Chapter 2

Non-penetrating Rigid Body Simulation

David Baraff

This paper surveys recent work on dynamic simulation of rigid bodies with non-interpenetration constraints. The problems of collision detection and contact force determination are discussed, along with both frictionless and frictional collision and contact. For the collision-detection problem, methods which deal with objects' motion as a single continuous function of time are compared and contrasted to discretization methods. For the problem of determining interaction forces between contacting objects, both penalty and constraint-based methods are considered.

Author address (May 1994): David Baraff, School of Computer Science, Carnegie Mellon University, Pittburgh, PA 15213, USA. Email: baraff@cs.cmu.edu

1.0 Introduction

The disparity between what we can model and visualize on a computer versus what we can physically simulate has become quite large. We have become fairly adept at creating high-quality images of complex models, but our ability to perform realistic physical simulation on these models lags far behind our ability to visualize them. In computer graphics and robotics applications, a major concern is modeling systems of objects' motions over time. For the most part, we wish to simulate objects' motions as realistically as is practical for a particular application. In particular, since most physical objects are impenetrable, it is important that our simulations correctly handle the issues of collision and contact between objects accurately.

In this paper we survey the problems of simulating rigid-body motion with non-interpenetration constraints and consider current solution techniques. For many computer graphics and robotics applications the abstraction of a perfectly rigid body (though physically unobtainable) perfectly characterizes the problems' degrees of freedom that are of interest to us and is in general an excellent approximation of the sorts of objects we wish to simulate.

Physical simulation with non-interpenetration constraints is important for the following reasons:

- By using physical simulation, we can more easily understand and visualize complicated mechanical systems and processes.

- Physical simulation can be used to perform experiments and test hypothesis in situations for which real-world experiments would be difficult, costly, or impractical to perform. Examples include testing robot-motion planning algorithms, or performing virtual physical experiments for educational and instructive purposes.

- A simulated physical environment can be used as a natural, intuitive means of interacting with many design and modeling tasks. The ability to interactively move 3D objects compliantly and without interpenetration could greatly simply many computer-aided design and layout systems.

- Realistic simulation, in conjunction with some degree of control, is an extremely powerful method for creating realistic computer animation.

In this survey, we concentrate on solution methods and analysis of the simulation problem rather than on the application of solution methods to particular problems.

2.0 Simulations and Impenetrability Constraints

The ability to perform realistic physical simulation requires being able to model the phenomena of collision and contact between objects. In this section, we describe the various steps necessary to perform such simulations by considering a very simple example problem.

2.1 Sample Problem

The problem we consider is simulating the dynamics of a point-mass particle in three-space ($\mathbf{R}^3$). The particle's position x at time t is a function $x(t)$. Likewise, the velocity v of the particle at time t is $v(t)$, and the particle is assumed to have unit mass. An arbitrary external force $F(t)$ acts on the particle at time t.

2.1.1 Unconstrained Motion

When there are no constraints on the particle's motion (that is, no obstacles for the particle to encounter), the motion of the particle is simple. Using Newton's $F = ma$, the differential equation

$$\ddot{x}(t) = F(t) \tag{1}$$

describes the particle's motion. Given initial conditions (values at time zero for $x(0)$ and $\dot{x}(0) = v(0)$) this sort of equation is easily solved using numerical methods. The extension and solution of equation (1) for a collection of rigid bodies is straightforward[5][19].

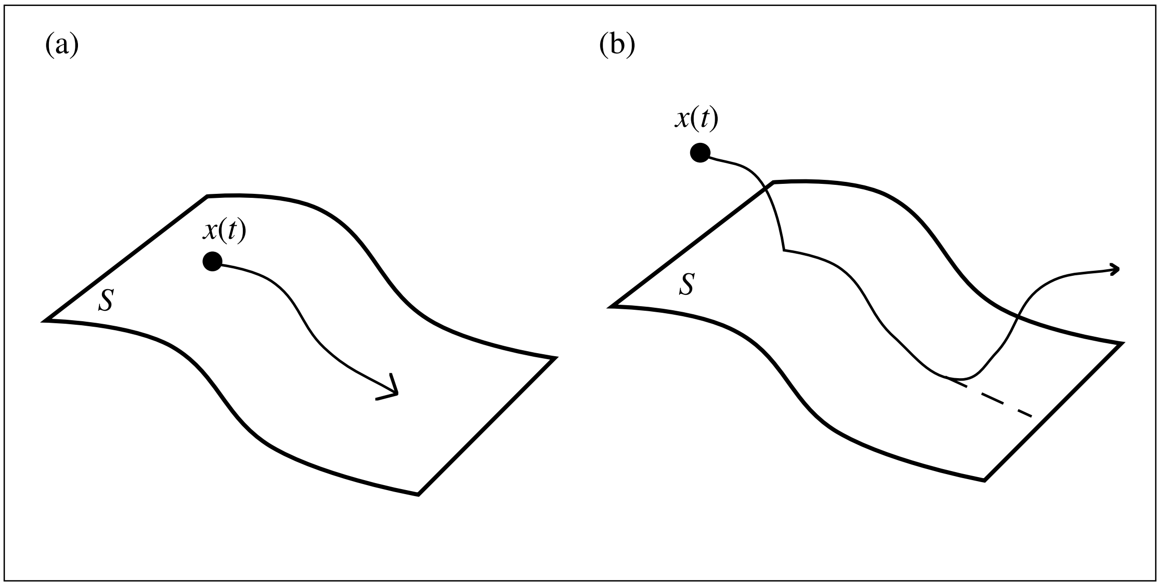
2.1.2 Equality-constrained Motion

Now suppose that S is a surface in $\mathbf{R}^3$, and supposed the particle is constrained to always lie on S. Let S be modeled implicitly by a scalar function C; that is, a point p lies on S if and only if $C(p) = 0$. Using C, the constraint on the particle can be written

$$C(x(t)) = 0. \tag{2}$$

Since this constraint can be written as an equality, this is an example of an *equality-constrained* dynamics problem (figure 1a). Equality-constrained dynamics problems have been studied extensively. Jointed figures (robot arms, mechanisms) are the most common examples of dynamic simulations with equality constraints. Simulations of jointed figures are often conceptualized in terms of this simple particle/surface problem through the use of configuration space[25].

Figure 1. A particle and a constraint surface. (a) The particle, constrained by $C(x(t)) = 0$, must remain on the surface S. (b) To satisfy the constraint $C(x(t)) \geq 0$, the particle may lie on or "above" the surface S, but may not move "below" the surface.



2.1.3 Inequality-constrained Motion

Now suppose that the constraint of the previous problem is relaxed, so that the particle is constrained to lie either on or "above" S (figure 1b). If points p lying above S satisfy $C(p) > 0$, then the constraint that the particle lie on or above S can be written

$$C(x(t)) \geq 0. \tag{3}$$

Since the constraint requires an inequality, this is an example of an *inequality-constrained* dynamics problem.

The inequality-constrained dynamics problem is a superset of the unconstrained and the equality-constrained dynamics problems and is the most difficult problem of the three. The inequality-constrained dynamics problem subsumes the unconstrained dynamics problem: if at time t the particle satisfies $C(x(t)) > 0$, then for some period of time following t, the motion of the particle can be regarded as unconstrained. Similarly, the equality constrained problem is a special case of the inequality constrained problem—the equality constraint $C(x(t)) = 0$ can be replaced by the constraints $C(x(t)) \geq 0$ and $-C(x(t)) \geq 0$.

