Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies

David Baraff

Program of Computer Graphics
Cornell University
Ithaca, NY 14853

Abstract

A method for analytically calculating the forces between systems of rigid bodies in resting (non-colliding) contact is presented. The systems of bodies may either be in motion or static equilibrium and adjacent bodies may touch at multiple points. The analytic formulation of the forces between bodies in non-colliding contact can be modified to deal with colliding bodies. Accordingly, an improved method for analytically calculating the forces between systems of rigid bodies in colliding contact is also presented. Both methods can be applied to systems with arbitrary holonomic geometric constraints, such as linked figures. The analytical formulations used treat both holonomic and non-holonomic constraints in a consistent manner.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism

Additional Key Words and Phrases: dynamics, constraints, simulation

1. Introduction

Recent work has focused on using the laws of Newtonian dynamics to simulate the motions of systems of rigid bodies. A realistic simulation of rigid bodies demands that no two bodies inter-penetrate. In order to enforce this constraint a simulator must first detect potential inter-penetration between two bodies and then act to prevent the two bodies from penetrating. However, in keeping with the laws of Newtonian dynamics, a realistic simulation should not prevent inter-penetration in an arbitrary manner. The simulator should calculate what forces would actually arise in nature to prevent bodies from inter-penetrating and then use these forces to derive the actual motion of the bodies. In order to calculate these forces an explicit formulation is necessary.

Traditional techniques from engineering and physics are not applicable to the problem of calculating forces between bodies in resting contact. These techniques assume that the systems of bodies being analyzed are in equilibrium. However, many of the simulations in computer graphics involve systems of bodies that are not in equilibrium. For example, the forces between the bricks in figure 1 cannot be calculated using traditional techniques.

©1989 ACM-0-89791-312-4/89/007/0223 $00.75

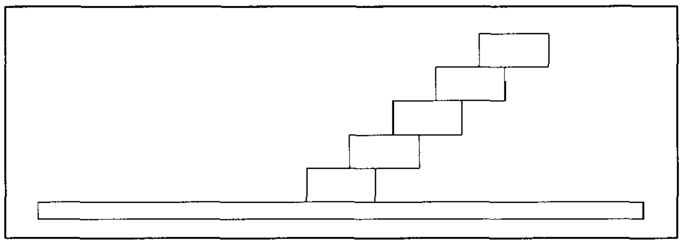


Figure 1. Overbalanced stack of bricks.

This paper focuses on the following specific problem: given a number of *non-colliding* rigid polyhedral bodies, calculate the forces that would naturally arise to prevent bodies from inter-penetrating. The bodies may also be constrained to satisfy certain geometrical relationships such as those present in articulated figures[2, 8]. An analytical solution to the problem is presented that uses linear programming techniques to formulate and heuristically solve a system of inequality and equality constraints on the forces. The system of constraints guarantees that the contact forces will prevent inter-penetration and satisfy the laws of Newtonian dynamics. The solution is generalized to yield an improved algorithm for calculating forces between colliding systems of rigid polyhedral bodies.

2. Previous Work

Analytical methods for calculating the forces between *colliding* rigid bodies have been presented by Moore and Wilhelms[13] and Hahn[7]. Both methods calculated the impulse between a single pair of bodies that collided at a single point. Moore and Wilhelms modeled simultaneous collisions as a slightly staggered series of single collisions and used non-analytical methods (below) to deal with bodies in resting contact. Hahn prevented bodies in resting contact from inter-penetrating by modeling their contact as a series of frequently occurring collisions. This model may be suitable for preventing inter-penetration in animation applications; however, it is not a valid analytical model of forces between bodies in resting contact.

"Penalty" methods that introduce restoring forces when objects inter-penetrate have also been presented. Terzopolous *et al.*[18] and Platt and Barr[16] produced highly realistic animations of rigid and deformable non-penetrating bodies. Inter-penetration was prevented in both papers by introducing arbitrary penalty forces that acted to separate penetrating bodies; a natural solution method, since dynamical correctness of these forces was not a focus of either paper. Moore and Wilhelms[13] introduced spring forces (a penalty method) to prevent bodies in resting contact from penetrating.

2.1 Penalty Methods vs. Analytical Methods

Analytical methods offer several advantages over penalty methods. Penalty methods for rigid bodies are often computationally expensive, give only approximate results, and may require adjustments for different simulation conditions (figure 2).

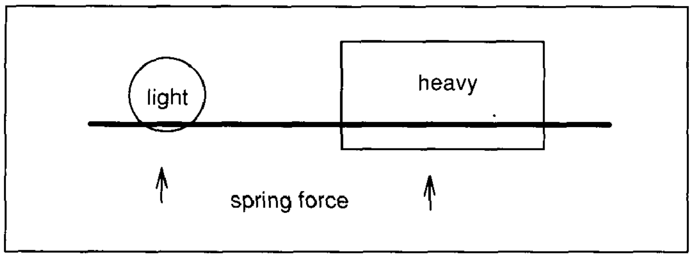


Figure 2. Differing amounts of inter-penetration from a penalty method.

These undesirable behaviors arise from the attempt to model infinite quantities (infinite rigidity of bodies, infinitely hard surfaces) with finite values. In particular, the differential equations that arise using penalty methods may be "stiff" and require an excessive number of time-steps during simulation to obtain accurate results. Additionally, the correctness of the simulation under penalty methods is very difficult to verify. In their defense, penalty methods for rigid bodies are simple to implement and are easily extendible to non-rigid bodies.

In contrast, analytical methods for rigid bodies give exact answers and produce differential equations that require far fewer time steps during simulation. The correctness of the simulation when using an analytical method is easily provable because analytical solutions are based directly on the laws of Newtonian dynamics. Analytical methods however are much more complex to derive and implement.

3. Simulation using Analytical Methods

Simulations of rigid bodies employing analytical methods should treat collision forces and resting contact forces differently. Analytically, collision forces are discontinuous impulsive forces in that they exist for a single instant of time and have the dimensions of mass times velocity (or equivalently force times time). Resting contact forces, or more simply, contact forces, are continuous over some non-zero interval of time and have the dimensions of mass times acceleration. The effects of collision forces are independent of non-impulsive forces such as contact forces or gravity. Impulsive forces cause discontinuities in a body's velocity; contact forces do not.

Our simulator iterates through time (time steps) by solving a first order system of coupled ordinary differential equations[1, 2]. Given the net force and torque on each body, the differential equations can be solved to yield the motion of the bodies. We adopt the usual method for solving the system of differential equations by using numerical integration procedures such as fourth order Runge-Kutta or Adams-Moulton[17] with adaptive time-step parameters. The integrator is given initial conditions in the form of the starting orientations, positions, and linear and angular velocities of all the bodies. As stated above, analytical methods introduce discontinuities in some of the velocities when collisions occur. It is unwise to blithely integrate over these collision times.[1]

[1]Numerical integrators assume that the functions they are integrating are continuous functions of time. If a function being integrated is discontinuous at some time t_0, the integrator must integrate up to t_0, stop, and then restart at t_0 with new initial conditions.

Finding the time at which a collision occurs can be viewed as a root-finding problem. The collision time is found by using backtracking methods similar to those described by Moore and Wilhelms[13]. The collision time is bracketed by successively shorter time intervals until the colliding objects touch within a suitable tolerance.[2] Once the collision time has been calculated, the integrator is stopped (at the collision time) and collision forces are computed. Collision forces may be calculated using previous methods[7, 13] or by the improved collision method presented in section 8. The collision forces are used to compute the new velocities of the colliding bodies and then the integrator is restarted with new initial conditions. The new initial conditions are the positions and orientations at the time of the collision and the newly computed velocities. We call this series of steps *resolving* the collision. Since the computation of the new velocities (due to the collision forces) is independent of any contact forces, contact forces are not calculated until *after* collisions are resolved. Accordingly, the formulation of the resting contact problem implicitly assumes that no bodies in contact are colliding i.e. collisions have been resolved.

4. Modeling Contact

We will use the following terminology. Let two bodies A and B be in contact (colliding or resting) at some time t_0; A and B touch each other at some number of *contact points*. At time t_0, let p be the position of an arbitrary contact point in some world space. Define p_a and p_b as the positions of the two points of A and B that satisfy $p_a(t_0) = p = p_b(t_0)$ (figure 3).

