Dynamic Simulation of Non-Penetrating Rigid Bodies

David Baraff
Ph.D Thesis

92-1275
March 1992

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

DYNAMIC SIMULATION OF NON-PENETRATING RIGID BODIES

A Dissertation
Presented to the Faculty of the Graduate School
of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

by
David Baraff
May 1992

DYNAMIC SIMULATION OF NON-PENETRATING RIGID BODIES

David Baraff, Ph.D.
Cornell University 1992

This thesis examines the problems and difficulties in the forward dynamic simulation of rigid bodies subject to non-penetration constraints. By adopting a simple but well-defined model of rigid body dynamics, we are able to focus on and gain insight into some of the inherent difficulties of rigid body simulation. Additionally, computationally practical solutions to some of the problems encountered in this thesis are presented.

Enforcing non-penetration constraints is essentially a two step process. The first step of the simulation involves the detection of potential contacts between bodies. This thesis presents collision detection algorithms for the dynamic simulation of bodies that are composed of both polyhedra and convex closed curved surfaces. The collision detection algorithms exploit temporal coherence to achieve fast running times and are a practical solution to the problem of collision detection during simulation.

The second step of the simulation involves the computation of the contact forces between bodies that maintain the non-penetration constraint. This thesis considers first the problem of computing contact forces between a pair of bodies that contact at a point without friction. A mathematical formulation for the contact force between the bodies is presented, and then modified to yield a formulation that is

computationally practical for use in a simulator. After considering the dynamics of single point contacts, systems with multiple contacts are considered both in terms of computational complexity measures and practical solution methods. The methods used in this thesis to compute constraint forces are also theoretically and practically compared with a popular method for preventing inter-penetration called the "penalty method".

After considering frictionless systems, this thesis considers systems of bodies that behave according to the classical Coulomb model of friction (which includes both sliding and dry friction). This leads us to consider systems in which there are no solutions to the classical constraint force equations, as well as systems which admit multiple solutions for the constraint force equations and whose subsequent behavior is thus indeterminate. Both computational and practical complexity results for simulating such systems are discussed.

Table of Contents

List of Figures

Chapter 1

Introduction

The research problem motivating this thesis has a long and detailed history, having been a driving force behind the development of both continuous mathematics and the high speed digital computer. The problem in question is the problem of *dynamics*: the study of the motions of bodies. Calculus, for instance, was invented by Newton, as he pondered the motion of bodies under the influence of gravity. The first high speed (at the time) digital computer, the Eniac, was built to solve ballistics problems.

The dynamics problems originally posed by Newton were all of an *unconstrained* nature; for example, the motion of two bodies subject only to gravitational forces between them. In contrast, the dynamics problems considered in this thesis are all of a *constrained* nature. In particular, this thesis will consider the problem of simulating the dynamics of rigid bodies subject to *non-penetration constraints*; that is, constraints that require bodies to behave as if they were solid, and avoid inter-penetration.

In order for a simulation to be computable, it is necessary to make simplifying assumptions about the environment being simulated. In order to make simulation

practical, we usually try to find the simplest assumptions that adequately model a particular environment. As the requirements on our simulation are increased (more complex phenomenon, greater accuracy), we must also increase the complexity of our simulation model. It is naturally assumed, therefore, that realistic simulation is hard mainly because it requires a very complex simulation model. However, as this thesis will show, even very simple simulation models yield complex results and questions.

1.1 Thesis concerns and restrictions

In this thesis, we will study the problems involved in simulating the motions of bodies in accordance with relatively simple physical models of bodies' dynamics. Empirical correctness is not a major concern of this thesis. We do not consider the dynamics models used in this thesis to be empirically correct. That is not to say that our dynamics models are *ad hoc*, or deliberately wrong—a more apt description would be to say that our "simple" dynamics models are incomplete descriptions of more "complicated" dynamics models (which are themselves considered simplifications of even more complex models). Given these simple models, our interest lies in the ramifications of using these models as a basis for simulation. The viewpoint taken by this thesis then is not unlike the viewpoint of rational mechanics[25]. Rational mechanics involves logically deducing the dynamics of systems based on a set of axioms. Rational mechanics gives a "correct result" in as much as the axiom set is valid (with respect to the real world) and complete.

Unlike rational mechanics, this thesis will also be concerned with the practical and computational difficulties that arise in trying to perform simulations based on our simple models. If in fact our simple models are "subsets" of more complicated models, then any findings concerning the difficulties of simulations that use our

simple models should serve as a rough lower bound on the simulation difficulties that will be encountered when more complex models are used. Given that the complex models will yield additional difficulties, it is sensible to first discover the problems stemming from the simple models before moving on to work with the more complex models.

A secondary concern of this thesis is offering practical solutions to the problems encountered in actually using our simplified dynamics model for simulation. For some applications (for example, computer animation, virtual reality, or robot motion planning research) a simulator based on the simple dynamics model considered in this thesis is an end-goal in itself. In these cases, it is important to have practical and efficient simulation algorithms at hand. At present, the vast majority of work in constrained dynamics involving non-penetration constraints uses a simple approximation technique that does not completely enforce the non-penetration constraints (even discounting numerical tolerances). In this thesis, techniques that completely enforce non-penetration constraints (within numerical tolerances) are used. These techniques yield significantly better results and have a firmer theoretical basis than the simple approximation methods, but are considerably more complicated.

The bodies considered in this thesis will be restricted to be perfectly rigid bodies. This implies that bodies propagate forces instantly, without delay, and that complex "micromechanical" processes (such as the elastic response during a collision) are replaced with simple "macromechanical" results. A more important restriction concerns the modeling of contact between bodies. Contact between bodies will always be subject to the restriction that it be modeled as occurring at some finite number of points between bodies. Friction, when considered, will be restricted to the Coulomb model of friction. As we shall see, these few simple laws yield simulation problems of considerable depth and complexity.

1.2 Thesis overview

Chapter 2 presents the prototypical problem that characterizes the problem of dynamic simulation with non-penetration constraints.

Chapter 3 gives an overview of the simulation process, in terms of the major computational steps required.

Chapter 4 summarizes previous work, as classified by the major computational steps of simulation defined in Chapter 3.

Chapter 5 presents collision detection algorithms that are well-suited for the type of environment encountered in dynamics simulation.

Chapter 6 develops the local motion constraint that arises between contacting bodies to prevent inter-penetration.

Chapter 7 analyzes the penalty method, which is currently the most popular method for preventing inter-penetration during simulation.

Chapter 8 formulates equations to solve for constraint forces for frictionless systems.

Chapters 9 and 10 introduce a friction model and discuss the complications that arise from its use. Chapter 11 discusses the computational complexity issues that arise from the introduction of friction.

Chapter 12 proposes a physical interpretation of the computational difficulties arising from friction. Chapters 13 and 14 discuss principles and computational methods for simulations with friction.

Chapter 15 concludes this thesis by discussing future directions for simulations with non-penetration constraints.

Chapter 2

A Simple Dynamics Problem with Non-penetration Constraints

In considering the steps needed to solve dynamics problems with non-penetration constraints, it is helpful to have a dynamics problem with non-penetration constraints firmly in mind. The simple dynamics problem presented in this chapter illustrates the important issues of dynamic simulation involving non-penetration constraints. The problem is to simulate the dynamics of a point mass particle in three-space ($\mathbf{R}^3$). The position x of the particle at time t in $\mathbf{R}^3$ is written $x(t)$. 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 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

$$\ddot{x}(t) = F(t) \tag{2–1}$$

describes the particle's motion. In numerical simulation though, this second order differential equation is converted to a coupled first order differential equation by writing

$$\begin{aligned} \dot{x}(t) &= v(t) \\ \dot{v}(t) &= F(t) \end{aligned}$$

or in vector notation,

$$\frac{\mathrm{d}}{\mathrm{dt}} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ F(t) \end{pmatrix}. \tag{2–2}$$

The variables x and v are called *state variables*, and the collection of x and v is called the *state* of the system. The variable x is a *spatial* variable while v is a *velocity* variable.

2.2 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

$$C(x(t)) = 0. \tag{2–3}$$

Since this constraint can be written as an equality, this is an example of an *equality constrained* dynamics problem (Figure 2.1).

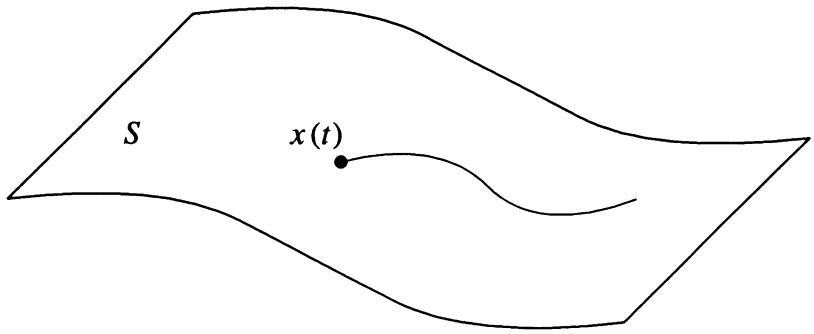


Figure 2.1: An equality constraint between a particle and a surface. The particle, constrained by $C(x(t)) = 0$, must always lie on the surface S.

2.3 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 2.2). If points p lying above S satisfy $C(p) > 0$, then the constraint that the particle lie on or above S can be written

$$C(x(t)) \geq 0. \tag{2–4}$$

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 equality constrained dynamics problems, and is thus the most difficult problem of the three. Clearly, 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

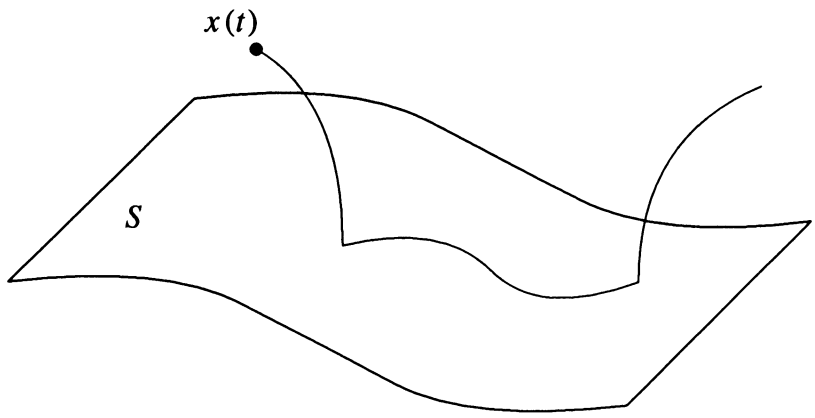


Figure 2.2: An inequality constraint between a particle and a surface. The particle, constrained by $C(x(t)) \geq 0$, can move either on or "above" the surface S.

the inequality constrained problem—the equality constraint $C(x(t)) = 0$ can be replaced by the constraints $C(x(t)) \geq 0$ and $-C(x(t)) \geq 0$.