Likewise, jointed mechanisms are a subset of the systems that can be simulated with inequality constraints. A jointed mechanism can be thought of as a simulation where contact between objects is never broken. Thus, simulation systems which can handle impenetrability constraints can automatically simulate jointed mechanisms as well, and can even combine jointed mechanisms with non-interpenetration constraints.

2.2 Simulation Steps

Lets consider the various steps that are necessary for inequality-constrained simulations. After describing each of these steps we will consider various approaches for implementing these computations during simulation.

2.2.1 Collision/Contact Detection

Simulating an inequality-constrained dynamics problem requires that we are able to detect collisions and contact. Numerical solution methods for equation (1) take discrete steps forward in time. Having determined a position $x(t_0)$ and a velocity $v(t_0)$ for the particle at some time t_0, and given the forces acting at t_0, the solver computes values $x(t_0 + \Delta t)$ and $v(t_0 + \Delta t)$. However, the numerical solver has no knowledge of the constraint placed on x by the presence of the surface S. Suppose at time t_0, the particle was just above the surface, and had a velocity towards the surface. Depending on the stepsize Δt, $x(t_0 + \Delta t)$ may lie below the surface. In this case, the simulator must recognize the fact that $x(t_0 + \Delta t)$ represents an illegal position, and then search for the instant between t_0 and $t_0 + \Delta t$ that contact first occurred. This is the problem of collision detection.

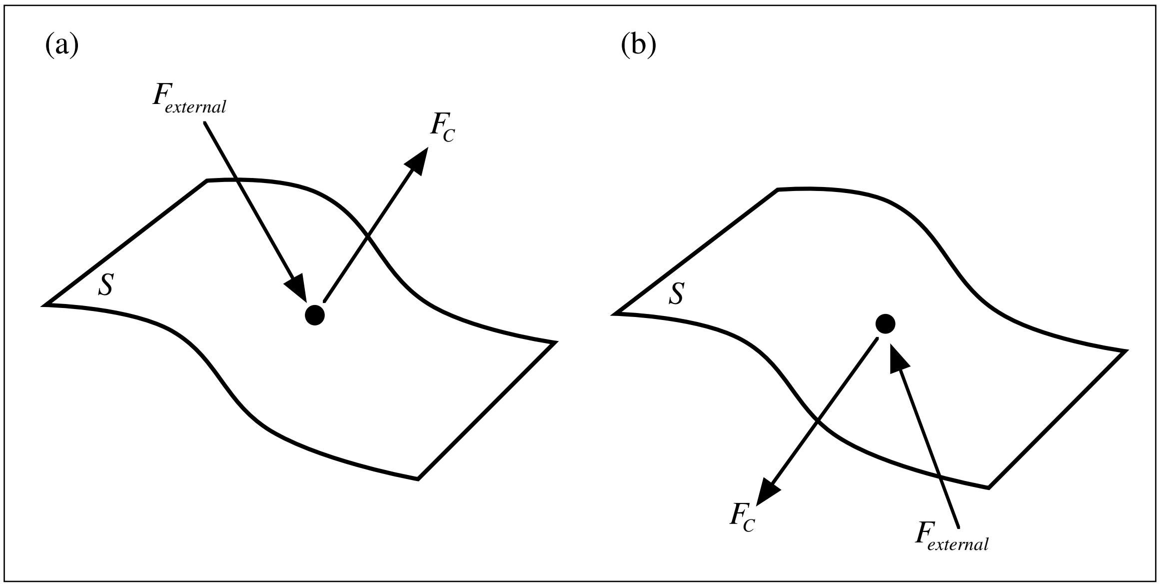
For any time t that the particle is in contact with the surface, the simulator must detect this, and note that contact occurs between the particle and the surface at the point $x(t)$. For this simple example, the entire collision/contact detection problem is clearly trivial since nothing more than evaluating $C(x(t))$ is necessary, but for systems with complicated shapes, the problem is much more difficult. Note that once the particle comes in contact with the surface, it is necessary to ensure that the particle is not allowed to move below the surface. This is in contrast to equality-constrained systems, where collision/contact detection is not an issue, and it is not necessary to determine which constraints need to be enforced and which don't (since each constraint is always enforced).

2.2.2 Constraint Forces and Impulses

Suppose that at time t_0 the particle is in contact with the surface. If the particle has a velocity toward the surface then a collision has occurred. Treating the objects as perfectly rigid, we introduce an impulsive force between the particle and the surface, that produces an immediate change in the particle's velocity. Without friction, such an impulse is computed very simply[19], and there is little more to say about the problem. Frictional collisions however are much more complicated.

Additionally, we must consider the case when objects are in contact, but are not colliding. Suppose that the particle at time t_0 has a velocity neither toward nor away from the surface. Depending on the external force $F(t)$ acting on the particle, and the velocity and curvature of the surface, we may need to add a constraint force between the particle and the surface to maintain the inequality constraint $C(x(t)) \geq 0$. For example, if gravity attempts to accelerate the particle

Figure 2. (a) For both the equality- and inequality-constrained cases, an external force $F_{external}$ acting downwards on the particle is countered by a constraint force F_C, acting in a direction normal to the surface S at $x(t)$. (b) An external force that attempts to push the particle upwards off the surface gives rise to a constraint force F_C only for the constraint $C(x(t)) = 0$. For the constraint $C(x(t)) \geq 0$, no constraint force acts and the particle moves upwards off the surface S.

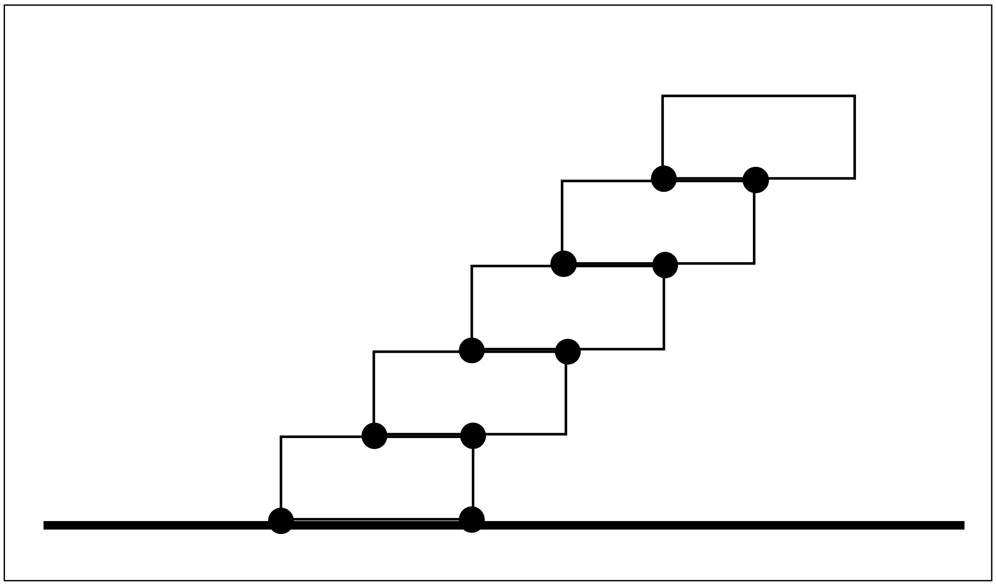


downward, an upward constraint force must act to prevent that. Even if there is no external force acting, if the surface is curved, and the particle has a nonzero velocity, a constraint force may still be necessary to keep the particle from moving downward through the surface.