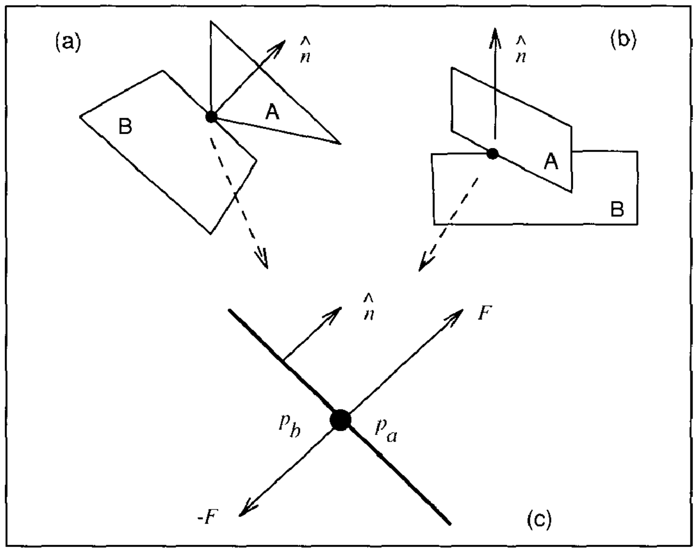


Figure 3. (a) Vertex-plane contact (side view). (b) Edge-edge contact. (c) Contact geometry.

p_a and p_b are functions of time; they track the motion of two specific points of A and B that coincide at time t_0. Both p_a and p_b vary according to the independent rigid body motions of A and B. The relative motion, $\dot{p}_a(t_0) - \dot{p}_b(t_0)$, indicates whether A and B are colliding, resting, or separating at point p at time t_0.

At each contact point there is a *contact force* $\vec{F}$, (possibly zero) between A and B and a unit surface normal $\hat{n}$. For vertex-plane contact, the body whose vertex is the contact point is identified as body A. $\hat{n}$ is defined as the outwards unit surface

[2]We have found that using the relative displacements of inter-penetrating bodies to implement a *regula falsa* method results in much faster convergence than the simpler bisection method. Our tolerance was chosen empirically.

normal of B at p_b (figure 3a). For edge-edge contact, one body is identified as body A arbitrarily. $\hat{n}$ is defined as a unit vector mutually perpendicular to the two contacting edges and directed away from B (figure 3b). In the absence of friction, $\vec{F}$ is colinear with $\hat{n}$ for both vertex-plane and vertex-edge contacts. Thus, we may write $\vec{F} = f\hat{n}$ where f is the unknown magnitude of the contact force. From Newton's third law, if the force on A is $f\hat{n}$ then the force on B will be $-f\hat{n}$. Our goal is to calculate contact force magnitudes that prevent inter-penetration.

4.1 Degenerate Contact Points

For vertex-vertex contacts, one body is identified as body A arbitrarily; for vertex-edge contacts, the body whose vertex is a contact point is identified as body A. Physically, vertex-vertex contacts (figure 4a) and vertex-edge contacts are *indeterminate*: the surface normal $\hat{n}$ is not well defined and the direction of the contact force between A and B cannot be determined. Consequently, the physically correct behavior of A and B may be indeterminate during the interval in which degenerate contact points occur. In the absence of degenerate contact points, the physically correct behavior of A and B is unique.

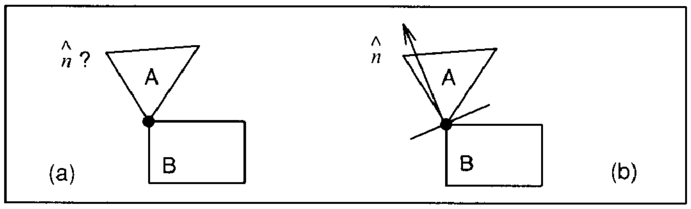


Figure 4. (a) Indeterminate vertex-vertex contact or vertex-edge contact (side view). (b) Removing indeterminacy by choosing $\hat{n}$.

Since the contact force direction is indeterminate, a direction for $\vec{F}$ must be chosen. To be consistent with our description of vertex-plane and edge-edge contacts we set $\hat{n}$ to be an arbitrary unit vector directed away from B.[3] We then write $\vec{F} = f\hat{n}$, with f an unknown magnitude. The choice of $\hat{n}$ at degenerate contact points implicitly determines a particular behavior for A and B. Usually, indeterminate configurations exist only instantaneously, so the choice of $\hat{n}$ for indeterminate configurations has little effect on the simulation.

However, if a configuration of bodies has degenerate contact points, calculating the correct contact force magnitudes is an NP-complete problem. This result follows directly from a theorem due to Palmer[15]. The problem remains NP-complete even after a direction for $\vec{F}$ is chosen. Solving NP-complete problems currently requires exponential time and is considered intractable. (See Palmer[15] and Garey and Johnson[3] for further discussion). The NP-completeness may be avoided by converting indeterminate contacts to determinate contacts. Our system does this by imagining that B has been locally extended in a plane normal to $\hat{n}$ as in figure 4b. This extension of B converts each indeterminate contact to the vertex-plane contact of figure 3a. Henceforth, indeterminate contact points are assumed to have been resolved in this manner.

4.2 Restricting Contact Points

Figure 5 shows regions of contact points between contacting bodies. In order to prevent inter-penetration, it is sufficient to consider relative motion at only the endpoints and vertices of the line segment and polygon contact regions shown. For polygonal contact regions, the motion produced by any distribution of contact forces over the entire contact region can be produced by equivalent forces acting on only the vertices of the contact region; the same is true for line segment contact regions[15]. Interior points of contact regions are not considered as contact points.

[3]Our system chooses $\hat{n}$ by averaging nearby surface normals.

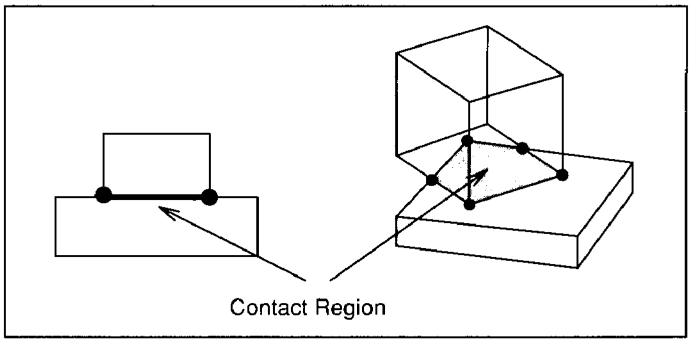


Figure 5. The contact force is assumed to be zero except at points marked with ●.

Thus, a configuration of bodies can be considered to have only finitely many contact points. Let n be the number of contact points; for $1 \le i \le n$, the known surface normal and unknown force magnitude at the ith contact point are written $\hat{n}_i$ and f_i. The unknown f_i's are grouped into a single vector of scalar unknowns, $\vec{f}$. For simplicity, we shall refer to f_i (the ith element of $\vec{f}$) as the contact force at point i, even though it is only the magnitude of the contact force. The actual force $\vec{F}_i$ at contact point i is given by $\vec{F}_i = f_i\hat{n}_i$.

5. Calculating Dynamically Correct Contact Forces

We can now place exact conditions on the contact forces we wish to calculate. A vector $\vec{f}$ of contact force magnitudes is *correct* if it satisfies the following conditions:

(1) The contact forces do not allow the bodies to inter-penetrate.

(2) The contact forces can "push" but not "pull".

(3) The contact forces occur only at contact points; once two bodies have separated at a contact point, there is no force between them at that contact point.

(4) Viewed as a function of time, contact forces are continuous.

Condition (4) is phrased somewhat informally, but the intuitive idea is that the force at a given contact point should vary smoothly over time (in the time interval between successive collisions). A correct vector of contact forces will produce motion that is dynamically correct. Note that more than one correct $\vec{f}$ may exist for a given configuration. Normally, the unique solution of forces for an "overdetermined" structure is found using the equations of compatibility; the assumption of absolute rigidity precludes the use of these equations[4]. However, all correct vectors $\vec{f}$ result in the same (dynamically correct) motion.