2.4 The penalty method

Constrained dynamics problems can be very difficult to solve. A very common method for solving a problem with constraints is the *penalty method.* 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

$$\text{minimize } f(z) \quad \text{such that} \quad g(z) = 0 \tag{2–5}$$

can be rewritten as an unconstrained problem

$$\text{minimize } f(z) + kg(z)^2 \quad \text{as } k \to \infty. \tag{2–6}$$

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 (2–6). In the limit, as k goes to infinity, the solution of Equation (2–6) must satisfy $g(z) = 0$ while minimizing $f(z)$. In practice, z is obtained by solving Equation (2–6) 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 (2–6) can become very poorly conditioned and difficult to solve[14]. 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 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]

$$-k\left(x(t) - p(x(t))\right).$$

[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.

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

$$\frac{\mathrm{d}}{\mathrm{dt}} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ -k\,(x(t) - p(x(t))) + F(t) \end{pmatrix}. \tag{2–7}$$

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 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.

For neither the equality constrained problem nor the inequality constrained problem is it obvious that the penalty method must give a correct solution, given a sufficiently large value of k. It is clear however that except for a limited class of simulations, the penalty method is an unsuitable method, because of the ill-conditioning of the differential equations of motion. Further analysis of the penalty method will be deferred to Chapter 7.

2.5 Exact solution methods

The penalty method of the previous section involves a formulation of the problem in which constraints are only approximately satisfied. A completely different viewpoint is taken in this section, in that constraints are required to be satisfied "exactly" (within numerical tolerances).

2.5.1 The equality constrained problem

The equality constrained problem of the particle that must remain in contact with the surface S can be solved in two ways. The first solution method involves converting the constrained problem to an unconstrained problem—essentially the method of Lagrangian dynamics. This can be done if a parametric representation for S can be found.

Otherwise, the problem can be solved by introducing a *constraint force* $F_c(t)$ into the problem. The role of the constraint force is to act on the particle so that it remains in contact with the surface. Additionally, the constraint force must be *workless*; that is, the work done on the particle by the constraint force must always be zero. For the particle/surface example, this requirement forces $F_c(t)$ to act on the particle in a direction normal to the surface S at $x(t)$. Thus, the constraint force's responsibility is to adjust the acceleration of the particle normal to the surface, so that the particle will remain on the surface. The normal to S at the point $x(t)$ is

the vector $\nabla C(x(t))$ where[2]

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

Then $F_c(t)$ can be written $F_c(t) = \lambda(t)\nabla C(x(t))$ where $\lambda(t)$ is an unknown scalar. The scalar $\lambda(t)$ can be either positive or negative. In Figure 2.3a, the constraint force $F_c(t)$ acts to oppose a large external force $F(t)$. The constraint force acts in the same direction as $\nabla C(x(t))$; in this case, $\lambda(t) > 0$. In Figure 2.3b, the external force $F(t)$ now pushes the particle away from the surface in the $\nabla C(x(t))$ direction. In this case, the constraint force must act opposite $\nabla C(x(t))$; hence $\lambda(t) < 0$.

The equation of motion for the particle then becomes

$$\ddot{x}(t) = F(t) + F_c(t) = F(t) + \lambda(t)\nabla C(x(t)). \tag{2–8}$$

How is $\lambda(t)$ computed?

In equality constrained dynamics problems, it is assumed that the constraint is initially met; that is, at time zero,

$$C(x(0)) = 0. \tag{2–9}$$

Additionally,

$$\dot{C}(x(0)) = \frac{\mathrm{d}}{\mathrm{dt}} C(x(0)) = \nabla C(x(0)) \cdot \dot{x}(0) \tag{2–10}$$

must also be zero as well. If it is not, then the particle has a velocity that is not completely tangential to S at the point $x(0)$; this situation will require an *impulsive* constraint force, and will be dealt with shortly. If however both $C(x(t))$

[2] It is assumed that the function C modeling S is chosen such that $\nabla C(p)$ does not vanish on S.

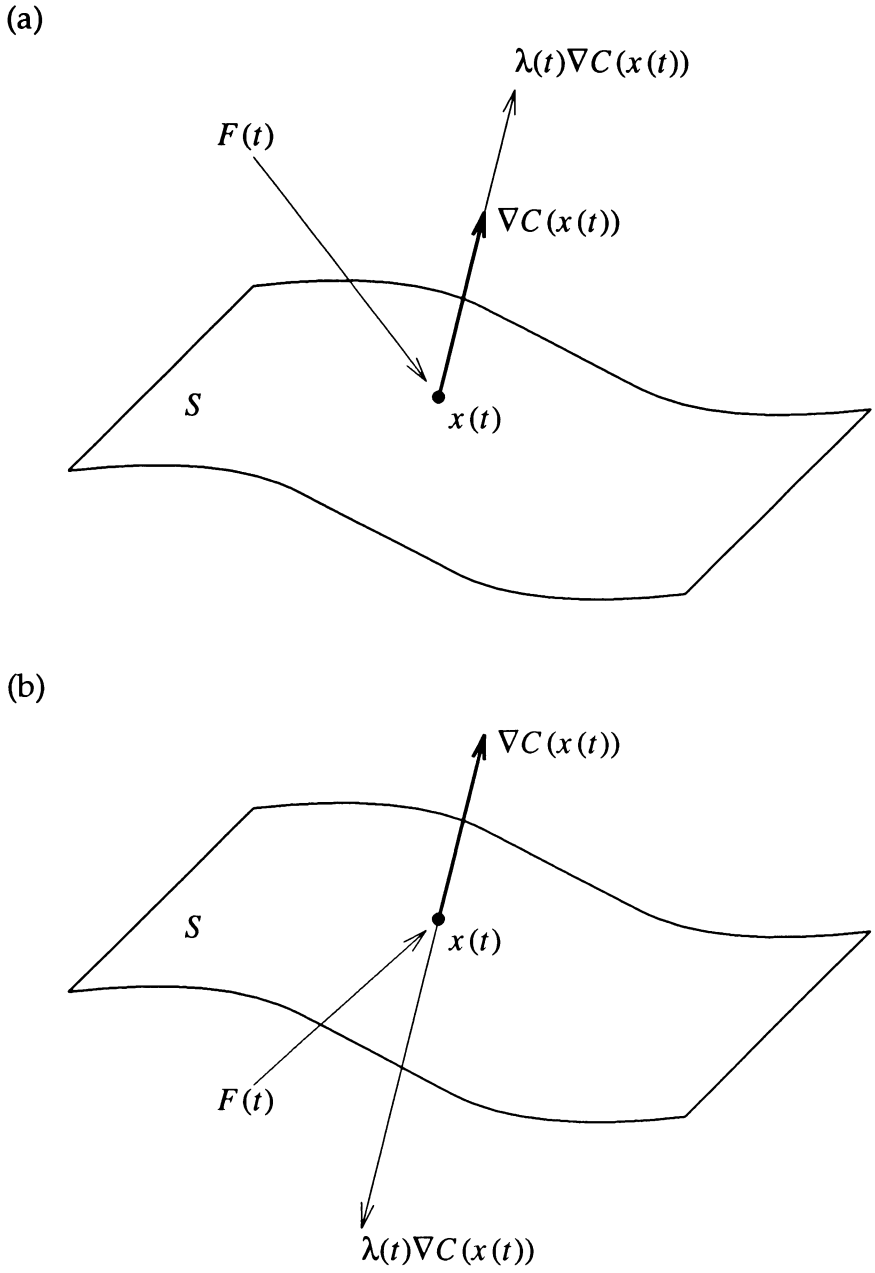


Figure 2.3: Constraint forces parallel to the surface normal $\nabla C(x(t))$ for an equality constrained problem. (a) To maintain the particle/surface constraint, $\lambda(t) > 0$ so that the constraint force $F_c(t) = \lambda(t)\nabla C(x(t))$ pushes the particle in the positive $\nabla C(x(t))$ direction. (b) The constraint force $F_c(t)$ must be applied opposite to $\nabla C(x(t))$; thus $\lambda(t) < 0$.

and $\dot{C}(x(t))$ are zero for $t = 0$, then both quantities will remain zero as long as $\ddot{C}(x(t)) = \frac{d^2}{dt^2}C(x(t))$ is zero. Intuitively, $C(x(t))$ is a measure of the particle's displacement normal to S, and $\dot{C}(x(t))$ is a measure of the particle's velocity normal to S. Thus, $\ddot{C}(x(t))$ is a measure of the particle's acceleration normal to S. The constraint force $\lambda(t)\nabla C(x(t))$ is chosen so that $\ddot{C}(x(t))$ is always zero.

Starting from $\dot{C}(x(t)) = \nabla C(x(t)) \cdot \dot{x}(t)$ and taking its derivative,

$$\begin{aligned}
\ddot{C}(x(t)) &= \frac{d}{dt}(\nabla C(x(t)) \cdot \dot{x}(t)) \\
&= \left(\frac{d}{dt}\nabla C(x(t))\right) \cdot \dot{x}(t) + \nabla C(x(t)) \cdot \left(\frac{d}{dt}\dot{x}(t)\right) \qquad (2\text{–}11) \\
&= \dot{x}(t) \cdot \left(\nabla^2 C(x(t))\dot{x}(t)\right) + \nabla C(x(t)) \cdot \ddot{x}(t)
\end{aligned}$$

where the term $\nabla^2 C(x(t))$ is the matrix of second partial derivatives of C (that is, the curvature of the surface), evaluated at the point $x(t)$. Using Equation (2–8),

$$\begin{aligned}
\ddot{C}(x(t)) &= \left(\dot{x}(t) \cdot \left(\nabla^2 C(x(t))\dot{x}(t)\right)\right) + \left(\nabla C(x(t)) \cdot \ddot{x}(t)\right) \\
&= \left(\dot{x}(t) \cdot \left(\nabla^2 C(x(t))\dot{x}(t)\right)\right) + \left(\nabla C(x(t)) \cdot \left(F(t) + \lambda(t)\nabla C(x(t))\right)\right) \\
&= \dot{x}(t) \cdot \left(\nabla^2 C(x(t))\dot{x}(t)\right) + \nabla C(x(t)) \cdot F(t) \qquad (2\text{–}12) \\
&\quad + \lambda(t)\nabla C(x(t)) \cdot \nabla C(x(t)).
\end{aligned}$$

Setting $\ddot{C}(x(t))$ to zero and solving for $\lambda(t)$,

$$\lambda(t) = -\frac{\dot{x}(t) \cdot \left(\nabla^2 C(x(t))\dot{x}(t)\right) + \nabla C(x(t)) \cdot F(t)}{\nabla C(x(t)) \cdot \nabla C(x(t))}. \qquad (2\text{–}13)$$

Having found $\lambda(t)$, Equation (2–8) can be solved numerically, and the motion of the particle will satisfy the equality constraint.