In the equality-constrained problem the constraint force (or constraint forces, for a system with multiple constraints) are easily formulated for and solved. However, in the inequality-constrained case, care must be taken in specifying the correct behavior of constraint forces. For example, if an external force acts to push the particle downward (figure 2a), then in both the equality- and inequality-constrained cases, an upward pointing constraint force acts on the particle. However, if an external force acts to push the particle upward (figure 2b), then only in the equality-constrained case would a constraint force act downward to prevent the particle from moving away from the surface. In the inequality-constrained case, we do not wish constraint forces to be able to "glue" objects together; we wish to allow the particle to move away from the surface.

For the simple particle/surface problem, determining when constraints should release is trivial, since there is only one contact point. However for more complicated configurations such as figure 3, it is not immediately obvious which constraints should release and which should not. A determination of which contacts will break can be made by solving a nonlinear system of simultaneous equations.

Figure 3. A stack of overbalanced bricks. The ten contact points in this structure are marked with circles. It is not immediately obvious at which points contact will break.



2.2.3 Friction Forces

Constraint forces act to prevent interpenetration and are conservative; that is, they act normal to the contact surface and perform no net work. In contrast, if we wish to model the effect of friction, we will need to compute friction forces. Friction forces act tangentially to the contact surface to prevent or oppose sliding between objects at contact points. The Coulomb friction model specifies a relationship between the constraint force and the friction force. The relationship of these forces at each contact point depends on whether or not bodies are currently sliding relative to one another or are at relative rest.

3.0 Collision/contact determination

The two problems of collision detection and contact point determination are very similar, and can really be considered a single problem, which we shall call the *collision/contact determination* problem. There is an immense amount of literature concerning this problem, but the majority of collision/contact determination algorithms fall into one of two groups. First, the collision/contact determination problem from time t_0 to t_1 can be viewed as a single, continuous function of time. Given this viewpoint, the basic problem to be solved is "at what time, and where, do bodies first come into contact?" Second, the problem can be considered discretely, at a sequence of time values $t_0 < t_0 + \Delta t_1 < t_0 + \Delta t_2 < \cdots < t_0 + \Delta t_n < t_1$. In this viewpoint, the basic problem is "given the positions of bodies at time $t_0 + \Delta t_i$, where do bodies interpenetrate and contact each other?"

3.1 Continuum Methods

The first approach, which could be called the "continuum view", presupposes a specified motion of bodies over some time interval. Examples include Canny[8], who describes an algorithm for determining the first collision between rigid polyhedral objects with constant angular velocity. Using the constant angular velocity assumption, Canny reduces the problem of determining the first instant of collision to the problem of finding roots of (relatively low order) polynomials. Gilbert and Hong[17] relax the problem of collision detection to include arbitrary trajectories of rigid convex polyhedra. Since no closed-form solution exists for the time at which the the first intersection occurs, an iterative numerical method is used to determine this time. Their algorithm is based in part on previous work by Gilbert, Johnson, and Keerthi[18]. Von Herzen, Barr, and Zatz[43] describe an algorithm that determines the first collision between parametrically defined time-dependent curved surfaces. In this work, the motion of the bodies is not necessarily rigid (that is, the actual shape of the surfaces may vary over time). Very recently, "interval analysis" has attracted considerable attention in the computer graphics community as a method to deal with collision detection problems. (The method used by Von Herzen, Barr, and Zatz can in fact be considered a special case of interval analysis methods.) Duff[12] describes a collision detection method using interval analysis that handles rigid-body motion of implicit curved surfaces, but with restrictions on the motion path. Snyder[39] and Snyder[40] use interval analysis to find the first time of collision between both parametric and implicit time-dependent curved surfaces. The latter work treats the case when contact occurs over a region or curve.

However, algorithms of the above nature cannot be used to *globally* solve the collision detection problem of the simulation problem described in section 2.1. All of the above algorithms presuppose a specified motion path for the bodies over time; however, such a path is exactly what the simulator is trying to compute. Continuum view algorithms can however be used over suitably short intervals to guarantee that collisions between objects are not missed.

Consider for example the particle/surface problem in section 2.1. Numerical methods for solving equation (1) advance the state of the particle from an initial time t_0 to a later time $t_0 + \Delta t$. Suppose that we find that both the state at t_0 and the state at $t_0 + \Delta t$ are legal states. This does not guarantee that the particle did not contact (and even dip below) the surface at some time between t_0 and $t_0+\Delta t$. To guarantee that the particle did not collide with the surface over the interval Δt, we must consider the *entire* motion path during this interval. But as we have observed, we do not in general know this path (since we are still trying to determine if any collisions occurred), and it is unlikely that the actual path, even if free from collisions, could be described in such a manner as is needed by continuum methods.

Von Herzen, Barr and Zatz[43], and Snyder[40] use their continuum methods to solve this problem by considering the properties of the numerical differential equation solver used to solve equation (1) (or its rigid-body equivalent). Typically, the evolution of a system from time t_0 to $t_0 + \Delta t$ is modeled as a polynomial function of t. If the polynomial model is of sufficiently high degree and the interval Δt is sufficiently small, the projected state at time $t_0 + \Delta t$ differs from the actual state by a small amount. The projected state is thus considered to be an acceptable numerical solution to the problem. Since the accuracy of the simulation depends directly on the numerical method used to solve equation (1), it is argued that one might as well regard the polynomial trajectory of the system between time t_0 and $t_0 + \Delta t$ as the "true" trajectory, and check for any collisions along this polynomial trajectory. The continuum methods of Von Herzen, Barr and Zatz, and Snyder can work directly with a polynomial representation of the motion path between t_0 and $t_0 + \Delta t$ and verify that it is either free of collisions, or find the first point and time of collision. If a collision is found at time t_c, a new motion path is computed from time t_c to some future point in time, and the entire repeats itself. Continuum methods which cannot handle polynomial paths might be used by linearizing the polynomial motion between time t_0 and $t_0 + \Delta t$ into smaller linear segments.

3.2 Discrete Methods

The second approach, which could be described as the "discrete view," involves the geometric analysis of bodies at a fixed instant of time. The goal under this approach is to produce algorithms that solve a single, static instance of a problem with optimal asymptotic time complexity. Work of this sort includes an approximately $O(n \log n)$ algorithm by Cameron and Culley[7] for computing the distance between two convex polyhedra with n vertices. Gilbert, Johnson, and Keerthi[18] describe an algorithm for computing the minimum distance between convex polyhedra with n vertices; the algorithm appears to have a running time somewhat larger than $O(n)$ but smaller than $O(n \log n)$. Related work by Gilbert and Foo[16] extends the algorithm to handle smooth convex shapes and concludes that smooth representations become more efficient than polyhedral representations in the neighborhood of $n = 100$. For the problem of determining disjointness of two convex polyhedra with n vertices (without regard to the distance between them), an $O(n)$ algorithm is readily available, based on Megiddo's work on linear programming problems with constant dimension[34][37].

Unfortunately, the above algorithms, although they achieve small asymptotic time complexity, do so at the price of large run time constants. For example, the linear time algorithm for deciding disjointness between convex polyhedra with n vertices is not practical unless n is quite large. More importantly, the use of discrete view algorithms essentially ignores any similarity that may exist with the current collision/contact determination problem and previous collision/contact determination problems. Proper use of previous information can result in very simple, yet efficient collision/contact determination algorithms.

