$. We assume that the elastic moduli and viscosity coefficients of the material are equal for all points in the interior of each tetrahedron, and do not change with time. Finally, we assume that the total mass of each element is constant over time, and uniformly distributed within it. (Note that its density will vary as it gets deformed).

5.1 Barycentric coordinates

Let T be a tetrahedron in U, with vertices $u_1, \ldots, u_4$. If u is a point in T, we can write it as a convex linear combination of the vertices u_i,

$$u = \alpha_1\, u_1 + \alpha_2\, u_2 + \alpha_3\, u_3 + \alpha_4\, u_4$$

where $0 \le \alpha_i \le 1$, for $i = 1, \ldots, 4$, and $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1$. The α_i's are called the *barycentric coordinates of u in T*. If the vertices $u_1, \ldots, u_4$ are mapped to the points $p_1, \ldots, p_4$ by some configuration f, the current position $p = f(u)$ of point u will be

$$p = \alpha_1\, p_1 + \alpha_2\, p_2 + \alpha_3\, p_3 + \alpha_4\, p_4 \qquad (17)$$

Analogous interpolation formulas hold for the current velocity and acceleration of the point u, in terms of the current velocities and accelerations of the vertices of T.

5.2 Kinetic energy

Let T be a tetrahedron of mass μ and vertex velocities $v_1, \ldots, v_4$. The kinetic energy of T is given by the integral

$$K_T = \int_0^1 \int_0^{1-\alpha_3} \int_0^{1-\alpha_3-\alpha_2} \frac{v^2}{2}(6\mu) d\alpha_1\, d\alpha_2\, d\alpha_3$$

where $v = \alpha_1\, v_1 + \alpha_2\, v_2 + \alpha_3\, v_3 + (1 - \alpha_1 - \alpha_2 - \alpha_3) v_4$. This integral evaluates to

$$K_T = \frac{\mu}{20} \left(\sum_{i=1}^{4} v_i^2 + \sum_{i=1}^{4} \sum_{j=1}^{i-1} v_i \cdot v_j \right) \qquad (18)$$

5.3 Elastic energy

We will assume that for each tetrahedron T there is a configuration g_T of the body where T is *relaxed*, i.e. has zero elastic energy. If the current configuration f maps the vertices of T to points $p_i = [x_i, y_i, z_i]^\top$, the strain tensor $\mathrm{C} = \mathbf{J}\,(f \circ {g_T}^{-1})$ at any point u in the interior of T can be computed as $\mathrm{C} = \mathrm{B}\,\mathrm{A}^{-1}$, where

$$\mathrm{B} = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \\ x_4 - x_1 & y_4 - y_1 & z_4 - z_1 \end{bmatrix}$$

and A is computed in a similar way from the vertices of T in its relaxed configuration.

From the strain tensor, we compute the elastic energy density $\phi_f(u)$ by formula (15). Its integral over the tetrahedron T is merely the product of $\phi_f(u)$ by its volume V_T in its relaxed configuration.

5.4 Viscosity losses

The loss of energy due to viscous friction inside a tetrahedron is obtained by multiplying the power loss per unit volume ω by the current volume of the tetrahedron.

Since $\mathbf{J}\, f = \mathrm{C} = \mathrm{B}\mathrm{A}^{-1}$ and A is constant, $\mathbf{J}\, f'$ reduces to $\mathrm{B}'\,\mathrm{A}^{-1}$, and hence $\mathrm{V} = \mathrm{B}'\,\mathrm{B}^{-1}$. The viscous power loss density ω is then computed by formula (16).

6 Inequality conditions and discrete events

In general, integration of the differential equations cannot continue forever. Besides the equality-type constraints, there are other inequality-type *conditions* that must be satisfied. In general, we consider conditions that can be expressed by inequalities $\Gamma_i(q, q', \lambda, t) > 0$ for $i = 1, \ldots, m$, where the Γ_i are continuous functions, and λ_i are the Lagrange multipliers of the constraint forces.

Inequality conditions arise, for example, when we want to avoid interpenetration between bodies. This condition can be translated into a combination of algebraic inequalities applied to the vertex coordinates. Also, when a body is resting or sliding on another, the contact force must always push the bodies apart, rather than against each other. This condition can be written as $\lambda_i > 0$, where λ_i is the Lagrange multiplier associated with the contact force.

6.1 Discrete event detection

While integrating the equations of motion, we must stop the integration whenever we reach a state where one of the Γ_i is about to become negative. In order to continue the simulation past that moment, it will be necessary to change the system's state, the equations of motion, or the set of conditions that need to be satisfied. In any case, we'll then say that a *discrete event* has occured at that moment.