2.5.2 The inequality constrained problem

The solution to the inequality case is only slightly more complicated for this simple example. Suppose that initially the particle lies above the surface S, and does not come into contact with the surface until time t_1. During the time interval zero to t_1 the constraint is said to be *inactive* and the motion of the particle satisfies

$$\ddot{x}(t) = F(t).$$

The first problem then is to determine t_1, that is, to determine when the particle contacts the surface. This is the problem of *collision detection.*

After the particle contacts the surface at time t_1, the constraint $C(x(t)) \geq 0$ becomes active. To prevent the particle from moving below the surface, a constraint force must act. In what follows, it is assumed that $\nabla C(x(t))$ points "upwards" (towards the region of space that the particle is permitted to move in) as in Figure 2.4. As in the equality constrained problem, the constraint force is assumed to do no work on the particle; so again, the constraint force can be written as $F_c(t) = \lambda(t)\nabla C(x(t))$. Unlike the equality constrained problem, the constraint force must be directed away from the surface, in the $\nabla C(x(t))$ direction. This means that $\lambda(t)$ must satisfy $\lambda(t) \geq 0$. In Figure 2.4a, where the external force $F(t)$ pushes the particle downwards, the constraint force $\lambda\nabla C(x(t))$ is exactly the same as in Figure 2.3a. However, when the external force $F(t)$ of Figure 2.4b acts on the particle, *no* constraint force acts (that is, $\lambda(t)$ is zero), and the particle breaks contact with S, accelerating upwards. If $\lambda(t)$ was allowed to be negative, as in Figure 2.3b, the particle, once in contact with the surface, could be held to the surface for all time by computing a value for $\lambda(t)$ as in the equality constrained problem.

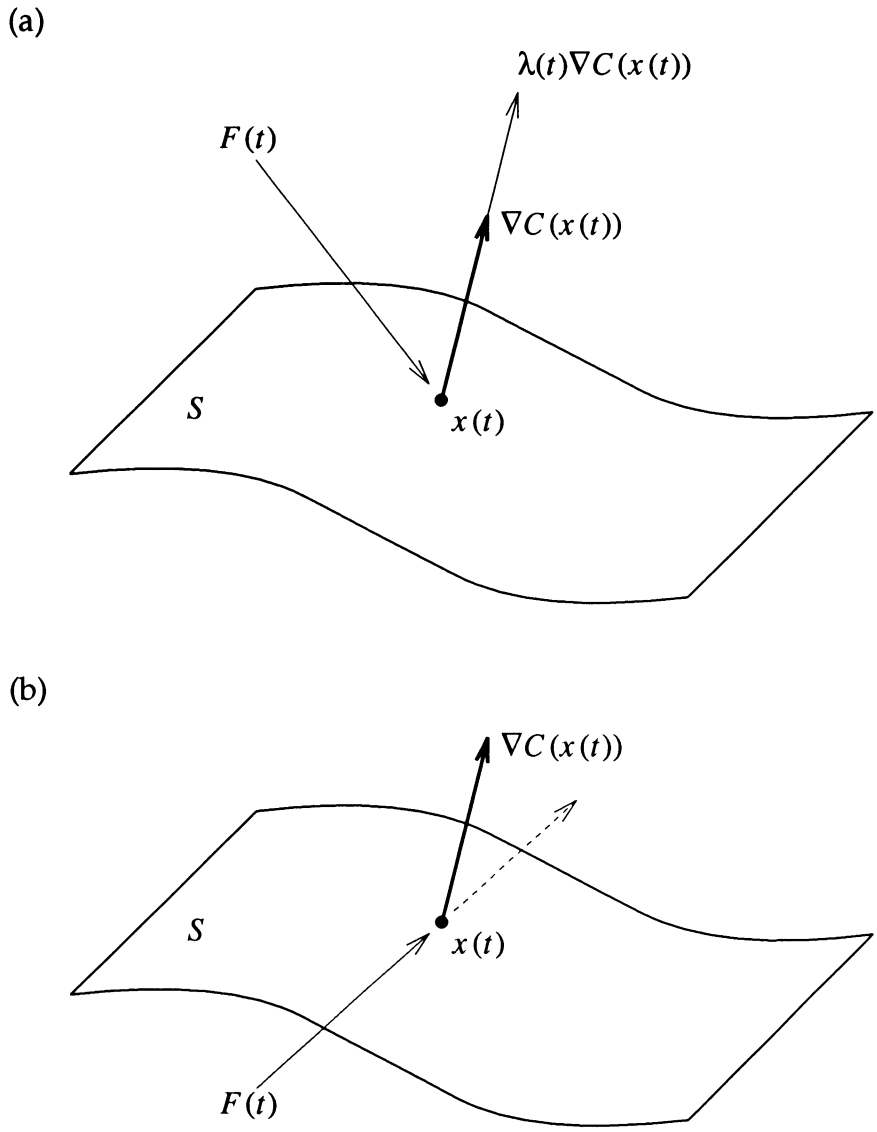


Figure 2.4: Constraint forces parallel to the surface normal $\nabla C(x(t))$ for an inequality constrained problem. (a) A constraint force $F_c(t) = \lambda(t)\nabla C(x(t))$ with $\lambda(t) > 0$ acts to prevent the particle from accelerating downwards. (b) The external force $F(t)$ accelerates the particle upwards, and contact is broken. No constraint force acts on the particle.

First derivative considerations

If at time t_1 (when the particle first contacts the surface)

$$\dot{C}(x(t_1)) = \nabla C(x(t_1)) \cdot \dot{x}(t_1) = \nabla C(x(t_1)) \cdot v(t_1) < 0, \tag{2–14}$$

then the particle is said to have *collided* with the surface. To prevent the particle from moving below the surface, the velocity $v(t)$ must undergo a discontinuity at time t_1; that is the only means of completely enforcing the constraint. The discontinuity is said to be the result of an *impulsive* constraint force that acts normal to the surface at the point $x(t_1)$. An impulsive force has the dimensions of force times time, or equivalently mass times velocity, and produces a discontinuity in the velocity. Impulsive forces are the limiting effect of large forces acting over small time intervals, as the force strength goes to infinity and the time interval goes to zero. Thus, the impulsive constraint force will also be normal to the surface, and can be expressed as $j\nabla C(x(t_1))$ where j is a positive scalar.

Because the particle and the surface are regarded as rigid bodies, the strength of the impulsive force is found by using the empirical relation

$$v_n^+ = -\epsilon v_n^-. \tag{2–15}$$

In this relation, v_n^+ is the velocity of the particle normal to the surface after the impulsive constraint force is applied, and v_n^- is the normal velocity prior to the collision. The scalar $0 \leq \epsilon \leq 1$ is called the *coefficient of restitution* and depends on the material properties of the particle and the surface.

Since the particle has unit mass, its velocity $v^+(t_1)$ after application of the impulse $j\nabla C(x(t_1))$ is

$$v^+(t_1) = v(t_1) + j\nabla C(x(t_1)). \tag{2–16}$$

(This equation can be formally derived by defining $v^+(t_1) = \lim_{\Delta t \to 0} v(t_1 + \Delta t)$ when a force $\lambda \nabla C(x(t))$ acts on the particle from time t_1 to time $t_1 + \Delta t$, with λ

a scalar, and $\lim_{\Delta t \to 0} \lambda \Delta t = j$.) The normal velocities v_n^- and v_n^+ are

$$v_n^- = \nabla C(x(t_1)) \cdot v(t_1) \qquad \text{and} \qquad v_n^+ = \nabla C(x(t_1)) \cdot v^+(t_1). \tag{2–17}$$

Substituting into Equation (2–15),

$$\nabla C(x(t_1)) \cdot v^+(t_1) = -\epsilon \nabla C(x(t_1)) \cdot v(t_1) \tag{2–18}$$

or

$$\nabla C(x(t_1)) \cdot (v(t_1) + j \nabla C(x(t_1))) = -\epsilon \nabla C(x(t_1)) \cdot v(t_1). \tag{2–19}$$

Solving for j,

$$\begin{aligned} j &= \frac{-\epsilon \nabla C(x(t_1)) \cdot v(t_1) - \nabla C(x(t_1)) \cdot v(t_1)}{\nabla C(x(t_1)) \cdot \nabla C(x(t_1))} \\ &= -\frac{(\epsilon + 1) \nabla C(x(t_1)) \cdot v(t_1)}{\nabla C(x(t_1)) \cdot \nabla C(x(t_1))} \\ &= -\frac{(\epsilon + 1) \dot{C}(x(t_1))}{\nabla C(x(t_1)) \cdot \nabla C(x(t_1))}. \end{aligned} \tag{2–20}$$

Note that since $\dot{C}(x(t_1)) < 0$ and $\nabla C(x(t_1)) \cdot \nabla C(x(t_1)) > 0$, then $j > 0$. This indicates that the constraint impulse acts upwards on the particle, as expected.

If $\epsilon > 0$, then the particle will have a velocity along $\nabla C(x(t_1))$ after the collision; the particle has bounced off the surface, and is now moving away from the surface. The constraint changes back to being inactive, and the motion of the particle is considered unconstrained until such time as it contacts the surface again. Note that this entire process happens over a zero length time interval; thus, the motion of the particle is piecewise continuous.

Otherwise, if $\epsilon = 0$, then the particle has no velocity normal to the surface after the collision. In this case, the constraint stays active, and a regular, non-impulsive constraint force can be considered.

Second derivative considerations

The case when $\dot{C}(x(t)) < 0$ has just been discussed. If $\dot{C}(x(t)) > 0$, the particle has a velocity directed away from the surface, and the constraint immediately becomes inactive after time t. Thus, in considering the constraint forces that prevent the particle from moving below the surface at time t, we may assume

$$C(x(t)) = \dot{C}(x(t)) = 0. \tag{2–21}$$

In the equality constrained problem, $C(x(t)) = 0$ was maintained by choosing $\lambda(t)$ such that $\ddot{C}(x(t)) = 0$. For the inequality constrained problem, $C(x(t)) \geq 0$ is maintained by choosing $\lambda(t)$ so that $\ddot{C}(x(t)) \geq 0$, subject to the condition $\lambda(t) \geq 0$. Substituting from Equation (2–12) into $\ddot{C}(x(t)) \geq 0$, we obtain

$$\begin{aligned} \ddot{C}(x(t)) &= \dot{x}(t) \cdot \left(\nabla^2 C(x(t))\dot{x}(t)\right) + \nabla C(x(t)) \cdot F(t) \\ &\quad + \lambda(t)\nabla C(x(t)) \cdot \nabla C(x(t)) \geq 0. \end{aligned} \tag{2–22}$$

Since $\nabla C(x(t)) \cdot \nabla C(x(t))$ is always positive, $\ddot{C}(x(t)) \geq 0$ is equivalent to

$$\lambda(t) \geq -\frac{\dot{x}(t) \cdot \left(\nabla^2 C(x(t))\dot{x}(t)\right) + \nabla C(x(t)) \cdot F(t)}{\nabla C(x(t)) \cdot \nabla C(x(t))}. \tag{2–23}$$

Equation (2–23) places only a lower bound on $\lambda(t)$, and does not specify $\lambda(t)$ uniquely. However, a reasonable restriction to impose on $\lambda(t)$ is that it be zero whenever $\ddot{C}(x(t)) > 0$. This is sensible because if $\ddot{C}(x(t)) > 0$ then $C(x(t))$ is an increasing function and the particle will move away from the surface immediately after time zero. When this happens, the constraint becomes inactive and the constraint force is non-existent; or equivalently, $\lambda(t)$ is zero. This restriction can be written very simply as

$$\lambda(t)\ddot{C}(x(t)) = 0 \tag{2–24}$$

which, along with $\ddot{C}(x(t)) \geq 0$ and $\lambda(t) \geq 0$, asserts that either $\ddot{C}(x(t)) > 0$ and $\lambda(t) = 0$, or $\ddot{C}(x(t)) = 0$.