The earliest example of such an algorithm occurs in work by Cundall[11]. Cundall's work on dynamic simulation of polyhedral blocks uses a collision/contact determination algorithm that specifically exploits information obtained from the previous determination. However, Cundall's approach is somewhat more complicated than needed for dynamic simulation and is restricted to polyhedra. The collision detection algorithm in Gilbert, Johnson, and Keerthi[18] is structured so that it can use information previously computed to obtain a faster running time, but this is not a main consideration of the algorithm. Recently, work by Baraff[2] and Lin and Canny[26][27] has focused on collision detection methods for dynamic simulation that efficiently reuse previously computed information. In particular, Lin and Canny describe a collision detection algorithm for convex polyhedra that takes roughly $O(1)$ time to test a pair of polyhedra. Baraff[3] describes a coherence based bounding box test that detects overlap between n bounding boxes in roughly $O(n)$ time over the course of a simulation. Methods for coherence-based collision detection among convex curved surfaces are also described. A disadvantage of coherence-based detection methods is that they may miss collisions that occur between the sampled time steps. (Consider simulating a bullet fired through a thin pane of glass). While continuum based methods do not suffer from this problem, the more comprehensive continuum methods (notably those based on interval analysis) are considerably more complicated to implement and are not (currently) as fast as coherence-based methods.

4.0 Frictionless Systems

Assuming that one can solve the collision detection problem and determine when and where objects contact each other, the problem of enforcing non-interpenetration constraints remains. In this section, we describe the two basic approaches for preventing interpenetration between contacting objects without friction. The first approach computes constraint forces that are designed to exactly cancel any external accelerations that would result in interpenetration. This method requires solving nonlinear systems of equations, and is fairly complicated. However, it results in simulations where interpenetration is completely eliminated (within numerical tolerances).

The second method, called the *penalty method* is much less complicated, but does not completely eliminate interpenetration. Essentially, the penalty method models contacts by placing a spring (possibly damped) at each contact point, between the two contacting bodies. Interpenetration is allowed between the bodies at a contact point, but as the amount of interpenetration increases at a contact point, a repulsive restoring or "penalty" force acts between the objects, pushing them apart.

4.1 The Constraint-Based Method

We can maintain non-interpenetration constraints by computing a constraint force at each contact point that depends directly on any external forces acting on the bodies in the system. Each constraint force acts in a direction normal to the contact surface at the point of contact and exactly prevents interpenetration.

Consider again the inequality-constrained particle/surface problem of section 2.1. We introduce a constraint force $F_c(t)$ into the problem. Let the surface normal at the point $x(t)$ be given by the vector $\nabla C(x(t))$ where

$$\nabla C(x(t)) = \begin{pmatrix} \frac{\partial C}{\partial x}\big(x(t)\big) \\ \frac{\partial C}{\partial y}\big(x(t)\big) \\ \frac{\partial C}{\partial z}\big(x(t)\big) \end{pmatrix}.$$

Since $F_c(t)$ acts in this normal direction, we can write $F_c(t) = f_N(t)\nabla C(x(t))$ where $f_N(t)$ is an unknown scalar. We will assume that $\nabla C(x(t))$ points towards the region of space that the particle is allowed to occupy. The equation of motion for the particle then becomes

$$\ddot{x}(t) = F(t) + F_c(t) = F(t) + f_N(t)\nabla C(x(t)). \tag{4}$$

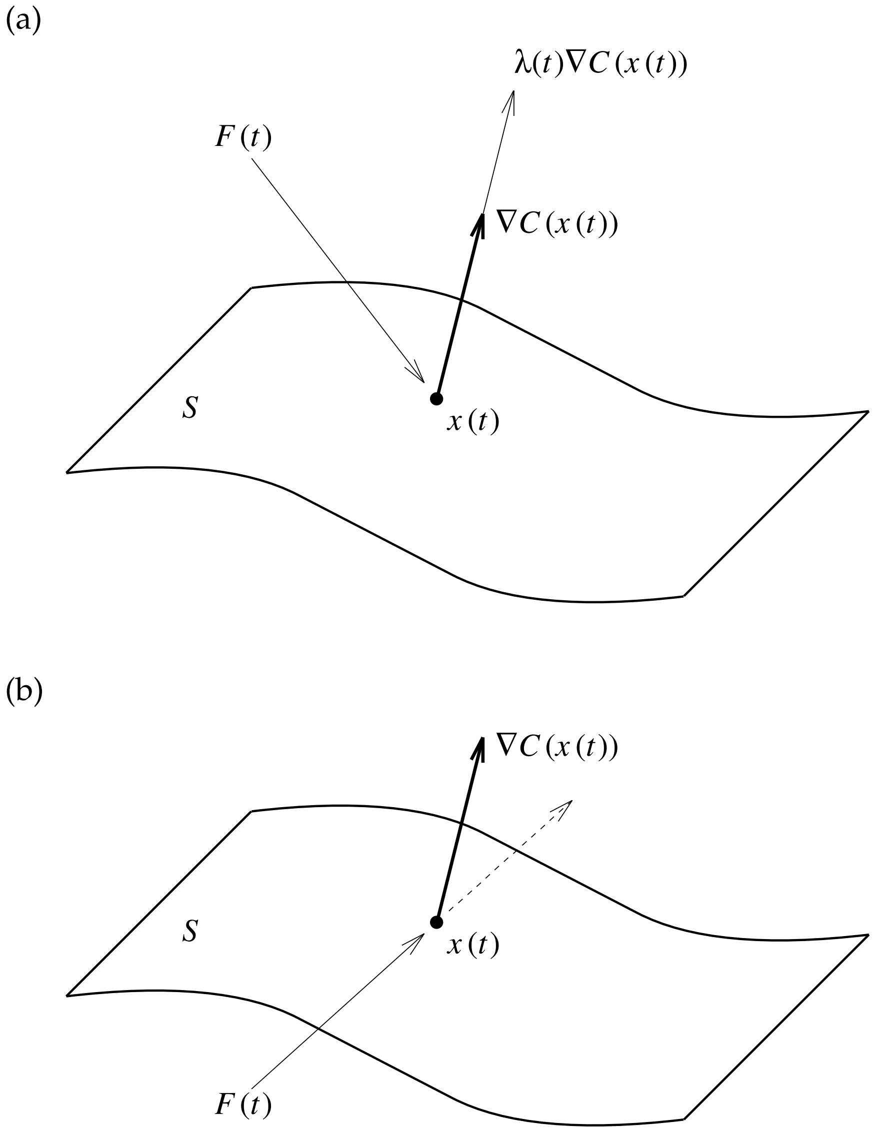
Since we are modeling contact between objects, the constraint force $F_c(t)$ acting on the particle must be directed away from the surface, in the $\nabla C(x(t))$ direction. That is, the surface can exert a "pushing" force on the particle to prevent interpenetration, but cannot "pull" on the particle to prevent it from leaving the surface. This means that $f_N(t)$ must satisfy $f_N(t) \geq 0$. If $f_N(t)$ was allowed to be negative, the particle, once in contact with the surface, could be held to the surface forever after.

4.1.1 Collisions

If the particle's velocity at time t_0 is directed inwards, towards the surface when it first comes into contact with the surface, the particle is said to have collided with the surface. Since the particle and the surface are both rigid, the particle's velocity must change discontinuously at the moment of impact. This is modeled by letting $F_c(t_0)$ be an *impulsive* force; that is, $F_c(t_0)$ has the dimensions of mass times velocity, and acts for a single instant. The strength of the impulse is calculated according to Newton's empirical law of restitution[23].

For the case when rigid bodies collide with only a single point of contact, the constraint impulse $F_c(t_0)$ is trivially computed by solving a system of one equation in one unknown. (The *direction* of $F_c(t_0)$ is already known to be normal to the surface. Only the scalar magnitude $f_N(t_0)$ needs to be determined.) If equality constraints involving one or both of the colliding bodies exist (as happens when an articulated rigid body undergoes a collision), additional impulsive constraint forces are needed to maintain the equality constraints. In this case, a number of constraint impulses must be computed simultaneously. If there are n equality constraints to be maintained, then a system of $n+1$ linear equations in $n+1$ unknowns must be solved. This is simply a special case of solving equality constrained dynamics problems, and is not particularly difficult. If the collision involves more than one contact point, the contacts can be regarded as occurring either simultaneously, or sequentially. Recent work on rigid-body simulations with impulsive collision forces include Lötstedt[30], Featherstone[14], Moore and Wilhelms[36], Baraff[1], and Cremer[10].