Suppose the integration took us from a valid extended state s_a at time t_a to a proposed extended state s_b at time t_b. We must check that all functions Γ_i remained of the same sign during that interval. Otherwise, we must detect the first instant after t_a, say t_e, at which one of the functions Γ_i becomes zero, and redo the integration from t_a to t_e.

We can suppose in practice that at most one function Γ becomes zero at any given instant. Simultaneous sign changes (which might correspond, for example, to bodies colliding in two or more points at the same time), have probability zero in general, and are meaningless anyway in the presence of numerical errors.

6.2 Detection by Hermite interpolation

So, let g be a function of time that describes the evolution of a condition function $\Gamma(q, q', \lambda, t)$ along the system's trajectory. In order to check whether the condition $g(t) > 0$ was satisfied throughout the integration step, we use Hermite interpolation of order k for g in that interval; that is, a polynomial of degree $2k+1$ whose values and derivatives to order k agree with those of g at t_a and t_b.

If the function Γ depends only on q, and possibly on t (which is true, for example, for the non-penetration conditions), we use a first-order Hermite interpolant, that is, the cubic polynomial whose values and first derivatives agree with those of g at t_a and t_b. These parameters are

$$\begin{aligned} g(t_a) &= \Gamma(q_a, t_a) \\ g'(t_a) &= \frac{\partial \Gamma}{\partial q}(q_a, t_a)^\top q'_a + \frac{\partial \Gamma}{\partial t}(q_a, t_a) \end{aligned} \tag{19}$$

and similarly for t_b.

In order to check whether the cubic polynomial is positive throughout the interval, we rewrite it in terms of the Bernstein-Bézier basis [5], that is

$$g(t) = P_1(1-u)^3 + P_2\, u(1-u)^2 + P_3\, u^2(1-u) + P_4\, u^3 \tag{20}$$

where $u = (t - t_a)/dt$, $dt = t_b - t_a$, and

$$\begin{aligned} P_1 &= g(t_a) & P_4 &= g(t_b) \\ P_2 &= g(t_a) + \frac{dt}{3} g'(t_a) & P_3 &= g(t_b) - \frac{dt}{3} g'(t_b) \end{aligned} \tag{21}$$

The advantage of this representation is that the value of $g(t)$ for any $t \in I = [t_a, t_b]$ is a convex combination of the coefficients $P_1, \ldots P_4$. Therefore, if these coefficients are all positive or all negative, the same can be said of $g(t)$ throughout the interval. If they have mixed signs, we bissect the interval with DeCasteljau's algorithm [5], and repeat the test in each half, recursively.

However, when the condition Γ depends on λ or on q', this cubic approximation cannot be used. To compute $g'(t_a)$ and $g'(t_b)$, we would need to compute λ' and q''. The former depends on the derivatives of the external forces, which may not be known; and the latter cannot be safely evaluated for the final state s_b, until we have verified that all preceding states were valid.

Therefore, in this case we just use a straight-line aproximation (Hermite interpolant of order $k = 0$) between the values of $g(t_a)$ and $g(t_b)$. Since $g(t_a)$ is assumed to be positive, the test for $g(t) > 0$ in I reduces to checking whether $g(t_b) > 0$. If this test fails, the approximate time when g becomes zero is $t_e = (t_a g(t_b) - t_b g(t_a))/(g(t_b) - g(t_a))$.

6.3 Collision detection

Realistic animation requires the detection and handling of collisions between the objects. In fact, we only need to watch for collision between a vertex and a non-adjacent face, or two non-adjacent edges; all other combinations are "coincidences" that occur with probability zero.

For typical models, with hundreds of facets, checking for collisions among all possible vertex-face and edge-edge pairs would be far too expensive. Fortunately, we can eliminate most of these tests by exploiting the spatial and temporal coherence of the scene.

Moreover, in typical situations, most of these pairs are widely separated in space. To take advantage of this fact, we compute an axis-aligned bounding box for each exposed vertex, edge, and face, over the current integration interval, and consider only pairs of elements whose bounding boxes intersect.

The bounding box of a surface vertex is computed by applying Hermite interpolation and Bernstein-Bézier range estimation to each coordinate, as in section 6.2. The bounding box for an edge is then obtained by enclosing the bounding boxes of its endpoints; and similarly for each face.

In order to quickly find the pairs of boxes that intersect, we use the incremental technique of Lin and Manocha [7]. We store the minimum and maximum coordinates of all boxes in three sorted lists, one for each axis. By scanning each list we can determine which pairs of boxes overlap when projected on the corresponding axis. Only the pairs whose bounding boxes overlap in all three axes are tested for collision; these pairs are kept in a set $\mathcal{S}$.