With this added restriction, $\lambda(t)$ is computed very easily. If $\lambda(t)$'s lower bound (Equation (2–23)) is positive, then setting $\lambda(t)$ equal to its lower bound results in $\lambda(t) > 0$ and $\ddot{C}(x(t)) = 0$; otherwise, $\lambda(t)$'s lower bound is negative. Since the denominator of Equation (2–23) is always positive, it must be that

$$\dot{x}(t) \cdot (\nabla^2 C(x(t))\dot{x}(t)) + \nabla C(x(t)) \cdot F(t) > 0. \tag{2–25}$$

Setting $\lambda(t) = 0$ satisfies $\lambda(t) \geq 0$ and $\lambda(t)\ddot{C}(x(t)) = 0$. Using Equation (2–22), $\ddot{C}(x(t))$ satisfies

$$\begin{aligned}
\ddot{C}(x(t)) &= \dot{x}(t) \cdot \left(\nabla^2 C(x(t))\dot{x}(t)\right) + \nabla C(x(t)) \cdot \ddot{x}(t) \\
&= \dot{x}(t) \cdot \left(\nabla^2 C(x(t))\dot{x}(t)\right) + \nabla C(x(t)) \cdot \left(F(t) + \lambda(t)\nabla C(x(t))\right) \\
&= \dot{x}(t) \cdot \left(\nabla^2 C(x(t))\dot{x}(t)\right) + \nabla C(x(t)) \cdot F(t) > 0.
\end{aligned}$$

Note that during the period of time that $\ddot{C}(x(t)) = 0$, the value of $\lambda(t)$ is the same as if the inequality constraint was in fact an equality constraint; during this time, the particle's behavior is the same as if the particle had been constrained to remain on the surface. However, when in the equality constrained problem $\lambda(t)$ would have decreased below zero, in the inequality constrained problem $\lambda(t)$ stops decreasing at zero and the particle breaks contact with the surface.

Chapter 3

Simulation Overview

Having described a simple, yet complete, problem with non-penetration constraints, the overall structure of a simulation algorithm for the problem is presented, along with the extensions needed to simulate an arbitrary system of rigid bodies.

Overall, the simulation process can be described as nothing more than the numerical solution to a first order ordinary differential equation (ODE). A first order ODE has the form $\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{Y}(t) = f(t, \mathbf{Y}(t))$. The ODE for a system of arbitrarily many rigid bodies is given in Section 3.1. The vector $\mathbf{Y}(t)$ is called the *state* of the system, and $\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{Y}(t) = f(t, \mathbf{Y}(t))$ is called the *state derivative.* Numerical methods for solving such equations are well known. The simulator discussed in this thesis uses an *explicit* solution method (as opposed to an *implicit* solution method) for solving the ODE. (Numerical solution techniques for ordinary differential equations are discussed by Shampine and Gordon[47].) An explicit solution method requires a value for $\mathbf{Y}(0)$, which is called the *initial state* of the system, and a time t_{end} for which the solution method is to compute $\mathbf{Y}(t_{end})$. In computing $\mathbf{Y}(t_{end})$, the solution method will need to be given values for the state derivative at various times from zero to t_{end}. The solution method is completely responsible for choosing values

of t at which to compute the state derivative, and for computing $\mathbf{Y}(t_{end})$ based on the computed state derivatives.

Thus, from a mathematical perspective, the work of a simulation lies "merely" in the task of periodically computing the state derivative. For each value of t for which the state derivative $\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{Y}(t)$ is to be computed, the sequence of steps in Figure 3.1 is performed. The meanings of the steps in Figure 3.1 are explained below. The computation of the state derivative will sometimes be referred to as a *time step*; successive time steps refers to successive computations of the state derivative for different values of t. The *interval* between two time steps is the difference in the value of t between the time steps.

3.1 Updating the current state

When the numerical ODE solver requests a value for $\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{Y}(t)$ at some time t_0, it begins by divulging the *current state* of the system; that is, its estimation of the value of $\mathbf{Y}(t_0)$.

For the particle/surface problem, the current state is simply the vector containing $x(t_0)$ and $v(t_0)$. For a collection of n rigid bodies, the current state is somewhat more complicated, but conceptually the same. The ODE for a collection of n rigid

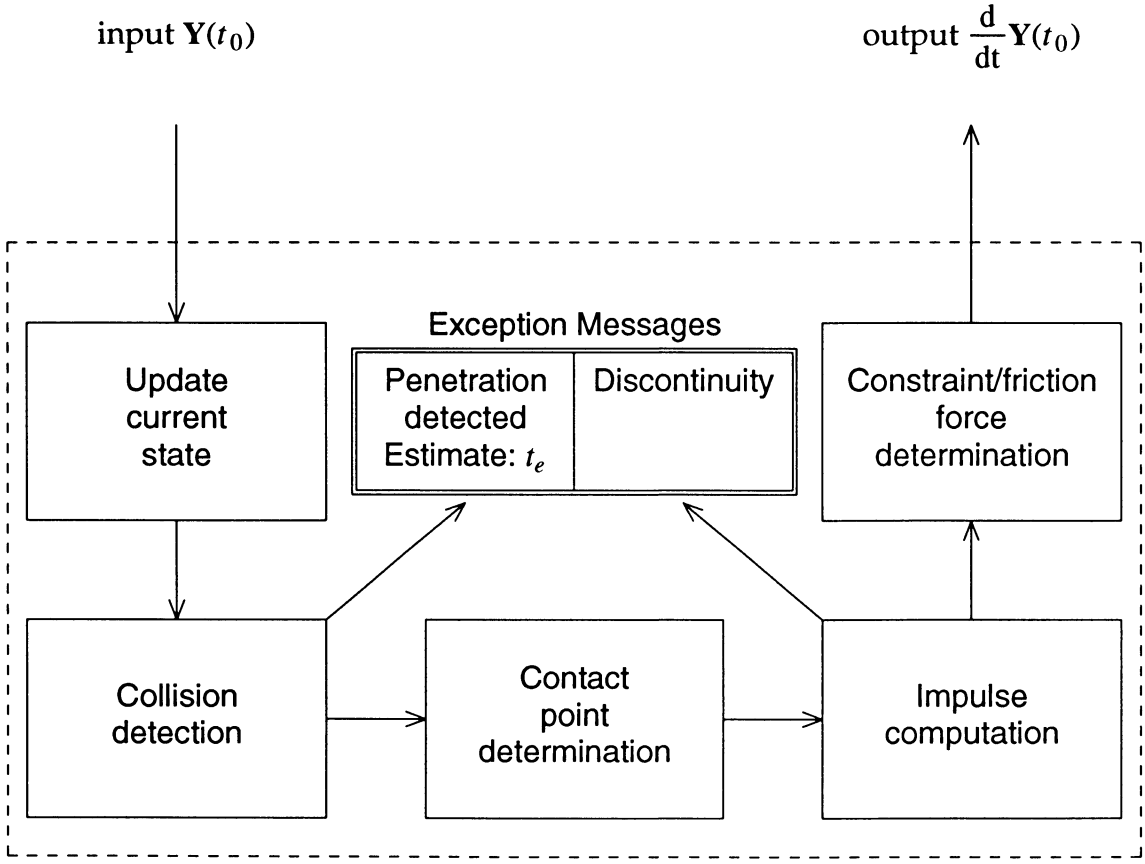


Figure 3.1: The computational steps required to compute $\frac{d}{dt}\mathbf{Y}(t)$. During the computation, various "exceptions" can be communicated to the ODE solver.

bodies has the form

$$\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{Y}(t) = \frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix} c_1(t) \\ q_1(t) \\ v_1(t) \\ \omega_1(t) \\ \vdots \\ c_n(t) \\ q_n(t) \\ v_n(t) \\ \omega_n(t) \end{pmatrix} = \begin{pmatrix} v_1(t) \\ \frac{1}{2}\omega_1(t)q_1(t) \\ F_1(t)/m_1 \\ I_1^{-1}\tau_1(t) \\ \vdots \\ v_n(t) \\ \frac{1}{2}\omega_n(t)q_n(t) \\ F_n(t)/m_n \\ I_n^{-1}\tau_n(t) \end{pmatrix} \tag{3–1}$$

where $c_i(t)$ is the position of the ith body's center of mass, $q_i(t)$ is the unit quaternion representation of the ith body's orientation[21,48], $v_i(t)$ and $\omega_i(t)$ are the linear and angular velocities of the ith body, $F_i(t)$ and $\tau_i(t)$ are the total force and torque acting on the ith body, and m_i and $I_i(t)$ are the ith body's mass and inertia tensor. The term $\omega_i(t)q_i(t)$ denotes quaternion multiplication between $0 + \omega_{x_i}(t)\mathbf{i} + \omega_{y_i}(t)\mathbf{j} + \omega_{z_i}(t)\mathbf{k}$ and $q_i(t)$.

The values of all the position and velocity variables of the bodies are taken from $\mathbf{Y}(t_0)$ during the computation the derivative state at time t_0.

3.2 Collision detection

In the particle/surface problem, the motion of the particle is treated as unconstrained whenever the particle lies above the surface. Suppose that at some time t_0 the particle is above, but near the surface, and has a velocity downwards (towards the surface). Based on the history of the particle's velocity and acceleration up to time t_0, the ODE solver computes a position for the particle at some time $t_0 + \Delta t$. However, because the ODE solver has no knowledge of the non-penetration

constraint between the particle and the surface, it does not know that the particle cannot continue to move downwards indefinitely. The ODE computes a state $\mathbf{Y}(t_0 + \Delta t)$ for the particle such that the particle lies below the surface; clearly, this is incorrect. The ODE solver then requests the value of $f(t_0 + \Delta t, \mathbf{Y}(t_0 + \Delta t))$, unaware of the fact that $\mathbf{Y}(t_0 + \Delta t)$ is an *illegal* (and hence incorrect) value for the state at time $t_0 + \Delta t$.

It is the collision detection step's responsibility to examine the current state (or rather the *proposed* current state). If the state is illegal, that is, if the non-penetration constraint has been violated, then at some time t_c lying between t_0 and $t_0 + \Delta t$, a collision must have occurred. The collision detector makes some estimate t_e of t_c, informs the ODE solver that the proposed state $\mathbf{Y}(t_0 + \Delta t)$ is illegal, and gives the estimated value t_e. The value of $f(t_0 + \Delta t, \mathbf{Y}(t_0 + \Delta t))$ is undefined since the proposed state $\mathbf{Y}(t_0 + \Delta t)$ is illegal, and computation of $\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{Y}(t_0 + \Delta t)$ is aborted. The ODE solver proceeds forward from time t_0 and computes a proposed state for $\mathbf{Y}(t_0 + t_e)$.