Figure 4. Constraint forces parallel to the surface normal $\nabla C(x(t))$ for an inequality constrained problem. (a) A constraint force $F_c(t)$, with magnitude $\lambda(t) = f_N(t) > 0$ and direction $\nabla C(x(t))$ acts to prevent the particle from accelerating downwards. (b) The external force $F(t)$ accelerates the particle upwards, and contact is broken. No constraint force acts on the particle.



4.1.2 Contact

Now consider the case when the particle in section 2.1 has no velocity normal to the surface, but only velocity tangential to the surface (or possibly no velocity at all). In this case, we need to compute a constraint force that maintains the non-interpenetration constraint. Again, for a single point of contact, the scalar $f_N(t_0)$ is easily computed. The conditions on $f_N(t_0)$ can be stated as follows. First, since the constraint force must be repulsive, we require $f_N(t_0) \geq 0$. Second, the constraint force $F_c(t_0) = f(t_0)\nabla C(x(t_0))$ must be strong enough to prevent the particle from accelerating downwards. If we let the scalar a_N denote the acceleration of the particle in the normal direction ∇C at time t_0, then to prevent interpenetration, we require $a_N \geq 0$ as well. If $a_N = 0$, the particle remains on the surface, otherwise $a_N > 0$ indicates the particle breaks contact with the surface. Lastly, since constraint forces must be workless, the constraint force must be zero if the particle is breaking contact. This condition is written as $f_N(t_0)a_N = 0$ which ensures that if a_N is positive, $f_N(t_0)$ is zero.

For the simple particle surface example, it turns out that a_N is a linear function of $f_N(t_0)$ and can be expressed as $a_N = cf_N(t_0) + d$. Thus, solving the system

$$f_N(t_0) \geq 0, \qquad cf_N(t_0) + d \geq 0 \qquad \text{and} \qquad f_N(t_0)(cf_N(t_0) + d) = 0 \tag{5}$$

is trivial for a single particle. (In the frictionless case, c is always positive, ensuring that a unique solution $f_N(t_0)$ exists.)

Suppose however that we have a configuration with n contact points. For each contact point, there will be a constraint force normal to the surface at the contact point. Let us denote the magnitude of the ith constraint force, at time t_0, as f_{Ni}. Similarly, let the acceleration normal to the surface at the ith contact point be a_{Ni}. If we let $\boldsymbol{f}_\mathbf{N}$ be the vector of f_{Ni}'s, and $\mathbf{a_N}$ be the vector of a_{Ni}'s, then we can write[28][2]

$$\mathbf{a_N} = \mathbf{A}\boldsymbol{f}_\mathbf{N} + \mathbf{b} \tag{6}$$

where A is an $n \times n$ positive semidefinite symmetric matrix, and $\mathbf{b}$ is an n-vector.

The same conditions as in equation (5) must hold at each contact point. This yields the system

$$\boldsymbol{f}_\mathbf{N} \geq \mathbf{0}, \qquad \mathbf{a_N} \geq \mathbf{0}, \qquad \text{and} \qquad f_{Ni}a_{Ni} = 0 \qquad for \;\; 1 \leq i \leq n. \tag{7}$$

where $\mathbf{0}$ is the null vector of n dimensions. Since each f_{Ni} and a_{Ni} must be nonnegative, requiring each product $f_{Ni}a_{Ni}$ to be zero is equivalent to requiring

$$\sum_{i=1}^{n} a_{Ni}f_{Ni}$$

to be zero. Since this is just the dot product of f_{Ni} and a_{Ni}, using $\mathbf{a_N} = \mathbf{A}\boldsymbol{f}_\mathbf{N} + \mathbf{b}$ we can rewrite the conditions on f_{Ni} as

$$\boldsymbol{f}_\mathbf{N} \geq \mathbf{0}, \qquad \mathbf{a_N} = \mathbf{A}\boldsymbol{f}_\mathbf{N} + \mathbf{b} \geq \mathbf{0}, \qquad \text{and} \qquad \boldsymbol{f}_\mathbf{N}{}^T\mathbf{a_N} = \boldsymbol{f}_\mathbf{N}{}^T(\mathbf{A}\boldsymbol{f}_\mathbf{N} + \mathbf{b}) = 0. \tag{8}$$

Equation (8) is known as a linear complementarity problem (LCP). Equivalently, we can regard this system as a quadratic programming problem by viewing equation (8) as an attempt to minimize the quadratic term $\boldsymbol{f_N}^T \mathbf{a_N}$ subject to the conditions $\boldsymbol{f_N} \geq 0$ and $\mathbf{a_N} \geq 0$.

Relatively little has been written about simulation methods using constraint forces (as compared with the amount of work done using the penalty method). Descriptions of the system of equations in terms of a quadratic program for calculating multiple constraint forces without friction (for polyhedra) appear in Kilmister and Reeve[23] and Ingleton[21]. Cottle[9] clarifies and simplifies some of Ingleton's results. In particular, it is shown that for frictionless systems, the matrix $\mathbf{A}$ is positive semidefinite. As a result, equation (8) always has at least one solution $\boldsymbol{f_N}$. Although multiple solutions are possible (if $\mathbf{A}$ is singular), the *acceleration* of the bodies from any solution $\boldsymbol{f_N}$ is unique.

All of the above work took place before the discovery of the importance of *P* and *NP* time-complexities. As a result, the computational complexity of computing multiple contact forces was not of concern in any of the above work. In work done subsequent to this discovery, Erdmann[13] discusses the problem of computing constraint forces without friction, and notes that the problem is simply solved by an exhaustive search method, requiring exponential time. The earliest description of a simulator using exact methods to calculate constraint forces (by quadratic programming) appears to be by Lötstedt[29]. Lötstedt notes that the quadratic program can be solved efficiently because it is convex. (It was shown in 1979 that convex quadratic programming is a polynomial-time problem.) More recently, Featherstone[14] has described a heuristic algorithm for solving the quadratic programs generated for configurations without friction that is intended to improve upon the obvious (but exponential) exhaustive search method. However, Featherstone's heuristic requires that configurations of bodies be statically determinate; that is, the normal forces that prevent inter-penetration for the configuration must be unique. Later work in Baraff[2] extends this formulation to deal with contacting curved surfaces. Baraff[4] discusses issues involved in the actual solution of equation (8) and presents simulations that solve equation (8) using quadratic programming and linear complementarity methods.

4.2 The Penalty Method

A vast number of simulations[11][20][24] [33][35][36][41] [42][45] have employed the penalty to enforce non-interpenetration constraints. Applications include the simulation of deformable bodies, cloth, and articulated rigid bodies. The penalty method is a very attractive model in some respects, because it is extremely simple to implement and very versatile. As a numerical method, it is nowhere near as complex as constraint-based methods. In particular, exploiting parallelism and/or hardware implementations with the penalty method is much easier to imagine than with the constraint-based methods discussed above.

The penalty method for constrained dynamics problems is based on a numerical solution method for *constrained optimization* problems. In both fields, the penalty method converts a constrained problem to an unconstrained problem where deviation from the constraint is penalized; that is, in the new problem, satisfaction of the constraint is encouraged, but not strictly enforced.