Thanks to temporal coherence, only a few pairs of elements need to be swapped in each list; and only the corresponding pairs of boxes may have to be added or deleted from the set $\mathcal{S}$. Therefore, the cost of keeping the lists sorted, and updating the set $\mathcal{S}$, is almost linear in the number of exposed faces.

7 Handling collisions

When two solid bodies collide, the material around the points of contact must deform, in order to prevent their interpenetration. The deformation gives rise to a contact force that tends to push the two bodies away from each other. The force disappears if and when the two bodies move apart.

7.1 The impulse model

For ideally rigid bodies, the force must be idealized as an *impulse* — an infinitely strong force, acting for an infinitely short time, with a finite integral. The

effect of the impulse is to instantaneously change the velocity of the bodies, without affecting their positions, so that the colliding surfaces are either sliding against or moving away from each other.

Computing these impulses is a very difficult problem, especially when the model allows multiple rolling and sliding contacts between the bodies, with static and dynamic friction. In general, the equality constraints allow infinitely many solutions for the impulses and constraint forces, and finding one that satisfies the inequality conditions is NP-hard [1, 2].

7.2 The spring model

The impulse model is not very appropriate for collisions between elastic bodies. The deformations in this case are macroscopic, and therefore the contact forces are finite and have nonzero duration. Therefore, we have chosen to use a simple spring-based model for the contact forces, which is a special case of the spring approach for equality constraints (section 3.1).

Specifically, when we detect a collision between two surface points a and b, we attach a virtual spring between them, and allow the integration to continue from the same state. The polyhedra representing the two bodies will then interpenetrate to some extent, but the spring will eventually stop and possibly reverse this motion. If the spring is strong enough, the interpenetration will be slight and hardly noticeable. We remove automatically the spring if and when it starts pulling the two bodies towards each other, instead of pushing them apart.

In the current implementation, we model only "sticky" (non-sliding) collisions. That is, a contact spring remains tied to the same surface points throughout its life. In a vertex-face collision, the spring is anchored to the vertex and to a specific point on the face, defined by the barycentric coordinates of the contact point at the moment the collision was detected. In an edge-edge collision, the spring is attached to a specific point along each edge, defined by the ratio of distances from the collision point to the edge endpoints.

If we allowed sliding contacts, we would have to worry about other kinds of events, for instance when a vertex that is sliding on a face hits a boundary edge. At that moment the vertex might fly away from the surface, or slide into the adjacent face, or (if the edge is concave) bounce back into the same face, or start sliding along the edge. Handling all these cases would require more time than we had available.

On the other hand, our collision model still allows objects to roll against each other: springs get added at the front edge of the contact region, and removed from the back edge.

The virtual springs have linear force and zero rest length, i.e. their stored energy is simply KL^2 where L is the current distance between the endpoints. The constant K must be chosen with some care: if too small, the bodies may push right through each other, and perhaps become tangled in a multitude of springs. If too strong, the integrator will have to use a small time step in order to follow the spring motion.

8 Implementation issues

8.1 Factoring the mass matrix

Observe that the kinetic energy formula (18) for a tetrahedron T depends only on the vertex velocities of T, and not on their positions. It follows that the mass matrix M is constant, depending only on the element masses and their adjacency relationships; and therefore needs to be computed only once, at the beginning of the simulation.

Moreover, it can be shown that M is symmetric and positive-definite. Therefore it can be factored as $\mathrm{M} = \mathrm{LDL}^\top$ where L is a lower triangular matrix with unit diagonal and D is a diagonal matrix with positive values [4]. This way, we can solve the system $\mathrm{M}q'' = F$ in $O(n^2)$ operations.

Finally, we take advantage of the fact that M is quite sparse: the number of non-zeros in M is exactly $3n_v + 6n_e$, where n_v is the number of vertices of the model, and n_e is the number of edges. For typical models, the factor matrix L is also sparse. Thus, by storing only the non-zero elements of L, the cost of evaluating the accelerations becomes practically linear in n.

8.2 Computing the internal forces

When computing the total force vector F, according to formula (4), we need to compute the partial derivatives of P and W with respect to the state coordinates. These derivatives are computed efficiently with the technique of Baur and Strassen [3], which is basically a sistematic application of the chain derivation rule.