If $t_e < t_c$, the proposed state $\mathbf{Y}(t_0 + t_e)$ is legal and it is now known that $t_e < t_c < t_0 + \Delta t$. Based on information from the state $\mathbf{Y}(t_e)$, it may be possible to produce a better estimate for t_c, which the solver can aim for. Failing that, the solver can at least bisect the time interval from t_e to $t_0 + \Delta t$ by estimating t_c to be $\frac{1}{2}(t_e + t_0 + \Delta t)$.

Conversely, if $t_c < t_e$, then the proposed state $\mathbf{Y}(t_e)$ is illegal, and the solver knows that $t_0 < t_c < t_e$. Again, either a better prediction for t_c can be made based on $\mathbf{Y}(t_e)$, or the time interval can be bisected. Eventually, the solver finds t_c to within a suitable numerical precision; that is, a state is reached for which the non-penetration constraint is satisfied within numerical tolerances and the velocities of bodies are such that a collision is occurring.

One potential difficulty lies in the fact that if two successive states $\mathbf{Y}(t_0)$ and

$\mathbf{Y}(t_0 + \Delta t)$ are determined to be legal, it is still possible that a collision might have occurred somewhere between these two states. This could happen if one body passed completely through another between time t_0 and time $t_0 + \Delta t$. Consider for example simulating the motion of a bullet shot through a thin pane of glass. In this case, the odds are against the solver requesting $\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{Y}(t)$ at a value of t for which the bullet inter-penetrates the glass. However, for reasonably complicated simulations, the numerical considerations in solving the differential equations of motion limit the interval between time steps and the problem arises infrequently. For this reason, we choose not to treat the case where a body passes completely through another body in one time step. Possible solutions to this problem are using some type of 4D space-time volume algorithms to detect collisions, or simply limiting the maximum interval between time steps.

3.3 Contact point determination

Once a legal state $\mathbf{Y}(t_0)$ has been achieved, the constraint forces at time t_0 must be computed. The first step in that computation is determining *where* the constraint forces can arise. For the particle/surface problem, this determination is simple; if the particle lies on the surface (within some numerical tolerance), a *contact point* is said to exist between the particle and the surface. The location of this contact point is simply the position of the particle. If a constraint impulse or a constraint force is needed, it will act between the particle and the surface at the contact point. For systems of rigid bodies, there may be arbitrarily many points of contact between the bodies, and a constraint force may act at each point. Thus, it is necessary to first determine all the contact points between the bodies before giving further consideration to the constraint forces.

In this thesis, contact between bodies has been restricted to the case when it

can be modeled as occurring at some finite number of points between bodies. This restriction does not apply to contact between polyhedral bodies. For polyhedral bodies, contact areas are always polygonal (considering single points and line segments as degenerate polygons). By imposing a non-penetration constraint at each vertex of the polygonal contact area, inter-penetration is prevented over the entire polygonal contact area. Additionally, any distribution of normal forces over the contact area is equivalent to some assignment of normal forces acting only at the vertices of the contact region. Thus, the definition of a contact point for the case of polyhedral contact is limited to the vertices of the contact region.

Although the collision detection and contact point determination steps are listed separately in Figure 3.1, the computation of each overlaps. As bodies are checked to see if non-penetration constraints have been violated, contact points are determined at the same time. Both steps can share a significant amount of computation, and performing them separately would be inefficient. When a proposed state is found to be illegal, the work done so far in determining contact points is wasted; but it would be much more wasteful to implement the two steps completely separately.

3.4 Impulse computation

Once the contact points have been determined, the relative velocity at each contact point can be computed. If the velocity at a contact point is such that the contact is breaking, no non-penetration constraint need be formulated at the contact point; equivalently, the non-penetration constraint at the contact point is inactive.

If the velocity at a contact point is such that a collision is taking place, an impulsive force will arise. If it is assumed that no two collisions occur at exactly the same time, the calculation of the impulse, for frictionless surfaces, is exactly as described in Section 2.5. Methods for dealing with simultaneous impulses, and

impulses involving friction are summarized in Chapter 4.

Since impulsive forces cause discontinuities in the velocity variables $v_i(t)$ and $\omega_i(t)$ of the rigid bodies, special care must be taken when impulses are applied. Numerical ODE solvers assume that the function $\mathbf{Y}(t)$ is smooth. Since $\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{Y}(t) = f(t, \mathbf{Y}(t))$ is only piecewise continuous, it must be treated as such. Accordingly, when an impulse is applied at time t_0, new values for the affected $v_i(t)$ and $\omega_i(t)$ must be computed, and the ODE solver must be informed that a discontinuity has been encountered. The ODE solver is then restarted with a new state $Y^+(t_0)$, where $Y^+(t_0)$ contains the new values of the affected $v_i(t)$ and $\omega_i(t)$. Note that if the ODE solver begins by requesting a value for $\frac{\mathrm{d}}{\mathrm{dt}}Y^+(t_0)$, the collision detection and contact point determination information computed for $\mathbf{Y}(t_0)$ can be used in computing $\frac{\mathrm{d}}{\mathrm{dt}}Y^+(t_0)$, since $\mathbf{Y}(t_0)$ and $Y^+(t_0)$ represent the same geometric positionings of bodies.

3.5 Constraint/friction force determination

This step is the easiest to describe, but the most difficult to perform. At this point in the computation of $\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{Y}(t_0)$, all contact points with active non-penetration constraints have been identified. Since any constraints requiring impulsive constraint forces have already been taken care of, all that remains is the formulation and solution of a system of equations for the unknown constraint forces. While constraint forces always act normally between contacting bodies, contacts involving friction give rise to forces that act tangentially between bodies, to oppose slipping. For contacts with friction, a system of equations for both the unknown constraint forces and tangential friction forces needs to be formulated and solved.

Once the constraint forces and the friction forces have been computed, the right-hand-side of Equation (3–1) is trivially computed, and the computation of $\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{Y}(t_0)$ is complete. The ODE solver computes a new (proposed) state $\mathbf{Y}(t_0 + \Delta t)$, and computation of $\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{Y}(t_0 + \Delta t)$ begins.

Chapter 4

Summary of Previous Work

In this chapter, previous work relating to the various simulation steps outlined in Chapter 3 is summarized.

4.1 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 inter-penetrate and contact each other?"

The first approach, which could be called the "continuum view", presupposes

a specified motion of bodies over some time interval. Examples include Canny[6], 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[18] relax the problem of collision detection to include arbitrary trajectories of rigid convex polyhedra. Since no closed-form solution exists for 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[19]. Von Herzen, Barr, and Zatz[53] 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). Unfortunately, algorithms of the above nature cannot be used in the simulation problem as described in Chapter 3. All of the above algorithms presuppose a specified motion for the bodies over time; however, such a path is exactly what the simulator is trying to compute, which means we cannot make use of these sorts of collision detection algorithms.

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[5] for computing the distance between two convex polyhedra with n vertices. Gilbert, Johnson, and Keerthi[19] 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[17] 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[37,43].

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; such algorithms are presented in Chapter 5. The collision detection algorithm in Gilbert, Johnson, and Keerthi[19] 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. Likewise, work by Cundall[10] 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 philosophy of collision/contact determination presented in Chapter 5 was largely inspired by Cundall's approach.

4.2 Impulse computation

The computation of impulses between two bodies that collide at a single contact point without friction is straightforward, as it involves the solution of a single linear

equation with one unknown. If equality constraints involving one 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 constraint. In this case, the impulses must be computed simultaneously, so a larger set of linear equations must be solved; this is simply a special case of solving equality constrained dynamics problems, and is not particularly difficult.

If the contact involves friction, analysis of the problem is more complicated, but the actual computation itself is simple, for two-dimensional systems. Wang and Mason[34,54] discuss the case of collisions with friction at a single contact point in two dimensions, and give a complete treatment of the problem. Their analysis of the problem yields simple formulas for the impulses arising from a collision. However, Wang and Mason's results are difficult to extend to three dimensions, which can be seen by considering Keller[24], who 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. We shall employ the same sort of analysis in considering impulses with friction in chapters 12 and 13. An animation system described by Moore and Wilhelms[39] uses penalty methods to compute constraint forces, but 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. We will use a similar simplification, applied to a number of contact points simultaneously (see below).

Collisions can also be regarded as occurring simultaneously at a number of contact points. For example, if a cube is dropped onto a level plane so that all

four vertices of the bottom face of the cube strike the plane together, the collisions between the vertices and the plane can be modeled as occurring at discrete instants of time, by the above methods, or as occurring simultaneously. There are large computational savings to be gained by modeling the collisions as occurring simultaneously. Cremer[8] discusses the advantages and disadvantages of the simultaneous model of collisions, and Featherstone[13] shows how to formulate the necessary equations for the frictionless case. The impact model for simultaneous collisions used in this thesis agrees with Featherstone's formulation, for the frictionless case. 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 this thesis computes simultaneous frictional impulses for three-dimensional bodies, based on a modification of the model proposed by Lötstedt. As stated, for frictionless systems, the model described in this thesis is the same as Featherstone's. However, for systems with friction, the model does not always properly conserve energy for impacts[3,54]. The general problem of correctly computing frictional impulses in three dimensions, and simultaneous frictional impulses in either two or three dimensions is not addressed by this thesis.

4.3 Penalty methods for constraint forces

A vast number of simulations[10,22,26,36,38,39,50,51,55] have employed the penalty method as described in Section 2.4 to enforce non-penetration constraints; applications include the simulation of deformable bodies, cloth, and articulated rigid bodies. The penalty method is a very attractive model in some respects; it is extremely simple to implement, and very versatile. As a numerical method, it is

nowhere near as complex as the methods we will consider throughout most of this thesis. In particular, exploiting parallelism and/or hardware implementations with the penalty method is much easier to imagine than with the methods we will be discussing throughout most of this thesis. Practically speaking though, the use of the penalty method in essentially dynamic conditions leads to stiff differential equations and is not a suitable method for the type of simulation we are interested in. The penalty method is reasonable only for situations that are essentially static; that is, where there is little relative motion between bodies and little to no change in constraint forces between bodies. We will consider the additional question of the theoretical correctness of the penalty method as a simulation algorithm in Chapter 7.

4.4 Exact methods for constraint forces

In comparison with the penalty method, relatively few dynamics systems use exact methods to calculate constraint forces. 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[25] and Ingleton[23]. Cottle[7] clarifies and simplifies some of Ingleton's results. Kilmister and Reeve, Ingleton, and Cottles' work will be our basic starting point when we consider systems with multiple contact points in Chapter 8. We will make use of the same quadratic programming formulation and extend the contact geometry allowed to include smooth curved surfaces.

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[12] 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. More recently, Featherstone[13] has described a heuristic algorithm for solving the quadratic programs generated for configurations without friction that is intended to improve upon the obvious 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. This is a very restrictive condition though, which we will not want to impose on our simulations. In this thesis, the computational complexity of computing multiple contact forces will be of great interest to us.