5.1 Non-penetration Constraints

To prevent inter-penetration it suffices to examine the relative motions of bodies at each contact point. At time t_0, let the ith contact point be at position p between bodies A and B and let the functions p_a and p_b be defined as in section 4. We would like to characterize the geometrical relationship between A and B in the neighborhood of p at some (future) time $t \ge t_0$. Define a *characteristic* function (of time) $\chi_i(t)$ that indicates at time t whether A

and B are: separate near p, touching near p, or inter-penetrating near p.[4] For vertex-plane contact a characteristic function may be written as

$$\chi_i(t) = \hat{n}(t)\cdot(p_a(t) - p_b(t)). \qquad (1)$$

This function (locally) characterizes vertex-plane contacts: $\chi_i(t)$ is positive, zero, or negative according to whether $p_a(t)$ is outside, on, or inside B (figure 6). The same function may be used for edge-edge contacts: $\chi_i(t)$ is positive, zero, or negative according to whether the edge of A is outside, on, or inside B. In both cases, $\chi_i(t) < 0$ signals inter-penetration, and must be prevented.

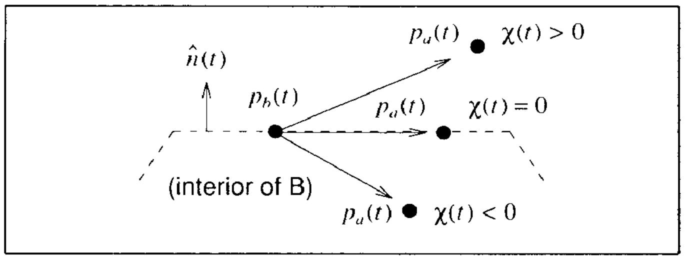


Figure 6. Characterization of three different positions of $p_a(t)$ relative to B at time t.

At time t_0, A and B touch $(p_a(t_0) = p_b(t_0))$; therefore

$$\chi_i(t_0) = \hat{n}(t_0)\cdot(p_a(t_0) - p_b(t_0)) = 0. \qquad (2)$$

Since $\chi_i(t) < 0$ signals inter-penetration, $\vec{f}$ must guarantee that χ_i is a non-decreasing function at time t_0; equivalently, $\vec{f}$ must not allow the relative displacement in the $\hat{n}$ direction to decrease at time t_0.

Appendix A gives a derivation for $\dot{\chi}$ and $\ddot{\chi}$. $\dot{\chi}$ measures relative velocity in the $\hat{n}$ direction, while $\ddot{\chi}$ is a measure of relative acceleration. As such, the contact forces $\vec{f}$ at time t_0 determine $\ddot{\chi}(t_0)$; but $\dot{\chi}(t_0)$ is independent of the contact forces that exist at time t_0. From the simulator's viewpoint, χ (displacement) and $\dot{\chi}$ (relative velocity) are given at time t_0, while $\ddot{\chi}$ depends on the contact forces at time t_0.

What happens then if $\dot{\chi}_i(t_0) < 0$? This indicates that A and B are colliding (since $\chi_i(t_0)$ is decreasing). Since collisions are resolved before calculating contact forces, this will not occur. Conversely, if $\dot{\chi}_i(t_0) > 0$ then A and B are separating at contact point i, *regardless of the contact forces at time* t_0. Immediately after t_0, this contact point will not exist and thus there will be no contact force here by condition (3). Contact forces vary continuously by condition (4) so the contact force, f_i, must be zero at time t_0.[5] Thus, contact points with $\dot{\chi}_i(t_0) > 0$ may be ignored since the force at these points is zero and χ_i is increasing. We will assume that these contact points are discarded by some preprocessing step and ignore their existence hereafter.

The remaining case is:[6]

$$\dot{\chi}_i(t_0) = 0. \qquad (3)$$

If $\vec{f}$ makes $\ddot{\chi}_i(t_0) < 0$, then p_a is accelerating into B and inter-penetration will immediately occur. Formally, if $\chi_i(t_0) = \dot{\chi}_i(t_0) = 0$ and $\ddot{\chi}_i(t_0) < 0$ then χ_i is decreasing at time t_0, resulting in immediate inter-penetration. Thus, condition (1) may be written as

$$\ddot{\chi}_i(t_0) \geq 0, \qquad (4)$$

corresponding to our intuition that $\vec{f}$ must not allow p_a to accelerate into B. Appendix A shows that the relative acceleration $\ddot{\chi}_i$ at time t_0 is a linear function of the contact force $\vec{f}$. To make this dependence clear, we will explicitly write $\ddot{\chi}_i$ as a function of $\vec{f}$, $\ddot{\chi}_i(\vec{f})$ with the implicit understanding that this occurs at time t_0. Condition (1) in the form

$$\ddot{\chi}_i(\vec{f}) \geq 0 \qquad (5)$$

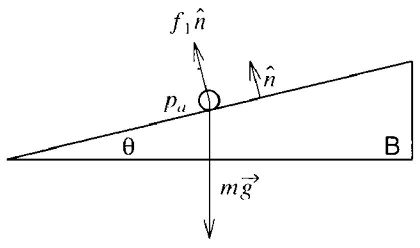
is now a constraint on $\vec{f}$. Figure 7 derives the constraints for a point mass resting on an inclined plane. (Refer to appendix A for a derivation of $\ddot{\chi}$ when body B is at rest).

Variables

p_a	contact point	f_1	force magnitude
m	mass of A	$\vec{F}$	total force
$\vec{g}$	gravity vector	$\hat{n}$	unit normal

Relations

$$\vec{F} = m\vec{g} + f_1\hat{n} \qquad \ddot{p}_a = \frac{\vec{F}}{m} \qquad \hat{n}\cdot\vec{g} = -|\vec{g}|\cos\theta$$



Constraints

$$\ddot{\chi}_1 = \hat{n}\cdot\ddot{p}_a = \hat{n}\cdot\left[\frac{m\vec{g}+f_1\hat{n}}{m}\right] = \frac{m(\hat{n}\cdot\vec{g})+f_1}{m} \geq 0 \quad \text{or}$$

$$f_1 \geq -m(\hat{n}\cdot\vec{g}) = m|\vec{g}|\cos\theta$$

Figure 7. Constraint equations for a point mass A (with position p_a) resting on a fixed inclined plane B.

Figure 8 derives constraints for the contact forces between a block (body A) and a (fixed) floor (body B). Barzel and Barr[2] derive $\ddot{p}_a$; note that centripetal acceleration terms are absent from figure 8 since the block is at rest. Derivations for the other relations in figure 8 are found in Goldstein[5].

5.2 Matrix Formulations of Conditions (1) and (2)

From the preceding section, $\ddot{\chi}(\vec{f})$ is a linear function. Thus condition (1) may be written as a linear inequality

$$\ddot{\chi}_i(\vec{f}) = a_{i1}f_1 + a_{i2}f_2 + \cdots + a_{in}f_n - b_i \geq 0 \qquad (6)$$

for $1 \leq i \leq n$. Condition (2), that contact forces only push, may be written as the inequality

$$f_i \geq 0 \qquad (7)$$

for $1 \leq i \leq n$. Conditions (1) and (2) can be written more concisely using matrix notation: let A be the matrix of a_{ij}'s and $\vec{b}$ the

[4]The use of a characteristic function serves several purposes. First, it makes possible simple correctness proofs for our methods. Second, it is extensible to contact between higher-order surfaces. Third, it allows a unified treatment of the different contact geometries.
[5]From calculus, if a continuous function $g(t)$ satisfies $g(t) = 0$ for $t > t_0$, then $g(t_0) = 0$.
[6]In practice, $\dot{\chi}_i(t_0)$ is compared to zero within an empirically determined tolerance value ϵ. This applies to all similar numerical comparisons in this paper.