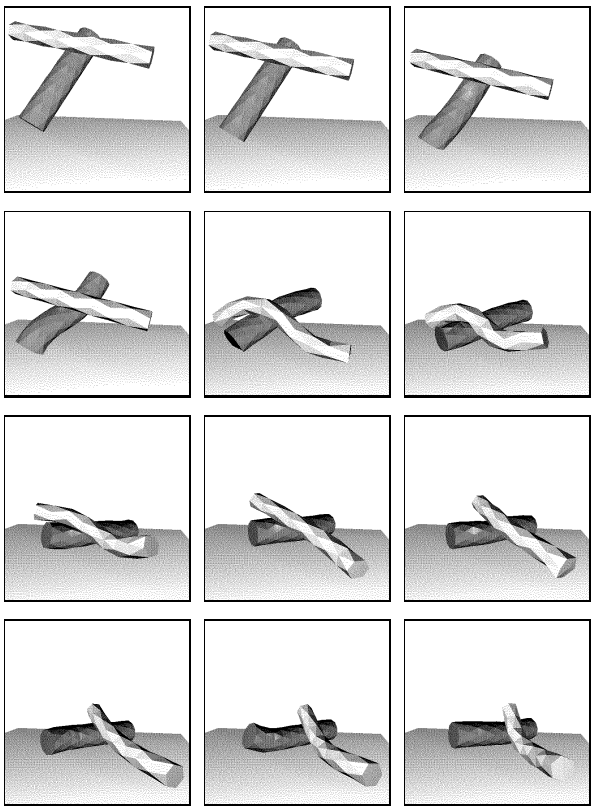
8.3 Numerical integration

To integrate the differential equation $q'' = \mathrm{M}^{-1}F$, we use the 4^{th} order adaptive method of Runge-Kutta-Fehlberg [4]. This routine automatically generates an estimate of the truncation error, and adjusts the step size so as to keep that error below a user specified numerical tolerance.

9 Results

Below we see an animation produced by our system, of two soft rubber cylinders falling on a hard floor. The 12 frames shown cover about 1 second of simulated time. The models have a total of 462 tetrahedra and 191 vertices, with 334 exposed triangles.

Construction and factorization of the mass matrix took 6 minutes, and the simulation itself took 1 hour and 5 minutes (3900 times "real time") on a SPARC 1000. The integrator took 6650 steps, averaging 150 microseconds of simulated time and 0.58 seconds of CPU time per step. Roughly 1/3 of this time was spent computing the vertex accelerations, and 2/3 testing for possible collisions among 120,000 edge-edge and 57,000 vertex-face pairs; 140 collisions were detected during the run.



10 Conclusions

We feel that the performance of our simulator is acceptable, given the complexity of the task. However, there is still plenty of room for improvement. The main problem we face at the moment is the "stiffness" of the differential equation, due to the contact springs, which forces the integrator to use a very small step size (around 100 microseconds when the error tolerance is set to 1mm). To solve this problem, we may have to replace the contact springs by some combination of impulses and equality constraints.

References

[1] D. Baraff. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10:292–352, 1993.

[2] D. Baraff. Fast contact force computation for non-penetrating rigid bodies. *Proc. SIGGRAPH'94*, 28(3):23–34, July 1994.

[3] W. Baur and V. Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983.

[4] R. Burden and J. Faires. *Numerical Analysis.* PWS-Kent, fourth edition, 1989.

[5] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer graphics/Principles and practice.* Addison-Wesley, second edition, 1990.

[6] H. Goldstein. *Classical Mechanics.* Addison-Wesley, second edition, 1980.

[7] M. C. Lin and D. Manocha. Efficient contact determination between geometric models. Technical report, University of North Carolina, Chapel Hill, NC, 1994.

[8] B. Mirtich and J. Canny. Impulse-based dynamic simulation. Technical report, Department of Computer Science,University of California, Berkeley, CA, 1994.

[9] M. Moore and J. Wilhelms. Collision detection and response for computer animation. *Proc. SIGGRAPH'88*, 22(4):289–298, August 1988.

[10] A. Pentland and J. Williams. Good vibrations: modal dynamics for graphics and animation. *Proc. SIGGRAPH'89*, 23(3):215–222, July 1989.

[11] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. *Proc. SIGGRAPH'88*, 22(4):269–278, August 1988.

[12] D. Terzopoulos, J. Platt, A. H. Barr, and K. Fleischer. Elastically deformable models. *Proc. SIGGRAPH'87*, 21(4):205–214, July 1987.

[13] A. Witkin, M. Gleicher, and W. Welch. Interactive dynamics. *Computer Graphics*, 24(2):11–22, March 1990.

[14] A. Witkin and W. Welch. Fast animation and control of nonrigid structures. *Proc. SIGGRAPH'90*, 24(4):243–250, August 1990.