4.5 Friction

Adding a model of friction to simulations using the penalty method is straightforward. Computing both friction and constraint forces using exact methods is considerably more difficult. For two-dimensional problems, an obvious exhaustive search method exists, but is not suitable to simulations with many contact points because it requires exponential time in the number of contact points. Mason and Wang[35] discuss the computation of friction and constraint forces in systems with only a single contact point (where the time complexity of the computation is not an issue). Lötstedt[28], Erdmann[12], and Mason and Wang all discuss interesting properties of the Coulomb friction model in relation to rigid body simulation. Their discussions and examples of the unexpected consequences of introducing Coulomb friction provide the starting point for much of the discussion on friction in this thesis. In particular, Erdmann describes the problem of computing contact forces

in terms of configuration space (Chapter 6). We will borrow from Erdmann by using configuration space to develop the dynamics of systems with a single point of contact, although we will dispense with configuration space when we move on to systems with multiple contact points and friction, as the configuration space approach offers no additional insights into the problem. Additionally, Lötstedt[30] proposes a modification of the Coulomb friction law for rigid body simulation in conjunction with a practical computational solution method for the modified friction model. We will examine Lötstedt's modification and subsequent solution method in Chapter 14. Again, the computational complexity in computing multiple contact forces with friction will be of considerable interest.

Chapter 5

Collision/Contact Determination

In summarizing previous work on collision/contact determination in Section 4.1, we noted that of the two groupings of algorithms (the discrete view and the continuum view), the continuum view, which is commonly used in robotics, is not applicable to the dynamics problem we are interested in. The other approach, the discrete view, could be used to solve the collision/contact determination problem, but at the price of ignoring any knowledge due to the inherent spatial continuity of the problem as a function of time. Instead, a third approach, which we will call the "coherence view", will be adopted and used to solve the collision/contact determination problem. The use of coherence need not be confined to collision/contact determination; in fact, critical use will be made of coherence in computing constraint and friction forces. In general, methods that exploit coherence should be considered throughout the entire simulation process.

As stated in Chapter 3, the simulation process consists of the repeated computation of $\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{Y}(t)$ at various times t. The numerical ODE solver is responsible for choosing the values of t at which the state derivative is to be computed. For any reasonably complicated simulation, the values of t chosen are such that the state

does not change greatly between successive values of t. As a result, there is almost always great geometric coherence between successive time steps. It is difficult to formally quantize the degree of coherence, and we will not attempt to do so. In practice, the degree of coherence present during simulations has been such that the algorithms presented in this chapter are much faster than their discrete view counterparts. Additionally, their simplicity has greatly decreased the programming difficulties that arise with any complex simulation system.

The coherence view attempts to capitalize on geometric coherence to achieve both simpler and more efficient collision/contact determination algorithms.[1] The series of collision/contact determination problems that need to be solved during the simulation are considered, according to the coherence view, as separate problems; however, the problems are considered to be related, and wherever possible, information from the solution of the previous problem is used to aid in the solution of the current problem. Using this view, algorithms that are both faster *and* simpler than their discrete view algorithms counterparts are possible.

In this chapter, we will discuss three separate problems of collision/contact determination. The first problem we consider is pairwise collision/contact determination between convex polyhedra. Next, pairwise determination between convex curved surfaces or a convex curved surface and a convex polyhedron is considered. Last, we will consider a simple coherence-based algorithm for detecting all pairwise-intersections among a group of rectangular solids aligned with the coordinate axes. This algorithm is used at the beginning of the collision/contact determination to quickly cull out pairs of objects whose rectangular bounding boxes do not overlap;

[1] For those interested in building a practical, robust simulation system with a rich variety of object geometries, it is sometimes hard to say which is more valuable—increased simplicity or decreased running time. Given the tendency complicated geometrical algorithms have to get stuck and abort on the special cases that always arise in practice, it sometimes seems that slow but simple algorithms end up saving time over a complicated algorithm that requires the simulation to be run and re-run and re-re-run and ...

such a step is necessary to avoid performing $O(n^2)$ pairwise collision/contact determinations among a group of n bodies. Using the algorithms developed to solve these three problems, we will be able to perform collision/contact determination on any body that can be described as the union of convex polyhedra and strictly convex closed surfaces. These polyhedra and curved surfaces will be referred to as *primitives*. For curved surfaces primitives, only closed surfaces with strictly positive curvature at every point will be allowed for collision/contact determination. Note that although polyhedral primitives must also be convex, this is not really a restriction, since any polyhedron can be decomposed into a union of convex polyhedra. The general case of collision/contact determination between non-convex curved surfaces is beyond the scope of this thesis; however, this limitation will *not* be imposed when considering the dynamics of bodies in subsequent chapters.

The discussion of the collision/contact determination step in Section 3.2 made mention of the fact that the collision/contact determination step was responsible for estimating the time of a collision. This will be covered in Chapter 6.

5.1 Convex polyhedra

Our primary mechanism for exploiting coherence will be through the use of *witnesses* to the decision problem of inter-penetration between primitives. A witness is some piece of information that can be used to quickly answer a decision problem. We will utilize coherence by caching witnesses from one time step to the next; hopefully a witness from the previous time step will be a witness during the current time step.

Since primitives are convex, a pair of primitives do not inter-penetrate if and only if a separating plane between them exists. A separating plane between two objects is a plane such that each object lies in a different half-space of the plane. A given plane can be verified to be a separating plane between two polyhedra in time $O(n)$

where n is the total number of vertices in the two polyhedra. Thus, a separating plane is a witness to the fact that two convex polyhedra do not inter-penetrate.

We would like to structure our algorithms so that their running time decreases as coherence increases. The basic philosophy behind utilizing coherence through cached witnesses is to choose a very simple, fast strategy that proposes a new witness based on the cached witness, and then quickly attempts to verify the new witness. If the proposed witness is invalid, a second, more time-consuming step is executed to find a valid witness. The idea is that the first step is fast enough, and succeeds often enough to more than make up the time lost on the occasions when it does fail.[2] In particular, the strategy should be extremely simple, so that it has both a small asymptotic time complexity and run time constant. The cost of initially finding a witness (for the very first time step of the simulation, or the first time two bodies become close enough to require more than a bounding box test) is unavoidable; either exhaustive search algorithms or the discrete view algorithms described in Section 4.1 can be used to initially compute these witnesses.

As mentioned in Section 4.1, algorithms by Cundall[10] and Gilbert, Johnson and Keerthi[19] come close to this philosophy. In particular, the algorithm Gilbert, Johnson and Keerthi describe could be initialized with the solution of the previous problem to decrease the running time; however, the initialized algorithm is still much more complicated than is needed for situations with a high degree of coherence. Cundall's approach to collision/contact detection is much simpler, and in fact inspired the coherence philosophy just presented. Cundall called a separating plane between two convex polyhedra a *common plane.* For each pair of convex polyhedra requiring consideration, he used numerical techniques to initially determine a separating plane that was approximately equidistant from each polyhedron of the pair; if such a plane

[2] In general, such a philosophy can be implemented hierarchically, as is done for memory caching on computers, but we will not bother to do so.

could not be found, then clearly the two polyhedra had inter-penetrated. He then noted several important facts about the separating plane. If the polyhedra were disjoint, this was easily shown because each polyhedra could trivially be shown to lie some distance from the separating plane. Furthermore, if the polyhedra were contacting, then the contact points were easy to find because they would have to lie on a common plane of contact (hence the term *common plane*); this greatly reduced the complexity of determining the contact points. Most important was Cundall's realization that it was not necessary to compute separating planes from scratch at each time step; a simple numerical iteration could compute a new separating plane by using the old separating plane as a starting point, because of the geometric coherence between successive time steps.

Cundall's approach is good, but it can be made even better. If a pair of convex polyhedra are disjoint or contacting (but not inter-penetrating), then a separating plane exists with the following property: either the plane embeds a face of one of the polyhedra, or the plane embeds an edge of one of the polyhedra and is parallel to an edge of the other polyhedra. We will call the face or edges in question the *defining* face or edges. Instead of caching a specific plane in $\mathbf{R}^3$ as the witness and numerically updating it, the defining face or edges are cached as the witness. At the next time step, a separating plane is formed from the defining face or edges, and then verified. The verification is simple; the vertices of the two polyhedra are checked to see that they lie on opposite sides of the separating plane. Since this verification takes $O(n)$ time (where n is the number of vertices), a linear time algorithm results whenever the cached face or two edges form a separating plane. Clearly, in addition to having a linear time cost, this algorithm will be considerably faster than other $O(n)$ algorithms, such as the linear programming algorithm discussed in Section 4.1.

However, an additional improvement can be made so that this verification can usually be performed in sublinear time. It is not always necessary to test all the

vertices of a polyhedron to verify that the polyhedron lies completely on one side of the separating plane. Instead, for each polyhedron, the vertex lying closest to the separating plane is cached. If at the next time step that cached vertex is closer to the separating plane than all of its neighbors, then by convexity all of the vertices of the polyhedron must lie on the same side of the separating plane. This gives the result that disjointness or contact between convex polyhedra can usually be determined in sublinear time when coherence is high.

On those occasions when the cached face or two edges fails to form a valid separating plane, faces or edges adjacent to the previously cached face or edges can be used to form a possible separating plane; however, this happens infrequently enough that it may be simpler to start from scratch and compute a new separating plane without using any prior knowledge. It should be obvious that the linear time cost of verifying a separating plane is of a much different order than the linear time cost of using a technique like linear programming[43] to find a separating plane; although both algorithms have $O(n)$ time complexity, their run time constants are considerably different.

Once the separating place has been found, the contact region between the two polyhedra is determined, assuming the polyhedra are not disjoint. Contact points between the two polyhedra can only occur on the separating plane. Given the separating plane, the contact points can be quickly and efficiently determined by comparing only those faces, edges, and vertices of the polyhedra that are coincident with the separating plane.

However, if no separating plane can be found, then the two polyhedra must be inter-penetrating. When two polyhedra inter-penetrate, it is almost always the case that either a vertex of one polyhedron is inside the other, or an edge of one

polyhedron has intersected a face of the other.[3] In this case, the inter-penetrating vertex, or intersecting edge and face are cached as a witness to the inter-penetration. Since this indicates a collision at some earlier time, the simulator will back up and attempt to compute $\frac{d}{dt}\mathbf{Y}(t)$ at some earlier time. Until the collision time is determined, the first action taken by the collision/contact determination step will be to check the cached vertex or edge and face to see if they indicate inter-penetration. Thus, until the collision time is found, states in which the inter-penetration still exists are identified as such with a minimum of computational overhead. In Chapter 6, we will show how the collision time may be estimated once an inter-penetration is detected.

5.2 Convex curved surfaces

The surfaces considered in this thesis are assumed to be twice-differentiable and without boundary; in this chapter, surfaces are also assumed to be strictly convex. Before we can consider collision/contact determination between curved surfaces, we must decide how the curved surfaces are to be represented.

There are basically two choices—surfaces can either be described by implicit functions or parametric functions. In the implicit case, a surface is described as a time-varying function $F(p,t)$ where p is a point in world space. At time t, point p is on the surface represented by F if

$$F(p,t) = 0. \tag{5–1}$$

In the parametric case, a surface is described by a time-varying function $S(u,v,t)$ where u and v are scalars and $S(u,v,t)$ is a point on the surface. If F and S

[3] An exception is the following. Stack two cubes of equal size atop one another so that their contacting faces exactly coincide. Lower the top one. This produces an inter-penetration such that no vertex is inside either cube, and no edge penetrates through any face.

describe the same surface, then

$$F(S(u,v,t),t)$$

is identically zero.