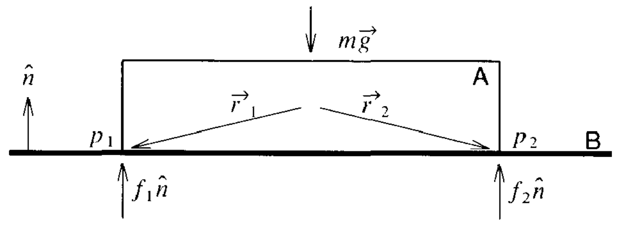
Variables

$p_{1,2}$	contact points	$\vec{r}_{1,2}$	body coordinates
$\vec{a}$	linear acceleration	$\vec{\alpha}$	angular acceleration
$f_{1,2}$	force magnitudes	$\vec{g}$	gravity vector
$\vec{F}$	total force	$\vec{\tau}$	total torque
m	mass	I	moment of inertia

Relations

$$\vec{F} = m\vec{g} + f_1\hat{n} + f_2\hat{n} \qquad \vec{\tau} = \vec{r}_1 \times f_1\hat{n} + \vec{r}_2 \times f_2\hat{n}$$

$$a = \frac{\vec{F}}{m} \qquad \vec{\alpha} = \frac{\vec{\tau}}{I} \qquad \ddot{p}_{1,2} = \vec{a} + \vec{\alpha} \times \vec{r}_{1,2} = \frac{\vec{F}}{m} + \frac{\vec{\tau}}{I} \times \vec{r}_{1,2}$$



Constraints

$$\ddot{\chi}_1 = \hat{n} \cdot \ddot{p}_1 = \hat{n} \cdot \left[\frac{m\vec{g} + f_1\hat{n} + f_2\hat{n}}{m} + \frac{\vec{r}_1 \times f_1\hat{n} + \vec{r}_2 \times f_2\hat{n}}{I} \times \vec{r}_1 \right] \geq 0$$

$$\ddot{\chi}_2 = \hat{n} \cdot \ddot{p}_2 = \hat{n} \cdot \left[\frac{m\vec{g} + f_1\hat{n} + f_2\hat{n}}{m} + \frac{\vec{r}_1 \times f_1\hat{n} + \vec{r}_2 \times f_2\hat{n}}{I} \times \vec{r}_2 \right] \geq 0$$

Figure 8. Constraint equations on the (unknown) contact force magnitudes $f_{1,2}$, for a block (A) supported by a fixed floor (B). The block is at rest.

vector of b_i's in equation (7). Then

$$A\vec{f} - \vec{b} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ f_n \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ b_n \end{bmatrix} = \begin{bmatrix} \ddot{\chi}_1(\vec{f}) \\ \ddot{\chi}_2(\vec{f}) \\ \cdot \\ \cdot \\ \ddot{\chi}_n(\vec{f}) \end{bmatrix}. \tag{8}$$

Comparing the left- and right-hand sides componentwise,

$$\ddot{\chi}_i(\vec{f}) = (A\vec{f})_i - b_i, \tag{9}$$

so condition (1) can be expressed concisely as

$$A\vec{f} - \vec{b} \geq \vec{0} \tag{10}$$

or equivalently as

$$A\vec{f} \geq \vec{b}. \tag{11}$$

Condition (2) is stated as $\vec{f} \geq \vec{0}$.[7]

5.3 Linear Programming

Finding a vector $\vec{x}$ that satisfies $M\vec{x} \geq \vec{c}$ (where M is a matrix and $\vec{c}$ is a vector) and minimizes a linear function $z(\vec{x})$ is an example of the *linear programming* problem[10, 14]. In linear programming, the constraints between $M\vec{x}$ and $\vec{c}$ may mix "=" constraints with "$\geq$" constraints. A general lower bound of the form $\vec{x} \geq \vec{0}$ is optional. Systems for which an $\vec{x}$ exists that satisfy all the constraints are *feasible systems* and each such $\vec{x}$ is a *feasible solution*. Otherwise, the system is *infeasible*. Systems for which a feasible $\vec{x}$ exists that minimizes z are *bounded systems* and each such $\vec{x}$ is an *optimal solution*. Finding feasible (but not necessarily optimal) solutions is also a linear programming problem.

An $\vec{f}$ that satisfies conditions (1) and (2) can be expressed as a linear programming problem: *choose $\vec{f}$ subject to the constraints*

$$A\vec{f} \geq \vec{b} \text{ and } \vec{f} \geq \vec{0}. \tag{12}$$

Linear programming is a polynomial time problem. A is typically sparse; thus linear programs involving A have expected $O(n)$ solution times if a sparsity exploiting linear programming package is used[12]. Appendix B discusses some numerical issues concerning the matrix A.

5.3 Formulating Conditions (3) and (4)

However, a feasible $\vec{f}$ with respect to equation (12) is not necessarily a correct $\vec{f}$. Intuitively, an $\vec{f}$ that satisfies conditions (1) and (2) may be an incorrect solution because it is "too strong". In figure 7, the only correct solution is $\vec{f} = (m|\vec{g}|\cos\theta)$. However, $\vec{f} = (2m|\vec{g}|\cos\theta)$ is a feasible solution (with respect to equation (12)) that prevents inter-penetration by incorrectly accelerating A away from B. Condition (3) will prevent this.

Recall that $\ddot{\chi}$ is a measure of relative acceleration. For the ith contact point, if

$$\ddot{\chi}_i(\vec{f}) > 0 \tag{13}$$

then $\dot{\chi}_i$ is strictly increasing (since $\dot{\chi}_i = 0$ by assumption) and A and B are separating at this contact point. Call such a contact point a *vanishing* contact point and call all other contact points (where $\ddot{\chi}_i(\vec{f}) = 0$) *non-vanishing* contact points (figure 9).

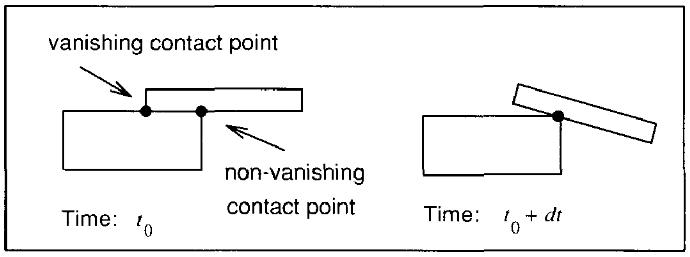


Figure 9. A vanishing contact point at time t_0. The bodies separate at the point immediately after time t_0.

From section 5.1, the contact force at a vanishing contact point is zero, by conditions (3) and (4). Condition (3) may be formulated as the statement

$$f_i\ddot{\chi}_i(\vec{f}) = 0 \tag{14}$$

because either contact point i is vanishing ($f_i = 0$ and $\ddot{\chi}_i(\vec{f}) > 0$) or non-vanishing ($\ddot{\chi}_i(\vec{f}) = 0$ and $f_i \geq 0$). Thus, the last constraint needed to guarantee correctness is equation (14), for $1 \leq i \leq n$. We can write all three constraints in the form

$$A\vec{f} \geq \vec{b}, \ \vec{f} \geq \vec{0} \text{ and } \sum_{i=1}^{n} f_i\ddot{\chi}_i(\vec{f}) = 0 \tag{15}$$

because $f_i \geq 0$ and $\ddot{\chi}_i(\vec{f}) \geq 0$ forces each term $f_i\ddot{\chi}_i(\vec{f})$ in the summation term of equation (15) to be non-negative, preventing cancellation. Since $\sum f_i\ddot{\chi}_i(\vec{f})$ is non-negative, any correct solution $\vec{f}$ minimizes this sum. Equation (15) can also be written as

$$A\vec{f} \geq \vec{b}, \ \vec{f} \geq \vec{0} \text{ and } \vec{f}^T A\vec{f} - \vec{f}^T\vec{b} = 0. \tag{16}$$

[7]For n-vectors $\vec{x}$ and $\vec{y}$, $\vec{x} \geq \vec{y}$ means $x_i \geq y_i$ for $1 \leq i \leq n$.