In optimization, a typical equality constrained problem such as

$$\text{minimize } f(z) \quad \text{such that} \quad g(z) = 0 \tag{9}$$

can be rewritten as an unconstrained problem

$$\text{minimize } f(z) + kg(z)^2 \quad \text{as } k \to \infty. \tag{10}$$

The term $kg(z)^2$ is called the *penalty function*. The idea is that as k grows larger, potential solutions for z must make $g(z)^2$ smaller, to minimize equation (10). In the limit, as k goes to infinity, the solution of equation (10) must satisfy $g(z) = 0$ while minimizing $f(z)$. In practice, z is obtained by solving equation (10) for a series of increasing values of k until the series of solutions converges (within numerical tolerance) to a limit. Although the method has a theoretically firm basis, in practice, it is not a very robust numerical method. The main problem is that as k grows, equation (10) can become very poorly conditioned and difficult to solve[15]. The main attraction of the method is that it a very simple way of turning a constrained problem into an unconstrained problem.

The equality constrained particle/surface problem in section 2.1 can be converted to an unconstrained dynamics problem in a similar fashion. The idea is that the particle is not explicitly constrained to lie on the surface. However, if the particle drifts off the surface, a penalty force acts on the particle, in a direction normal to the surface, so as to pull the particle back to the surface. Typically, the penalty force is modeled as a linear spring force; that is, the penalty force pulls the particle back towards the surface with a a strength equal to some constant k times the distance of the particle from the surface. If we let $p(x(t))$ denote the closest point on the surface to $x(t)$, then the penalty force is[1]

$$-k\,(x(t) - p(x(t)))\,.$$

Note that as $x(t)$ drifts farther from the surface, the force $-k\,(x(t) - p(x(t)))$ grows larger in magnitude. The introduction of the penalty force reduces the dynamics problem to simply an evaluation of the ordinary differential equation

$$\ddot{x}(t) = -k\,(x(t) - p(x(t))) + F(t). \tag{11}$$

If the penalty method for dynamics were to completely emulate the penalty method for constrained optimization, the simulation would be repeated with increasing values of k until the behavior of the particle approached a limit. However, the penalty method, as used by dynamics, chooses a single value for k. If the value chosen for k is too small, it may do an inadequate job

[1]If the surface is a plane, the closest point on the plane to the particle is unique. Otherwise, the closest point on the surface to the particle is unique as long as the particle remains in some neighborhood containing the surface; the extent of this neighborhood depends on the curvature of the surface. Since the penalty method is intended to keep the particle close to the surface, the closest point on the surface to the particle is always assumed to be well-defined.

of enforcing the constraint; in this case, the simulation would be rerun with a higher value of k. Unfortunately, as in the optimization problem, ill-conditioning occurs as k grows larger, in the guise of "stiffness" for the differential equation of motion. Stiff differential equations are expensive and difficult to solve.

The penalty method can also be used for the inequality constrained particle/surface problem. The modification is straightforward; whenever the particle lies below the surface, a penalty force acts upwards on the particle, as in the equality constrained case. However, when the particle lies above the surface, no penalty force acts on the particle. No distinction need be made between a collision at a contact point and simple resting behavior at a contact point. Moore and Wilhems[36] describe a system in which constraint-based methods are used to compute impulsive constraint forces for collisions and penalty forces are used for resting contact.

A basic question on the use of the penalty method for simulation arises: since the penalty method does not exactly maintain constraints, but allows some deviation, how does the penalty method compare with the constraint-based method of exactly enforcing the non-interpenetration constraints? For the equality-constrained case where no constraints are allowed to break, Rubin and Ungar[38] show that as the penalty constant k is increased without limit, the trajectory of the particle converges to the exact solution obtained using constraint methods. Baraff[3] contains a simpler proof of this behavior of the penalty method, and some additional discussion on modifying the penalty method for the inequality constrained case. While the penalty method is shown to work over any time interval which does not involve a constraint either breaking or forming, it has not been proven that simply setting penalty forces to zero when contact breaks will necessarily give the correct behavior.

Although the penalty method is useful in some contexts (namely largely static environments), it has become increasingly apparent that the performance of spring-and-damper systems for simulating rigid body motion is inefficient and has unpredictable accuracy in dynamic settings, where objects undergo large-scale displacements. For example, setting the penalty stiffnesses (the spring constants) too high significantly increases the cost of the simulation, while setting the stiffness too low can lead to an unacceptable degree of interpenetration. Moreover, a good choice for the penalty stiffnesses can vary greatly over the course of a simulation, and it is usually impossible to make a reasonable prediction for a suitable stiffness. Additionally, it is unclear how (or even if) one can use the penalty method to determine where contact between objects should break, and the penalty-force completely removed.

5.0 The Coulomb Friction Model

Friction adds considerable complication to rigid-body simulation. In this section, the classical Coulomb friction model is defined. Although the penalty method can be extended to add a tangential friction-like force, it is not clear how or if the complete Coulomb friction model, with its varying "dry" and "sliding" friction modes, can be best accommodated within the framework of the penalty method. In the next section, we survey work on constraint-based rigid-body simulation with friction.

5.1 Coordinate geometry at a contact point

We will call the force that arises between two bodies at a contact point a *contact force*. A contact force is the sum of a workless constraint force, and a friction force. The constraint force does no work on the system and acts normal to the contact surface; the constraint force is sometimes called the *normal force*. The friction force acts tangential to the contact surface, and may perform work by dissipating kinetic energy between contacting bodies.

At each contact point, let us define a local coordinate system as follows. Let $\hat{n}$, $\hat{t}_x$ and $\hat{t}_y$ denote mutually perpendicular unit vectors. The vector $\hat{n}$ is normal to the contact surface at the contact point while $\hat{t}_x$ and $\hat{t}_y$ span the plane tangent to the contact surface. We have already defined f_N and a_N, as the contact force and relative acceleration in the $\hat{n}$ direction. (In situations where we are discussing only a single contact point, we will drop subscripts and write f_N and a_N in place of f_{Ni} and a_{Ni}.) Similarly, we decompose the friction force along the $\hat{t}_x$ and $\hat{t}_y$ axes into magnitudes f_x and f_y.

Since friction acts to oppose any slipping motion, let us consider the slip velocity tangent to the contact surface. Let s denote the relative velocity between bodies at their point of contact and let s be projected onto the tangent plane and then decomposed along the $\hat{t}_x$ and $\hat{t}_y$ axes into magnitudes v_x and v_y:

$$v_x = \hat{t}_x \cdot s \qquad \text{and} \qquad v_y = \hat{t}_y \cdot s. \tag{12}$$

Thus, the tangential slip velocity (henceforth called the tangential velocity) is

$$v_x\hat{t}_x + v_y\hat{t}_y.$$

When the tangential velocity s is zero, let the tangential slip acceleration be described by magnitudes a_x and a_y where

$$a_x = \hat{t}_x \cdot \frac{d}{dt}s \qquad \text{and} \qquad a_y = \hat{t}_y \cdot \frac{d}{dt}s. \tag{13}$$

If the tangential velocity is initially zero, then as long as the tangential acceleration $a_x\hat{t}_x + a_y\hat{t}_y$ remains zero (and contact is not broken), the tangential velocity also remains zero.

The Coulomb model of friction is an accepted empirical relationship between the normal force magnitude f_N and the friction force magnitude $f_x^2 + f_y^2$. At all times, the condition

$$f_x^2 + f_y^2 \leq (\mu f_N)^2 \tag{14}$$

holds, where μ is a coefficient of friction that depends on material properties, and may be different at each contact point. If the tangential velocity is nonzero, we say that the friction force is *dynamic*; otherwise, the friction force is called *static*.