We will not attempt any formal manipulations of F and S; that is, it is assumed F and S are given to us as "black-box" functions. The only allowable operations on F or S are evaluating F or S, or their first or second derivatives at specific values of p and t (for F) and u, v and t (for S). For simplicity, derivations will be developed in the text only for surfaces described implicitly. Matching derivations for parametrically defined surfaces are found in Appendix A.

We will consider two convex curved surfaces A and B that are described implicitly by time-varying functions $F(p,t)$ and $G(p,t)$. Any point p_0 lying on A at time t satisfies $F(p_0,t) = 0$. Furthermore, a point p_1 that is *inside* of A satisfies $F(p_1,t) < 0$, while a point p_2 that is *outside* of A satisfies $F(p_2,t) > 0$. The other surface B is similarly defined in terms of G, and we will sometimes refer to F and G as both functions and surfaces.

The surface normal of F at point p and time t is written as $\nabla F(p,t)$, and is a column vector. Formally,

$$\frac{\partial}{\partial p}F(p,t) = F'(p,t) = \nabla F(p,t)^T.$$

Since F is negative on the interior of A and positive on the exterior of A, the surface normal $\nabla F(p,t)$ is always outwards pointing.

5.2.1 Extremal points of curved surfaces

First, let us consider the problem of verifying that two convex curved surfaces A and B do not inter-penetrate. Since A and B are strictly convex smooth surfaces, if

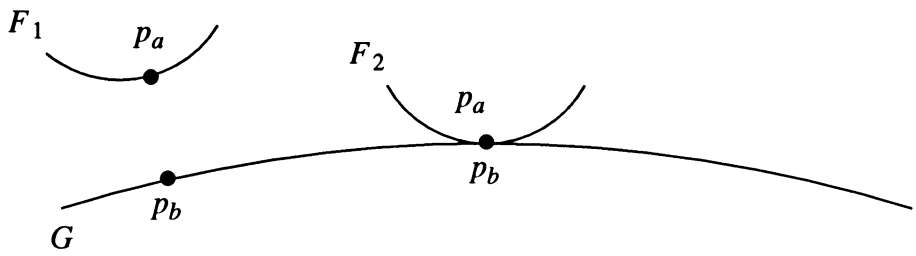


Figure 5.1: The minimum distance points p_a and p_b between two convex surfaces.

they do not inter-penetrate then there is a unique pair of points p_a and p_b lying on A and B that minimize the distance between A and B (Figure 5.1). If these points can be found, then they clearly demonstrate that A and B are not inter-penetrating. Furthermore, if the bodies are in contact, then $p_a = p_b$ is the contact point between A and B. Thus, in both cases, the pair of points p_a and p_b minimizing the distance between A and B seems a good choice for a witness.

What about the case when A and B inter-penetrate? In this case, the minimum distance between A and B is zero. In detecting contact between surfaces, if the depth δ of inter-penetration between A and B falls below some numerical threshold ϵ, then A and B must be considered to be contacting, rather than inter-penetrating. For instance, in Figure 5.2, suppose the inter-penetration depth between F_1 and G is δ_1; similarly, let δ_2 be the inter-penetration depth between F_2 and G. From the figure, $\delta_1 < \delta_2$. If

$$\delta_1 < \epsilon < \delta_2$$

then F_1 and G would be considered to be contacting, while F_2 and G would be considered to be inter-penetrating. Thus, when surfaces are not disjoint, the minimum distance, which is always zero, is not a useful piece of information. Although the points of intersection are not directly useful, one thought might be to examine the angle between surfaces at intersection points; however, as Figure 5.2

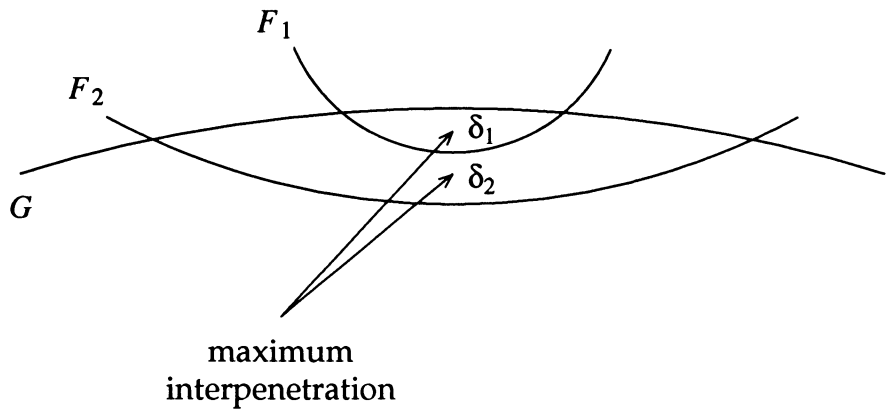


Figure 5.2: Inter-penetration between surfaces. Surface F_1 intersects G at a sharper angle than surface F_2 does, yet the maximum inter-penetration depth δ_2 between F_2 and G is larger than the maximum inter-penetration depth δ_1 between F_1 and G.

shows, the angle of intersection is independent of the depth of inter-penetration. For inter-penetration, the points of interest are the points that indicate the *maximum* distance between A and B in their intersection. Thus, a suitable witness for inter-penetration are the points of maximum distance between A and B.

We can formalize and unify our definition of a witness for both the inter-penetrating situation and the non-inter-penetrating situation by defining the general concepts of the *extreme distance* between A and B, and the *extremal points* between A and B. The extreme distance between A and B is defined as follows. If A and B are disjoint, then the extreme distance between A and B is just the normal minimum distance between A and B. If A and B are in contact, then the extreme distance is zero. If A and B have inter-penetrated, then the extreme distance is the maximum distance between A and B in their intersection. The extremal points of A and B are the two points p_a and p_b, lying on A and B respectively, that realize the extremal distance (Figure 5.3).

The extremal points can be computed by considering the following four necessary

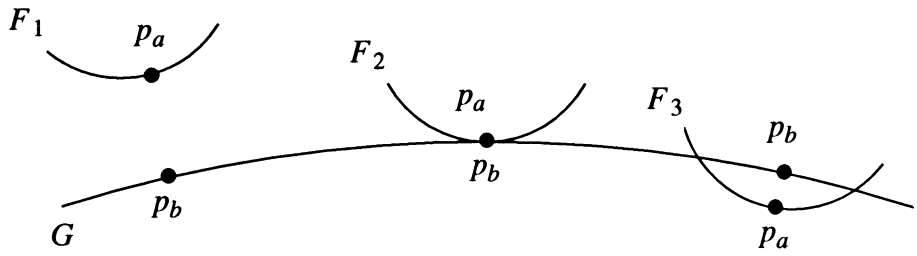


Figure 5.3: The definition of the extremal points. The extremal points p_a and p_b between F_1 and G are the points of minimum distance. The same is true for F_2 and G. The extremal points p_a and p_b between F_3 and G are the points that maximize the distance between F_3 and G in their intersection.

conditions for a pair of points p_a and p_b to be extremal at time t:

$$\begin{cases} E_1: & \nabla F(p_a,t) + \lambda_2 \nabla G(p_b,t) = \mathbf{0} \\ E_2: & F(p_a,t) = 0 \\ E_3: & G(p_b,t) = 0 \\ E_4: & (p_b - p_a) + \lambda_1 \nabla G(p_b,t) = \mathbf{0}. \end{cases} \tag{5–2}$$

The variables λ_1 and λ_2 are unconstrained scalar values (with no relation to the λ of Section 2.5) and $\mathbf{0}$ is a column vector of zeroes. Condition E_1 guarantees that the surface normals of F and G at p_a and p_b are colinear. Conditions E_2 and E_3 guarantee that p_a and p_b are points on A and B, and condition E_4 guarantees that p_b's displacement from p_a is colinear to the surface normals. Figure 5.4 shows the geometric intuition behind these four conditions; they are derived based on the Lagrange multiplier formulation for constrained minimization (hence the choice of λ for the multipliers of ∇G).

These four conditions form a non-linear set of eight equations in eight unknowns. However, the conditions are necessary but not sufficient for p_a and p_b to be extremal points. Because Equation (5–2) is non-linear, it will have multiple solutions. For example, the points that globally *maximize* the distance between A and B always

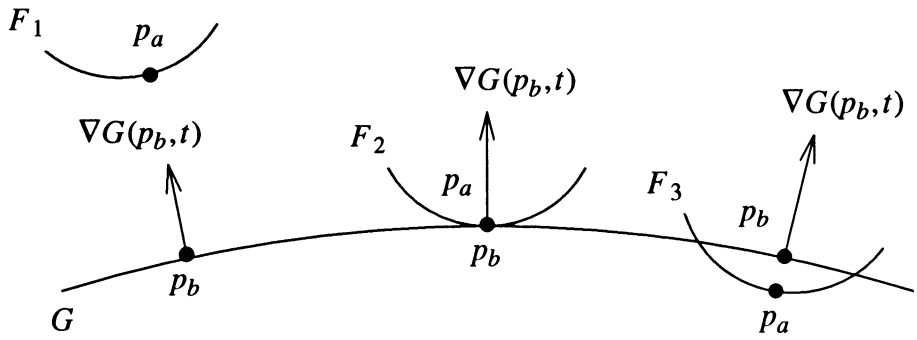


Figure 5.4: Geometric conditions for the extremal points. In order for p_a and p_b to be extremal, the surface normals ∇F and ∇G must be parallel, and point directly from p_a to p_b.

satisfy Equation (5–2), but are not a solution we are interested in. Non-linear equations are usually solved numerically using iterative methods. The numerical method proceeds from some initial estimate of the solution to an exact solution (within numerical tolerances). The initial estimate determines which solution is computed[11].

If we are initially computing the extremal points, we need an initial estimate sufficiently close to the extremal points to avoid computing a spurious solution. For implicitly defined convex surfaces, the following simple device can be used. A line segment passing though the centroids of A and B is constructed. The intersection of this line segment and A is roughly computed; that is, a point p on the line segment satisfying $F(p,t) = 0$ is computed. This can be done very easily using bisection, and since only a rough approximation is needed, a small number of bisection steps will be sufficient (ten or less). Similarly, an intersection between the line segment and B is computed. Using the two intersections as initial estimates for p_a and p_b, Equation (5–2) is iteratively solved to find p_a and p_b exactly. For parametric surfaces, a coarse mesh of surface points can be generated and stored for each surface. The parametric coordinates of the minimum distance pair of mesh points

can be used as an initial estimate. The initial solution of Equation (5–2) requires a fair amount of work, in terms of generating an initial estimate of p_a and p_b, and then solving Equation (5–2).