However, the term $\vec{f}^T A \vec{f}$ is *quadratic* in f_i; finding a correct $\vec{f}$ (a feasible solution to equation (16)) is an example of the *quadratic programming* problem. Quadratic programming, unlike linear programming, is an NP-hard problem in general[3]. For the (frictionless) contact model, A turns out to be positive semidefinite (PSD). Quadratic programs, when restricted to PSD matrices, can be theoretically solved in polynomial time[9], but no practical polynomial time algorithms are currently known[14]. Also, there is no reason to believe that A will remain PSD when friction is added to the model. For these reasons, we have not attempted to solve the problem of finding a correct $\vec{f}$ by direct methods. We have developed a successful heuristic algorithm for this problem. The algorithm, like any heuristic algorithm, can be made to fail on certain pathological examples.

6. Heuristic Solution Methods

A determinate configuration of bodies has only one physically correct motion; any correct $\vec{f}$ produces this motion. Thus, the correct set V of vanishing contact points for any configuration is unique. Once we know V, a correct $\vec{f}$ can be found using linear programming.

6.1 Calculating $\vec{f}$ given V

Suppose that we are given the (disjoint) index sets V and C:

$$V = \{j \mid \text{contact point } j \text{ is vanishing}\}.$$

$$C = \{k \mid \text{contact point } j \text{ is not vanishing}\}. \qquad (17)$$

For any correct solution $\vec{f}$, if $j \in V$ then $f_j = 0$, and if $k \in C$ then $\ddot{\chi}_k(\vec{f}) = 0$. Thus, given V, finding a correct $\vec{f}$ may be phrased as: *choose $\vec{f}$ subject to the constraints*

$$\forall j \in V \begin{Bmatrix} f_j = 0 \\ \text{and} \\ \ddot{\chi}_j(\vec{f}) \geq 0 \end{Bmatrix} \text{ and } \forall k \in C \begin{Bmatrix} f_k \geq 0 \\ \text{and} \\ \ddot{\chi}_k(\vec{f}) = 0 \end{Bmatrix}. \qquad (18)$$

The constraints of this new system are all linear and conditions (1) and (2) are enforced. Condition (3) is also enforced since $f_i \ddot{\chi}_i(\vec{f}) = 0$ for $i \in V$ or $i \in C$.

The new system is formed from the original constraint $A\vec{f} \geq \vec{b}$. For each non-vanishing contact point, the "≥" constraint is changed to a "=" constraint since $\ddot{\chi}_k(\vec{f}) = 0$ is equivalent to $(A\vec{f})_k = b_k$. For each vanishing contact point, f_j is set to zero and the "≥" constraint is retained. Additionally, the jth column of A may be set to zero (since $f_j = 0$) to exploit increased sparsity in A. Figure 10 shows a quadratic constraint system for four contact points, and the linear system formed when $V = \{1,3\}$ and $C = \{2,4\}$.

However, we have no "oracle" that will provide us with the set V and we are currently unable to (efficiently) determine which contact points are vanishing. Finding V is easy for some configurations (figure 9), but not so for others. (As an example, try to determine the vanishing contact points of figure 1. Later frames from the simulation are given in figure 14). We can however take a guess as to which contact points are vanishing and then use the new linear system to test the guess. If the guess is correct, the new linear system will be feasible. Any feasible solution $\vec{f}$ found by a linear programming routine will be a correct solution. If the guess is incorrect, no $\vec{f}$ will satisfy the new system. The linear programming routine will report that the new system is infeasible, indicating the incorrectness of the guess. The obvious question is: how do we guess which contact points are vanishing and which are not?

Original System

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \geq \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \text{ and}$$

$$f_i(A\vec{f} - b)_i = 0 \quad 1 \leq i \leq 4 \text{ (quadratic)}$$

New system

$$V = \{1,3\}, \; C = \{2,4\}$$

$$\begin{bmatrix} 0 & a_{12} & 0 & a_{14} \\ 0 & a_{22} & 0 & a_{24} \\ 0 & a_{22} & 0 & a_{24} \\ 0 & a_{32} & 0 & a_{34} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \begin{matrix} \geq \\ = \\ \geq \\ = \end{matrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \text{ and } \begin{matrix} f_1 = 0 \\ f_3 = 0 \end{matrix} \text{ (linear)}$$

Figure 10. Converting a quadratic system to a linear system, given V and C.

6.2 The Simplest Guess: $V = \varnothing$

To begin, note that a configuration that contains a vanishing contact point is, in a mathematical sense, *singular*. By this we mean to suggest that the existence of a vanishing contact point is a rare occurrence during a simulation.[8] A vanishing contact point occurs at a single instant of time t_0, when the contact point is in transition from existence to non-existence. Before t_0, the contact point is not vanishing, and after t_0 the contact point is non-existent and thus not considered. Thus vanishing contact points are only dealt with at an isolated instant of time t_0.

With this in mind, the obvious first guess is to set $V = \varnothing$, that is, guess that no contact points are vanishing. The linear system constructed from $V = \varnothing$ is: *choose $\vec{f}$ subject to the constraints*

$$A\vec{f} = \vec{b} \text{ and } \vec{f} \geq \vec{0}. \qquad (19)$$

Note that this problem cannot be solved using standard matrix techniques such as gaussian elimination, because of the inequality $\vec{f} \geq \vec{0}$, but must be solved as a linear programming problem. We have found that the guess $V = \varnothing$ is correct for the vast majority of cases.

6.3 Predicting a Non-Empty V

When a configuration with vanishing contact points is encountered, the initial guess $V = \varnothing$ results in an infeasible linear system. Our method of guessing V in this situation is to find an approximate solution $\vec{f}_a$ that satisfies the constraints

$$\ddot{\chi}_i(\vec{f}_a) \geq 0 \text{ and } \vec{f}_a \geq \vec{0} \qquad (20)$$

and use $\vec{f}_a$ to predict which contact points are vanishing.

Given an approximate solution $\vec{f}_a$, define the residual vector $\vec{r}$ as

$$A\vec{f}_a - \vec{b} = \begin{bmatrix} \ddot{\chi}_1(\vec{f}_a) \\ \ddot{\chi}_2(\vec{f}_a) \\ \cdot \\ \cdot \\ \ddot{\chi}_n(\vec{f}_a) \end{bmatrix} = \vec{r}. \qquad (21)$$

If $\vec{f}_a$ is in fact a correct solution, then for all vanishing contact

[8]A more precise statement is that vanishing contact points occur with measure zero.

points j, $r_j = \ddot{\chi}_j(\vec{f}_a) > 0$ and for all non-vanishing contact points k, $r_k = \ddot{\chi}_k(\vec{f}_a) = 0$. The hope is that an incorrect, yet approximate solution $\vec{f}_a$ will indicate through its residual $\vec{r}$ which contact points are vanishing. Contact point i is guessed to be vanishing if and only if $r_i > 0$. The method of section 6.1 is then used to test the guess.

6.4 Finding Approximates

The current heuristic used for finding an approximate solution is: *choose $\vec{f}_a$ to minimize the objective function*

$$z(\vec{f}_a) = \sum_{i=1}^{n} f_{a_i} \tag{22}$$

subject to the constraints

$$A\vec{f}_a \geq b \text{ and } \vec{f}_a \geq 0. \tag{23}$$

That is, we wish to find a minimum sum force solution that satisfies conditions (1) and (2). An optimal $\vec{f}_a$ can be found via linear programming since the objective function z is linear. Hopefully, an $\vec{f}_a$ that minimizes z will approximately minimize

$$\sum_{i=1}^{n} f_{a_i}\ddot{\chi}_i(\vec{f}_a) \tag{24}$$

and thus be a good approximate to a correct solution. (Recall that a *perfect* minimizer of equation (24) is a correct solution).

The physical intuition behind this choice is that correct contact forces do no net work; therefore $\vec{f}_a$ should be chosen to do as little net work as possible. Hopefully, the minimum force solution $\vec{f}_a$ will do little net work. In practice we have found that the residuals formed using z are a good predictor of the vanishing contact points.