Typically, the coefficient μ of static friction (μ_{static}) is larger than the coefficient μ of dynamic friction ($\mu_{dynamic}$). Aside from this, μ is roughly independent of the tangential velocity. Since we are computing contact forces for a specific instant of time t_0, we know which coefficient of friction to use, and we will not bother to make a distinction between μ_{static} and $\mu_{dynamic}$. (We also will not bother to index μ over different contact points.)

5.2 Dynamic friction conditions

When dynamic friction arises, the friction force magnitude achieves its upper bound of μf_N. Also, the friction force acts directly opposite the slip velocity. Knowing that $f_x^2 + f_y^2 = (\mu f_N)^2$ and that the friction force is opposite the slip,

$$f_x\hat{t}_x + f_y\hat{t}_y = -\mu f_N \frac{v_x\hat{t}_x + v_y\hat{t}_y}{\sqrt{v_x^2 + v_y^2}} \tag{15}$$

or

$$f_x = -\mu f_N \frac{v_x}{\sqrt{v_x^2 + v_y^2}} \qquad \text{and} \qquad f_y = -\mu f_N \frac{v_y}{\sqrt{v_x^2 + v_y^2}}. \tag{16}$$

Thus, given μ and the tangential velocity, the dynamic friction force can be written solely in terms of f_N. Since

$$\begin{aligned} f_N\hat{n} + f_x\hat{t}_x + f_y\hat{t}_y &= f_N\hat{n} + \frac{-\mu f_N v_x\hat{t}_x - \mu f_N v_y\hat{t}_y}{\sqrt{v_x^2 + v_y^2}} \\ &= f_N(\hat{n} + \frac{-\mu v_x\hat{t}_x - \mu v_y\hat{t}_y}{\sqrt{v_x^2 + v_y^2}}), \end{aligned} \tag{17}$$

the direction of the net contact force is predetermined given the tangential velocity and μ.

5.3 Static friction conditions

The conditions between f_x, f_y and f_N are more complex when the tangential velocity is zero, resulting in static friction. There are two possibilities, according to whether or not the tangential acceleration is zero. First, f_x and f_y may satisfy

$$a_x = a_y = 0 \qquad \text{and} \qquad f_x^2 + f_y^2 \leq (\mu f_N)^2, \tag{18}$$

indicating that the friction remains static. Second, f_x and f_y may satisfy

$$f_x^2 + f_y^2 = (\mu f_N)^2 \qquad \text{and} \qquad (f_x\hat{t}_x + f_y\hat{t}_y) \cdot (a_x\hat{t}_x + a_y\hat{t}_y) \leq 0, \tag{19}$$

indicating that slipping is occurring, and that the static friction force has become dynamic. In this case, the friction force must oppose the initial tangential acceleration and have magnitude μf_N. The friction force direction does not have to be directly opposite the tangential acceleration direction.

6.0 Collisions with Friction

When two bodies collide at a contact point with friction, we can consider the collision as taking place over some small but nonzero time interval. We then consider the limiting value of the product of the force acting between the bodies and the time interval, as we let the time interval tend to zero. This limiting value is the impulsive force of the collision. During the time interval of the collision, the normal and friction forces must satisfy the conditions of the previous two sections. It is possible that during this time interval, the direction of the relative tangential velocity between the bodies may change. For planar collisions, this happens if the relative tangential velocity passes through zero and reverses itself. Wang and Mason[31,44] describe the computation of the impulsive forces for all possible planar collisions involving one contact point. For three-dimensional collisions, the velocity direction can vary in the tangent plane of the collision, and the computational problem becomes much more difficult.

Thus, the analysis of a frictional collision involving a single contact point is complicated, although the actual computation is simple, for two-dimensional systems. However, Wang and Mason's results are difficult to extend to three dimensions, which can be seen by considering Keller[22], which presents a differential description of the three-dimensional motion of bodies undergoing a collision with friction at a single contact point. The differential equations appear easy to solve numerically, but very difficult to treat analytically. Keller's analysis involves regarding the collision as occurring over an arbitrarily small time interval, and examining the limiting behavior of the system as the time interval is brought to zero. The animation system described by Moore and Wilhelms[36] computes constraint impulses directly for collisions involving a single contact point. Moore and Wilhelms use a simplified description of the Coulomb friction law to compute frictional impulses at a single contact point in three dimensions.

Modeling simultaneous collisions is more complicated. For example, consider a cube dropped onto a level plane so that all four vertices of the bottom face of the cube strike the plane together. There are large computational savings to be gained by modeling the collisions as occurring simultaneously. Cremer[10] discusses the advantages and disadvantages of the simultaneous model of collisions. Lötstedt[30] computes simultaneous frictional impulses in three dimensions by using a modification of the Coulomb friction law that causes impacts to dissipate as much energy as possible. In general, it is unclear how to deal correctly with simultaneous impacts with friction, in either two or three dimensions. The simulator described in Baraff[3] computes simultaneous frictional impulses for three-dimensional bodies, based on a modification of the model proposed by Lötstedt. For frictionless systems, the model is equivalent to the model in Featherstone[14]. However, for systems with friction, the model does not always properly conserve energy for impacts[6,44]. The general problem of correctly computing frictional impulses in three dimensions, and simultaneous frictional impulses in either two or three dimensions is still an open problem.

7.0 Contact with Friction

Finding contact forces that satisfy the conditions in section 5.0 is extremely difficult. In fact, it is even possible for situations to arise in which *no* solution for the contact forces exist! When a solution does exist, it can be found (for two-dimensional problems) by an obvious exhaustive search method. Unfortunately, this is not a suitable solution method in practice because it requires exponential time in the number of contact points.

7.1 Dynamic Friction

When only dynamic friction is present at all contact points, the formulation of the contact forces is greatly simplified. Since the direction of the contact force at a contact point with dynamic friction is known *a priori*, it is only the magnitude of the force that need be calculated. As in the frictionless case, an LCP or quadratic program can be formulated to find the contact force magnitudes.

Unlike the frictionless case, there is no guarantee that a solution to the LCP exists. Furthermore, if distinct solutions do exist, again, unlike the frictionless case, these distinct solutions may give rise to different accelerations of the objects. We call the first possibility, nonexistence of solution, *inconsistency*. The second problem, nonuniqueness of motion behavior, is called *indeterminacy*.

Configurations with one contact point that exhibit inconsistency and indeterminacy have been discussed by Erdmann[13], Lötstedt[28], and Mason and Wang[32]. Lötstedt[30], realizing that indeterminacy and inconsistency present major difficulties for a simulation process, proposes a modification of the Coulomb friction law that eliminates both inconsistency and indeterminacy, and further causes the LCP to always be convex. Baraff[4] shows that determining if a given configuration of objects with dynamic friction is inconsistent is *NP*-complete. Baraff[3] discusses Lötstedt's modification (which avoids the *NP*-complete complexity issue by making inconsistency impossible) and proposes a reinterpretation of the correct behavior of inconsistent configurations. The reinterpretation leads to an algorithm which makes an arbitrary choice among several possible motion behaviors. Empirical evidence in the optimization literature suggests that Baraff's algorithm runs in expected polynomial time.

7.2 Static Friction

Currently, it is unknown how to directly solve for friction forces when contact points with static friction exist. The conditions in section 5.3 do not equate to any known optimization problem, even in the two-dimensional case (when the friction limit is

$$|f_x| \leq \mu f_N$$

as opposed to

$$f_x^2 + f_y^2 \leq (\mu f_N)^2$$

for the three-dimensional problem).