6.5 Dealing with Incorrect Predictions

The last question is what to do when there are vanishing contact points and $\vec{f}_a$ does not predict them correctly. One could of course test all the possible guesses, but for n contact points, there are 2^n different guesses, which would give an exponential algorithm. Our current implementation exploits the fact that configurations with vanishing contact points occur infrequently. When a correct solution $\vec{f}$ cannot be found, we use the approximate $\vec{f}_a$ obtained from the minimum sum force solution. Since $\vec{f}_a$ is not correct,

$$\sum_{i=1}^{n} f_{a_i}\ddot{\chi}_i(\vec{f}_a) > 0, \tag{25}$$

and the contact force $\vec{f}_a$ adds energy to the system of bodies, producing incorrect results. The effect is mitigated by the fact that vanishing configurations are singular which means the incorrect $\vec{f}_a$ is applied for only a short time. Shortly after applying the incorrect $\vec{f}_a$ the simulator reaches a configuration where a correct $\vec{f}$ can be found. We have found that the short duration over which $\vec{f}_a$ is applied, coupled with the fact that $\vec{f}_a$ is usually a good approximate of a correct solution produces satisfactory results.

7. Additional Constraints

Holonomic constraints, which express equality constraints between bodies (e.g. articulated figures) can be added to the non-holonomic inter-penetration constraint in a consistent manner. Barzel and Barr[2] maintained holonomic constraints by introducing constraint forces that had to satisfy a linear system

$$A\vec{f} = b. \tag{26}$$

This is consistent with our own formulation since linear programming allows equality constraints. Holonomic constraints are added to our system by imposing additional linear equality constraints on $\vec{f}$. Elements representing the holonomic constraint force are added to $\vec{f}$; these elements are not subject to condition (2), the non-negativity constraint. The entire system of constraints is solved as in section 6, except that the minimum sum force solution only takes into account the non-holonomic constraint forces. This is justified since the net work done by the holonomic forces is zero if the holonomic constraint equations are satisfied[5]. If no contact occurs, only holonomic constraints remain and the system of equations is the same as in Barzel and Barr[2]. We use the sparse linear programming package to solve this linear system in $O(n)$ time. Appendix B discusses some numerical issues involved with solving equation (26).

8. Simultaneous Collisions

The linear programming formulation for resting contact can be used to improve the performance of existing collision methods for certain configurations. There are two analytical methods for resolving collisions involving multiple contact points and/or bodies: impulses at contact points can be calculated and applied one at a time or the impulses can be calculated and applied simultaneously for all contact points. We call the former view the *propagation* model of collisions and the latter the *simultaneous* model of collisions; recent papers[7, 13] have used the propagation model for collisions. The two models are the same for collisions with a single contact point but may give give different results for some multiple contact point collisions. Figure 11 shows a collision between three equal mass billiard balls with no loss of kinetic energy.

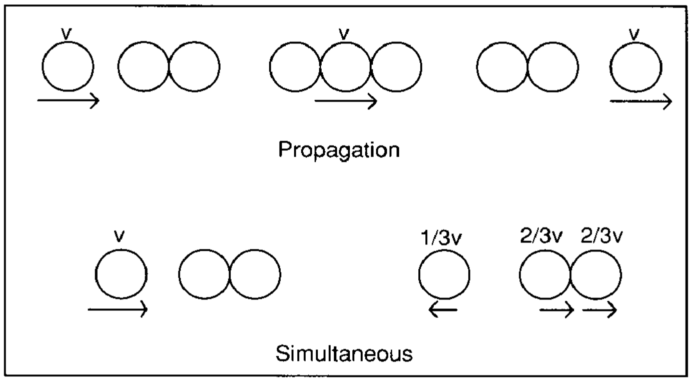


Figure 11. Propagation vs. simultaneous collisions.

The propagation model results in the right ball moving away from the motionless left and center balls. The simultaneous model results in the right and center balls moving with equal velocity away from the leftmost ball.

In other situations both models produce the same result, but the propagation model can require an excessive number of iterations to numerically converge. Figure 12 shows a ball of mass 9 colliding with a ball of mass 1, resting on an immovable floor. The collisions are totally inelastic; the propagation method iterates between calculating collision impulses between the two balls, and the smaller ball and the floor. After n iterations, the top ball will have $.9^n$ of its initial velocity v; a higher mass ratio would result in even slower convergence. In contrast, the simultaneous model would produce the limiting result (both balls at rest) in one iteration, regardless of the mass ratio.

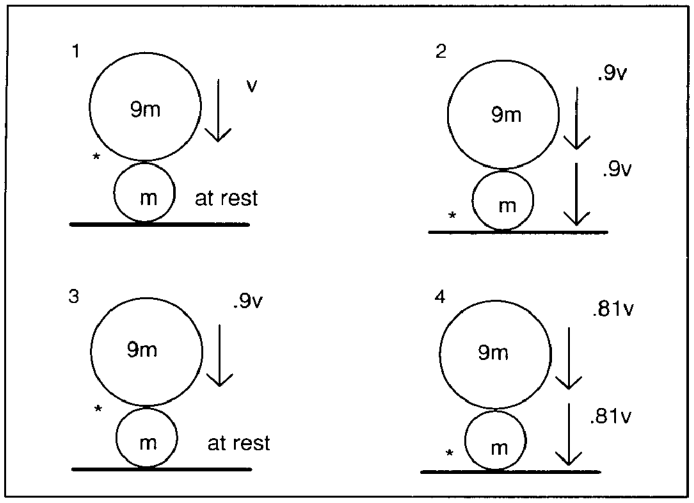


Figure 12. Convergence behavior of the propagation model. The * indicates where the collision impulse is being applied.

To calculate collision impulses for simultaneous collisions, we mimic the resting contact problem. At every contact point i, there is some impulse $J_i = j_i \hat{n}_i$ with $j_i \geq 0$ the unknown magnitude. The goal is to find a $\vec{j}$ that satisfies the laws of classical mechanics; given $\vec{j}$, the final linear and angular velocities of the bodies (the ultimate goal) can be calculated[11]. For contact point i, let v_i^- be the relative approach velocity in the $\hat{n}_i$ direction ($\dot{\chi}_i$ from section 5) and v_i^+ be the (unknown) relative recession velocity in the $\hat{n}_i$ direction. v_i^+ is a linear function of $\vec{j}$[11]. If $v_i^- > 0$, the bodies are separating.[9] Otherwise, the bodies are in resting contact ($v_i^- = 0$) or are colliding ($v_i^- < 0$).

The coefficient of restitution, ε, is defined for single contact collisions as

$$v_i^+ = -\varepsilon_i v_i^-. \tag{27}$$

The definition of ε does not readily extend to handle simultaneous collisions. The most that can be said is that if each $\varepsilon_i = 1$ then no kinetic energy is lost during the collision. We have chosen the following rules for simultaneous collisions. For each contact point in the collision, it is required that

$$v_i^+ \geq -\varepsilon_i v_i^- \tag{28}$$

i.e. the recession velocity must be at least as much as would occur for a single contact collision. The "$\geq$" is needed since body A might be pushed *away* from body B by some third body C (figure 13). Paralleling the resting contact problem, j_i is assumed to be zero if v_i^+ actually exceeds $-\varepsilon_i v_i^-$. A routine calculation shows that kinetic energy is conserved for multiple collisions when each $\varepsilon_i = 1$, and that for single contact point collisions, $v_i^+ = -\varepsilon_i v_i^-$[5]. Since v_i^+ is a linear function of $\vec{j}$, the constraints can be written as

$$v_i^+(\vec{j}) + \varepsilon_i v_i^- \geq 0, \;\; j_i \geq 0, \;\; j_i(v_i^+(\vec{j}) + \varepsilon_i v_i^-) = 0 \tag{29}$$

for $1 \leq i \leq n$. This constraint system has the same form as the constraints of section 5, and the heuristic methods of section 6 can be used to solve for $\vec{j}$. Note that for the case of two bodies colliding at a single point without friction, the system reduces to one equation in one unknown.

[9]As in section 5.1, contact points with $v_i^- > 0$ are discarded, since the bodies are separating at these contact points. This may immediately result in another round of collision resolution, but the excessive iterative behavior of figure 12 should not occur.