Baraff[4] shows that the solution set of friction forces is not necessarily connected. This suggests that solving for friction forces might require minimization of a nonconvex function. It is known that configurations with static friction can be indeterminate, but no inconsistent configurations have been found. Baraff shows that all one-point configurations with static friction are consistent and speculates that all configurations with static friction are consistent.

Lötstedt[30]'s modification of the Coulomb friction model enables him to solve for static friction forces using quadratic programming. Baraff[4] proposes a solution method that uses an iterative technique combined with quadratic programming to solve for friction forces according to the standard Coulomb friction model. The method is not guaranteed to return a solution, and is incompatible with his method for dealing with inconsistency due to dynamic friction.

References

[1] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Computer Graphics* (*Proc. SIGGRAPH*), volume 23, pages 223–232. ACM, July 1989.

[2] D. Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. In *Computer Graphics* (*Proc. SIGGRAPH*), volume 24, pages 19–28. ACM, August 1990.

[3] D. Baraff. *Dynamic Simulation of Non-penetrating Rigid Bodies*. PhD thesis, Cornell University, May 1992.

[4] D. Baraff. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, (to appear).

[5] R. Barzel and A.H. Barr. A modeling system based on dynamic constraints. In *Computer Graphics* (*Proc. SIGGRAPH*), volume 22, pages 179–188. ACM, July 1988.

[6] M.R. Brach. Rigid body collisions. *Journal of Applied Mechanics*, pages 164–170, 1989.

[7] S.A. Cameron and R.K. Culley. Determining the minimum translational distance between two convex polyhedra. In *International Conference on Robotics and Automation*, pages 591–596. IEEE, 1986.

[8] J. Canny. Collision detection for moving polyhedra. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2), 1986.

[9] R.W. Cottle. On a problem in linear inequalities. *Journal of the London Mathematical Society*, 43:378–384, 1968.

[10] J.F. Cremer. *An Architecture for General Purpose Physical System Simulation*. PhD thesis, Cornell University, April 1989.

[11] P.A. Cundall. Formulation of a three-dimensional distinct element model—Part I. A scheme to represent contacts in a system composed of many polyhedral blocks. *International Journal of Rock Mechanics, Mineral Science and Geomechanics*, 25, 1988.

[12] T. Duff. Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. In *Computer Graphics* (*Proc. SIGGRAPH*), volume 26, pages 131–138. ACM, July 1992.

[13] M.A. Erdmann. On motion planning with uncertainty. Master’s thesis, Massachusetts Institute of Technology, 1984.

[14] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer, 1987.

[15] R. Fletcher. *Practical Methods of Optimization*, volume 2. John Wiley & Sons, 1981.

[16] E.G. Gilbert and C.P. Foo. Computing the distance between smooth objects in three dimensional space. In *International Conference on Robotics and Automation*, pages 158–163. IEEE, 1989.

[17] E.G. Gilbert and S.M. Hong. A new algorithm for detecting the collision of moving objects. In *International Conference on Robotics and Automation*, pages 8–13. IEEE, 1989.

[18] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. A fast procedure for computing the distance between complex objects in three space. In *International Conference on Robotics and Automation*, pages 1883–1889. IEEE, 1987.

[19] H. Goldstein. *Classical Mechanics*. Addison-Wesley, Reading, 1983.

[20] J.P. Gourret, N.M. Thalmann, and D. Thalmann. Simulation of object and human skin deformations in a grasping task. In *Computer Graphics* (*Proc. SIGGRAPH*), volume 23, pages 21–30. ACM, July 1989.

[21] W. Ingleton. A problem in linear inequalities. *Proceedings of the London Mathematical Society*, 16(3):519–536, 1966.

[22] J.B. Keller. Impact with friction. *Journal of Applied Mechanics*, 53(1):1–4, 1986.

[23] W. Kilmister and J.E. Reeve. *Rational Mechanics*. Longman's, 1966.

[24] B. Lafleur, N. Magnenat-Thalmann, and D. Thalmann. Cloth animation with self-collision detection. In T.L. Kunii, editor, *Modeling in Computer Graphics*. Springer-Verlag, 1991.

[25] C. Lanczos. *The Variational Principles of Mechanics*. Dover Publications, Inc., 1970.

[26] M. Lin and J. Canny. Efficient collision detection for animation. In *Third Eurographics Workshop on Animation and Simulation*, September 1992.

[27] M.C. Lin and J.F. Canny. A fast algorithm for incremental distance calculation. In *International Conference on Robotics and Automation*, pages 1008–1014. IEEE, 1991.

[28] P. Lötstedt. Coulomb friction in two-dimensional rigid body systems. *Zeitschrift für Angewandte Mathematik un Mechanik*, 61:605–615, 1981.

[29] P. Lötstedt. Mechanical systems of rigid bodies subject to unilateral constraints. *SIAM Journal of Applied Mathematics*, 42(2):281–296, 1982.

[30] P. Lötstedt. Numerical simulation of time-dependent contact friction problems in rigid body mechanics. *SIAM Journal of Scientific Statistical Computing*, 5(2):370–393, 1984.

[31] M.T. Mason and Y. Wang. Modeling impact dynamics for robotic operations. In *International Conference on Robotics and Automation*, pages 678–685. IEEE, 1987.

[32] M.T. Mason and Y. Wang. On the inconsistency of rigid-body frictional planar mechanics. In *International Conference on Robotics and Automation*, pages 524–528. IEEE, 1988.

[33] M. McKenna and D. Zeltzer. Dynamic simulation of autonomous legged locomotion. In *Computer Graphics* (*Proc. SIGGRAPH*), volume 24, pages 29–38. ACM, August 1990.

[34] N. Megiddo. Linear time algorithms for linear programming in R^3 and related problems. *SIAM Journal of Computing*, 12(4):759–776, 1983.

[35] P.M. Moore. A flexible object animation system. Master's thesis, University of California, Santa Cruz, 1987.

[36] P.M. Moore and J. Wilhelms. Collision detection and reponse for computer animation. In *Computer Graphics* (*Proc. SIGGRAPH*), volume 22, pages 289–298. ACM, August 1988.

[37] F.P. Preparata and M.I. Shamos. *Computational Geometry*. Springer-Verlag, New York, 1985.

[38] H. Rubin and P. Ungar. Motion under a strong constraining force. *Communications on Pure and Applied Mathematics*, 10:65–87, 1957.

[39] J. Snyder. Interval analysis for computer graphics. In *Computer Graphics* (*Proc. SIGGRAPH*), volume 26, pages 121–130. ACM, July 1992.

[40] J. Snyder. Interval methods for multi-point collisions between time-dependent curved surfaces. In *Computer Graphics* (*Proc. SIGGRAPH*), volume 27. ACM, August 1993.

[41] D. Terzopoulos and K. Fleischer. Deformable models. *Visual Computer*, 4:306–331, 1988.

[42] D. Terzopoulos, J.C. Platt, and A.H. Barr. Elastically deformable models. In *Computer Graphics* (*Proc. SIGGRAPH*), volume 21, pages 205–214. ACM, August 1987.

[43] B. Von Herzen, A. Barr, and H. Zatz. Geometric collisions for time-dependent parametric surfaces. In *Computer Graphics* (*Proc. SIGGRAPH*), volume 24, pages 39–48. ACM, August 1990.

[44] Y. Wang and M.T. Mason. Two dimensional rigid body collisions with friction. *Journal of Applied Mechanics*, (to appear).

[45] J. Wilhelms. Toward automatic motion control. *IEEE Computer Graphics and Applications*, 7(4):11–22, 1987.