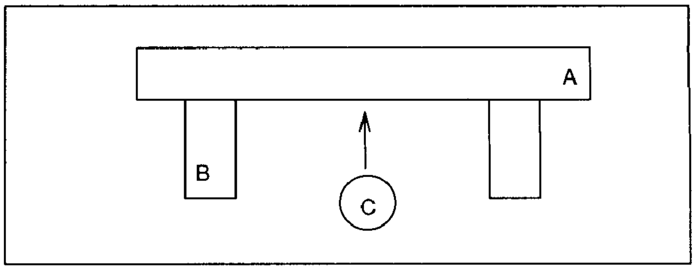


Figure 13. ***A*** **is struck from below by** ***C*** **and pushed away from** ***B*****. The impulse between** ***A*** **and** ***B*** **should be zero.**

The simultaneous collision method can also enforce holonomic constraints. Holonomic constraints are maintained by imposing additional linear equality constraints of the form $v_i^+(\vec{j}) = v_i^-$. Components of $\vec{j}$ representing the holonomic constraint impulses are not subject to the non-negativity constraint. For the case of two linked figures colliding at a single point, our method is equivalent to Moore and Wilhelms' method[13] except that our system of equation is one third the size of Moore and Wilhelms'. The reduction in size of the system is a consequence of regarding $\vec{j}$ as the only unknown in the problem; the final linear and angular momenta are expressed in terms of $\vec{j}$.

9. Conclusion

We have presented an analytical method for finding forces between contacting polyhedral bodies, based on linear programming techniques. The solution algorithm currently used is heuristic. A generalization of the formulations presented yields an analytical method for finding simultaneous impulsive forces between colliding polyhedral bodies. Both methods allow holonomic geometric constraints to be maintained. A simulator has been constructed and a variety of simulations have been produced (figure 14). The major drawback of the current solution algorithm is the necessity of solving linear systems of inequalities. Linear programming software is considerably more complex than the software used to solve systems of linear equations; software for linear equations is also more readily available. Additionally, linear equations currently enjoy a much greater diversity of solution techniques than linear programming[6].

The other major concern is that the heuristic algorithm used will occasionally fail and an approximate (but incorrect) solution will be used. This adds energy to the simulation but does not result in any unsatisfactory visual effects. We have not pursued the issue of error due to incorporating incorrect solutions into the simulator because we believe that such work would be premature. The addition of friction to the model would be likely to render such work inapplicable. Also, numerical techniques currently under investigation may preclude the need for any heuristic algorithms at all.

Acknowledgements

This research was conducted under two NSF grants entitled "Interactive Input and Display Techniques" (#DCR8203979) and "Visualization for Scientific Computing" (#ASC8715478) and an AT&T Bell Laboratories PhD Fellowship. Simulation and displays were performed on equipment generously donated by Hewlett Packard Corporation and Digital Equipment Corporation. I thank Michael Cohen for preliminary discussions on this work, Jim Cremer for talking me out of some very wrong ideas, and Don Greenberg and Roy Hall for sound editorial advice and encouragement.

Appendix A

We derive expressions for $\dot{\chi}$ and $\ddot{\chi}$:

$$\chi(t) = \hat{n}(t) \cdot (p_a(t) - p_b(t)), \tag{30}$$

$$\dot{\chi}(t) = \dot{\hat{n}}(t) \cdot (p_a(t) - p_b(t)) + \hat{n}(t) \cdot (\dot{p}_a(t) - \dot{p}_b(t)) \tag{31}$$

and

$$\ddot{\chi}(t) = \ddot{\hat{n}}(t) \cdot (p_a(t) - p_b(t)) + 2\dot{\hat{n}}(t) \cdot (\dot{p}_a(t) - \dot{p}_b(t)) + \hat{n}(t) \cdot (\ddot{p}_a(t) - \ddot{p}_b(t)). \tag{32}$$

At time t_0, $p_a(t_0) = p_b(t_0)$ so

$$\dot{\chi}(t_0) = \hat{n}(t_0) \cdot (\dot{p}_a(t_0) - \dot{p}_b(t_0)) \tag{33}$$

and

$$\ddot{\chi}(t_0) = \hat{n}(t_0) \cdot (\ddot{p}_a(t_0) - \ddot{p}_b(t_0)) + 2\dot{\hat{n}}(t_0) \cdot (\dot{p}_a(t_0) - \dot{p}_b(t_0)). \tag{34}$$

Since $\hat{n}(t_0)$, $\dot{\hat{n}}(t_0)$, $\dot{p}_a(t_0)$, and $\dot{p}_b(t_0)$ are independent of $\vec{f}$ and $\ddot{p}_a(t_0)$ and $\ddot{p}_b(t_0)$ depend linearly on $\vec{f}$[2, 5], $\ddot{\chi}(t_0)$ is a linear function of $\vec{f}$. For a vertex-plane contact with B fixed, $\dot{\hat{n}} = \ddot{p}_b = 0$ and $\ddot{\chi}(t_0) = \hat{n}(t_0) \cdot \ddot{p}_a(t_0)$.

Appendix B

The purely non-holonomic constraint equation

$$A\vec{f} \geq \vec{b}, \; \vec{f} \geq \vec{0} \text{ and } \vec{f}^T A \vec{f} - \vec{f}^T \vec{b} = 0 \tag{35}$$

often involves a singular matrix A, yielding multiple solutions. A is singular if the physical structure is overdetermined (such as a chair with more than three legs resting on a floor). Barzel and Barr[2] note that the purely holonomic constraint equation

$$A\vec{f} = \vec{b} \tag{36}$$

is also often underconstrained or overconstrained.

Underconstrained systems in both cases are easily handled by linear programming methods. Overconstrained systems that are feasible (admit a solution) are also handled by linear programming methods. However, infeasible overconstrained systems require special attention. Note that the infeasibility arises from the holonomic constraint equations. We have encountered infeasible systems when using the techniques described by Barzel and Barr[2] to assemble models. We did not encounter infeasible constraints from assembled models with holonomic constraints.

Barzel and Barr deal with infeasible holonomic constraints by selecting the least-squares solution. They find the least-squares solution by using singular-value decomposition (SVD), but note that this does not exploit sparsity and is relatively slow. SVD methods, however, cannot be used when there are non-holonomic constraints that must be maintained. One possibility is to use linear programming to find a solution that minimizes the 1-norm (as opposed to the 2-norm) of the residual in equation (36): *choose* $\vec{f}$ *such that*

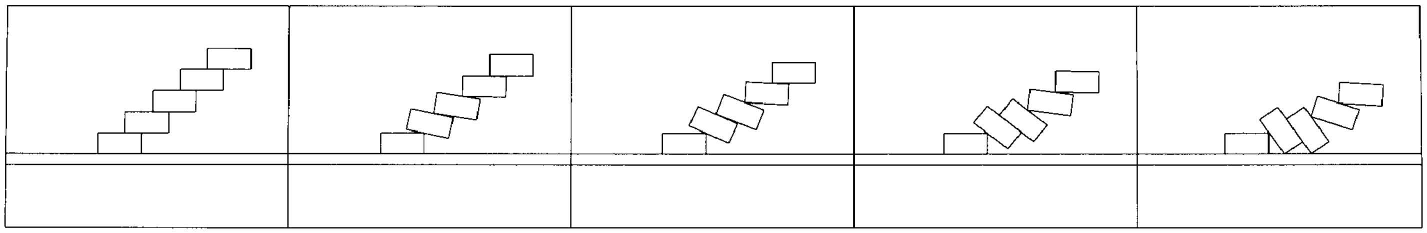
$$\|A\vec{f} - \vec{b}\|_1 \tag{37}$$

is minimized. (Equation (37) is a convex linear objective function, so the minimization is a linear programming problem). For purely holonomic systems, this approach might be faster than using an SVD method since sparsity can be exploited in linear programming methods. However, the complexity and relatively unrobust performance of linear programming methods (as compared with SVD methods) is such that the SVD method is probably preferable for purely holonomic systems.

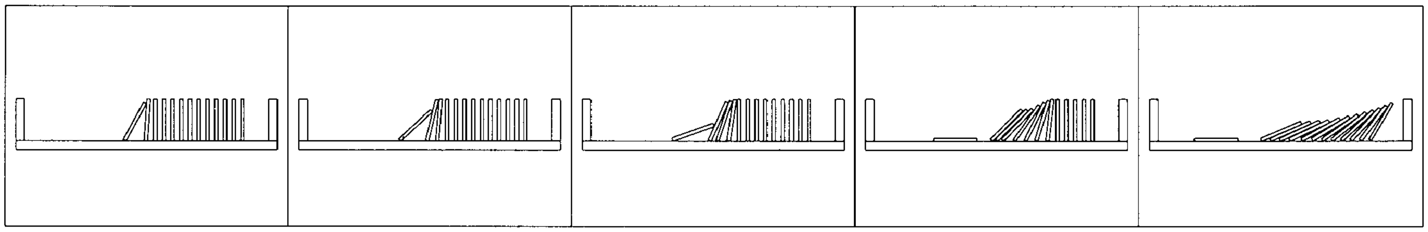
References

1. Barzel, R. and Barr, A.H., "Dynamic constraints," *Topics in Physically Based Modeling*, course notes, vol. 16, SIGGRAPH, 1987.
2. Barzel, R. and Barr, A.H., "A modeling system based on dynamic constraints," *Computer Graphics (Proc. SIGGRAPH)*, vol. 22, pp. 179-188, 1988.
3. Garey, M.R. and Johnson, D.S., *Computers and Intractability*, Freeman, New York, 1979.
4. Gere, J.M. and Timoshenko, S.P., *Mechanics of Materials*, Wadsworth, Belmont, California, 1984.
5. Goldstein, H., *Classical Mechanics*, Addison-Wesley, Reading, Massachusets, 1983.
6. Golub, G. and Van Loan, C., *Matrix Computations*, John Hopkins University Press, Baltimore, 1983.
7. Hahn, J.K., "Realistic animation of rigid bodies," *Computer Graphics (Proc. SIGGRAPH)*, vol. 22, pp. 299-308, 1988.
8. Isaacs, P.M. and Cohen, M.F., "Controlling dynamic simulation with kinematic constraints," *Computer Graphics (Proc. SIGGRAPH)*, vol. 21, pp. 215-224, 1987.
9. Kozlov, M.K., Tarasov, S.P., and Hacijan, L.G., "Polynomial solvability of convex quadratic programming," *Soviet Mathematics Doklady*, vol. 20, no. 5, pp. 1108-1111, 1979.
10. Llewellyn, R.W., *Linear Programming*, Holt, Rinehart and Winston, 1964.
11. MacMillan, W.D., *Dynamics of Rigid Bodies*, Dover, New York, 1960.
12. Marsten, R.E., "The design of the XMP linear programming library," *ACM Transactions on Mathematical Software*, vol. 7, no. 4, pp. 481-497, 1981.
13. Moore, M. and Wilhelms, J., "Collision detection and response for computer animation," *Computer Graphics (Proc. SIGGRAPH)*, vol. 22, pp. 289-298, 1988.
14. Murty, K.G., *Linear Complementarity, Linear and Nonlinear Programming*, Heldermann Verlag, Berlin, 1988.
15. Palmer, R.S., *Computational Complexity of Motion and Stability of Polygons*, PhD Diss., Cornell University, January 1987.
16. Platt, J.C. and Barr, A.H., "Constraint methods for flexible models," *Computer Graphics (Proc. SIGGRAPH)*, vol. 22, pp. 279-288, 1988.
17. Shampine, L.F. and Gordon, M.K., *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*, Freeman, 1975.
18. Terzopoulos, D., Platt, J.C., and Barr, A.H., "Elastically deformable models," *Computer Graphics (Proc. SIGGRAPH)*, vol. 21, pp. 205-214, 1987.

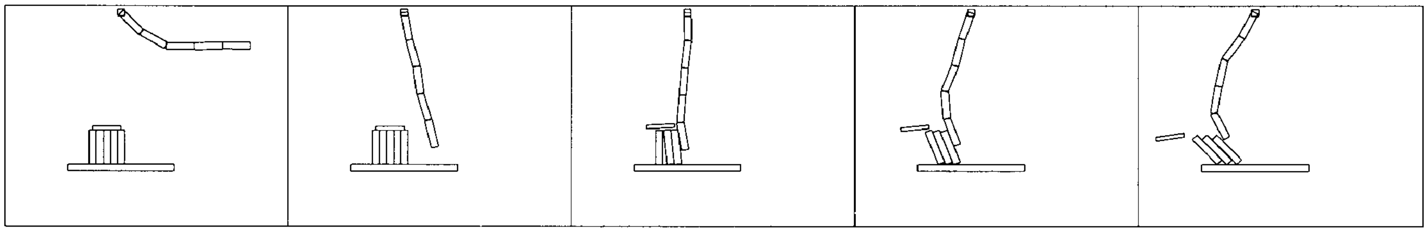
(a) Overbalanced stack of bricks.



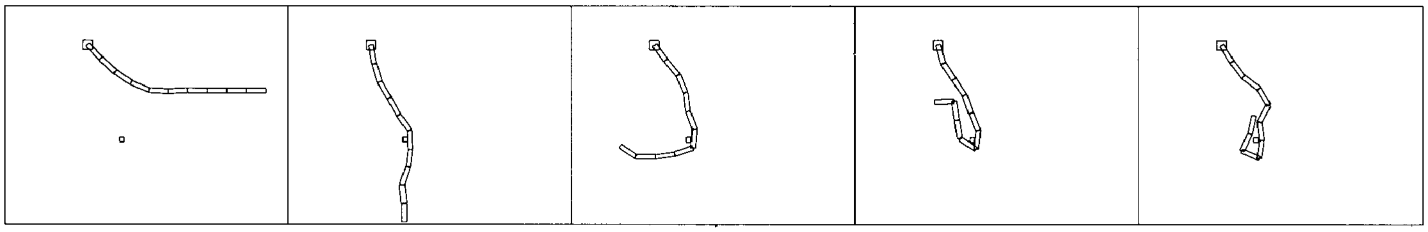
(b) Dominoes.



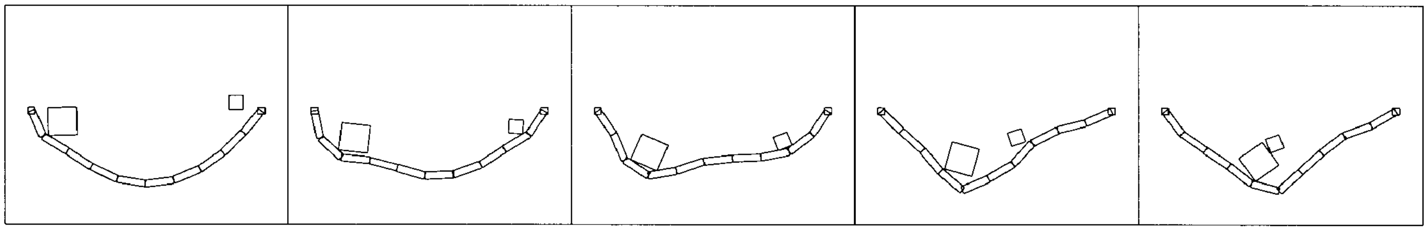
(c) Destructive chain.



(d) Chain curling around a fixed pivot.



(e) Two blocks falling into a chain.



(f) Many blocks falling into a chain.

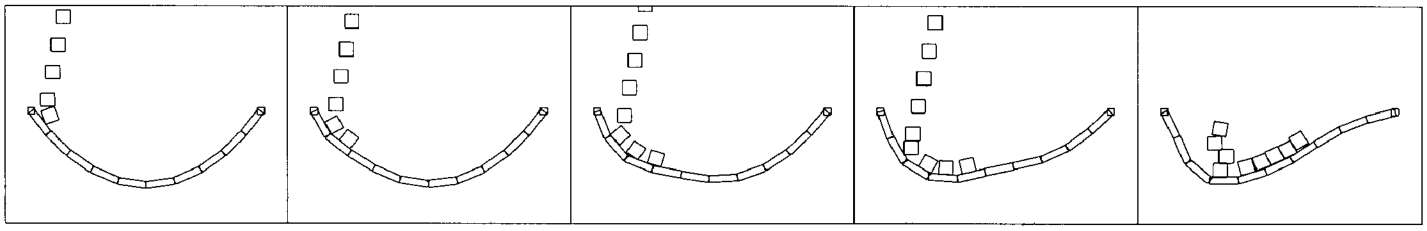


Figure 14. Assorted simulations (a-f).