However, once the extremal points p_a and p_b are determined, they are cached for the next step. In subsequent time steps, the cached values of p_a and p_b are used as initial estimates in the numerical solution of Equation (5–2). Because of coherence, the cached values p_a and p_b will greatly reduce the number of iterative steps needed to numerically solve Equation (5–2). Furthermore, the accuracy of the initial estimate can be substantially improved if the *derivatives* of the extremal points are cached at each time step. Suppose that at time t_0, the extremal points p_a and p_b, and their derivatives $\dot{p}_a$ and $\dot{p}_b$ are cached. If the next time step takes place at time $t_0 + \Delta t$, p_a and p_b at that time can be estimated as

$$p_a + \Delta t\, \dot{p}_a \qquad \text{and} \qquad p_b + \Delta t\, \dot{p}_b \tag{5–3}$$

respectively. Improving the accuracy of the initial estimate of p_a and p_b increases the overall speed of the algorithm. Sections 6.3 and 6.4 will discuss the computation of $\dot{p}_a$ and $\dot{p}_b$.

5.2.2 Polyhedron/curved surface determination

Essentially the same method for collision/contact determination between a pair of curved surfaces can be used for collision/contact determination between a curved surface and a polyhedron. The concept of extremal points is again used, however, different formulations are needed depending on whether the extremal point on the polygon lies on a face, an edge, or a vertex.

Let the curved surface be denoted A, and be described implicitly by a function F as before. Let B denote the polyhedron. Let the plane equation of a face of B be implicitly described by a function $G(p, t)$. Necessary conditions for the

extremal points to occur between this face and A are given by Equation (5–2), with the restriction that p_b actually lie inside the face. A simple initial step to compute p_a and p_b is to test each face of B against A, stopping if valid extremal points are found. Clearly, not all faces of B need be considered; faces of B whose outwards surface normal points away from A can obviously be ignored. However, if no extremal points exist between a face of B and A, then the edges of B must be tested.

An edge of B with endpoints P_0 and P_1 can be parameterized as the set of points $P_0 + cr$, where $r = P_1 - P_0$ and $0 \leq c \leq 1$. Necessary conditions for p_a and $p_b = P_0 + cr$ to be extremal are

$$\begin{cases} \nabla F(p_a, t)^T r = 0 \\ F(p_a, t) = 0 \\ (p_a - (P_0 + cr)) + \lambda_1 \nabla F(p_a, t) = \mathbf{0} \end{cases} \tag{5–4}$$

along with the condition that p_b be contained in the edge; that is, $0 \leq c \leq 1$. If no extremal points exist between an edge of B and A, then an extremal point will exist between a vertex P_0 of B and A. The conditions for a vertex P_0 of B and p_a to be extremal are

$$\begin{cases} F(p_a, t) = 0 \\ (p_a - P_0) + \lambda_1 \nabla F(p_a, t) = \mathbf{0}. \end{cases} \tag{5–5}$$

Once an extremal point is found between a face, edge, or vertex, the extremal points are cached, along with the face, edge or vertex on which the extremal point occurred. Subsequent solutions of the extremal points then take advantage of knowing which face, edge, or vertex to test to find the extremal points, along with a good starting estimate of the extremal points. In all cases, the velocity $\dot{p}_a$ is cached. If the extremal point lies on a face, $\dot{p}_b$ is also cached, and if the extremal point lies on an edge, $\dot{c}$ is also cached.

5.3 Bounding boxes

To reduce the number of pairwise collision/contact determinations necessary, a bounding box hierarchy can be imposed on the objects in the simulation environment. If two bounding boxes are found not to overlap, no further comparisons involving the contents of the boxes are needed. Given a collection of n rectangular bounding boxes, aligned with the coordinate axes, we would like to efficiently determine all pairs of boxes that overlap. A naive pairwise comparison of all pairs requires $O(n^2)$ work, and can clearly be improved. Discrete view algorithms for this problem in $\mathbf{R}^3$ exist with a time complexity of $O(nlogn + k)$ where k is the number of pairwise overlaps; a general result is that the problem can be solved in time $O(nlog^{d-2}n + k)$ for d-dimensional bounding boxes[43]. Again, a coherence view algorithm exists with substantially better performance.

5.3.1 The one-dimensional case

Consider the problem of detecting overlap between *one*-dimensional bounding boxes, aligned with the coordinate system. Such a bounding box can be described simply as an interval $[b, e]$. Let us consider a list of n such intervals, with the ith interval being $[b_i, e_i] \subset \mathbf{R}^1$. The problem is then defined to be the determination of all pairs i and j such that the intervals $[b_i, e_i]$ and $[b_j, e_j]$ intersect.

The problem can be solved initially by a *sort and sweep* algorithm. A sorted list of all the b_i and e_i values is created, from lowest to highest. The list is then swept, and a list of *active* intervals, initially empty, is maintained. Whenever some value b_i is encountered, all intervals on the active list are output as overlapping with interval i, and interval i is then added to the list (Figure 5.5a). Whenever some value e_i is encountered , interval i is removed from the active list (Figure 5.5b). The cost of this process is $O(n \log n)$ to create the sorted list, $O(n)$ to sweep through the list,

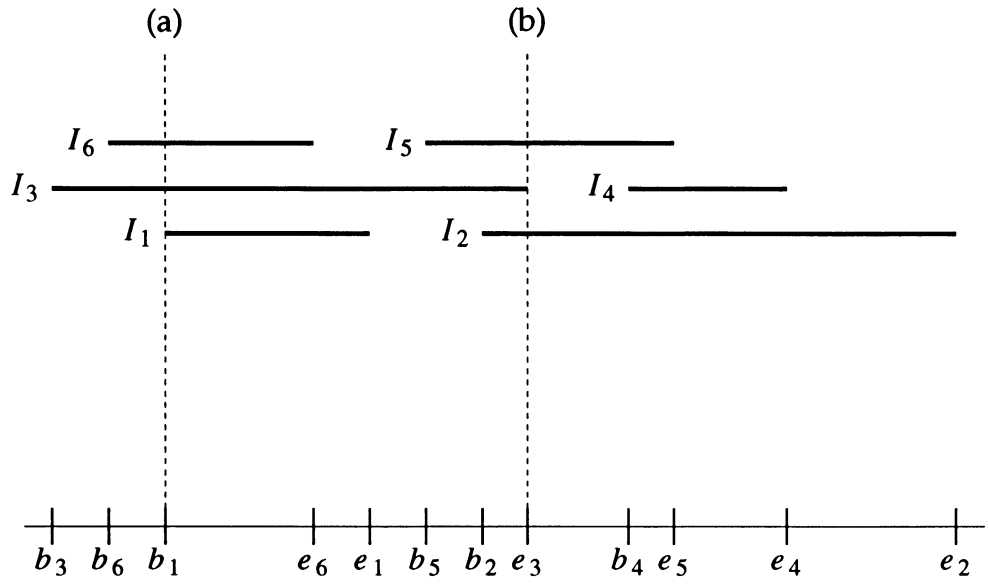


Figure 5.5: The sweep/sort algorithm. (a) When b_1 is encountered, the active list contains intervals 3 and 6; interval 1 is reported to overlap with these two intervals. Interval 1 is added to the active list and the algorithm continues. (b) When e_3 is encountered, the active list contains intervals 2, 3 and 5. Interval 3 is removed from the active list.

and $O(k)$ to output each overlap. This gives a total cost of $O(n \log n + k)$, and is an optimal algorithm for initially solving the problem.

Subsequent comparisons can be improved as follows. First, there is no need to use an $O(n \log n)$ algorithm to form the sorted list of b_i and e_i values. It is considerably more efficient to start with the order found for b_i and e_i values from the previous time step; if coherence is high, this ordering will be nearly correct. A sorting method called an *insertion sort*[46] is used to permute the "nearly sorted" list into a sorted list. The insertion sort algorithm works by moving items towards the beginning of the list, until a smaller item is encountered. Thus, the second item is interchanged with the first if necessary, then the third item is moved towards the beginning of the list until its proper place is found, and so on; each movement of an item indicates a change in the ordering of two values. After the last item on the list

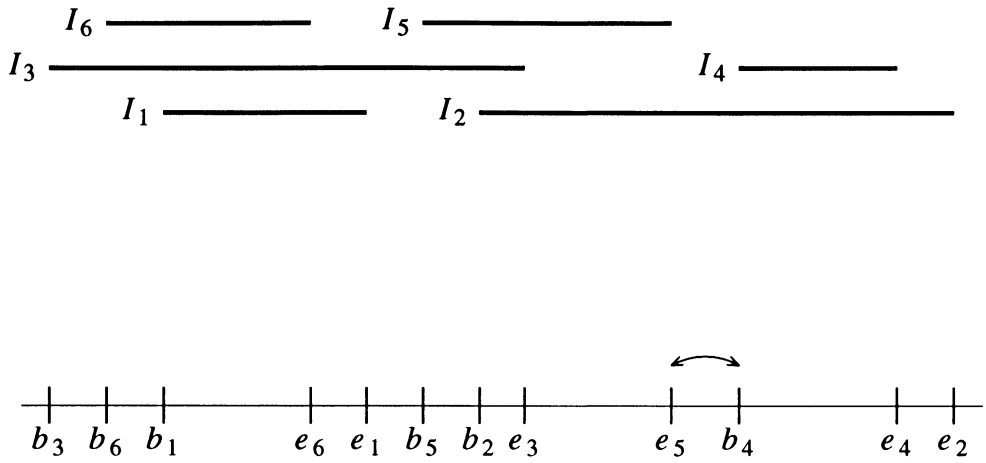


Figure 5.6: A coherence-based method of detecting overlaps. The order produced in Figure 5.5 is nearly correct for this arrangement of intervals. Only b_4 and e_5 need to be exchanged. When the exchange occurs, the change in overlap status between interval 4 and 5 is detected.

has been processed, the list is sorted. Such a sort takes time $O(n+s)$ where s is the number of "swaps" necessary to reorder the list. For example, the only difference between Figures 5.6 and 5.5 is that interval 4 has moved to the right. Starting from the ordered list of b_i and e_i values of Figure 5.5, only a single exchange is necessary to sort the list for Figure 5.6. The insertion sort is not recommended as a sorting procedure in general, since it may require $O(n^2)$ exchanges; however, it is a good algorithm for sorting a nearly sorted list, which is what occurs in our highly coherent environment. To complete the algorithm, note that if two intervals i and j overlap at the previous time step, but not at the current time step, one or more exchanges involving either a b_i or e_i value and a b_j or e_j value must occur. The converse is true as well when intervals i and j change from not overlapping at the previous time step to overlapping at the current time step.

Thus, if we maintain a table of overlapping intervals at each time step, the table can be updated at each time step with a total cost of $O(n+s)$. An optimal algorithm

would have time complexity $O(n+k)$, so our algorithm, with its $O(n+s)$ complexity is efficient when $s-k$ is small. Assuming coherence, the number of exchanges s necessary will be close to the actual number k of changes in overlap status, and the extra $O(s-k)$ work will be negligible. Thus, for the one-dimensional bounding box problem, the coherence view yields an efficient algorithm of extreme (if not maximal) simplicity that approaches optimality as coherence increases.

5.3.2 The three-dimensional case

Efficient discrete view algorithms for solving the bounding box intersection problem in $\mathbf{R}^3$ are much more complicated than the sweep and sort method for the one-dimensional case. However, these algorithms all have in common a step that is essentially a sort along a coordinate axis, as in the one-dimensional case. Each bounding box is described as three independent intervals $[b_i^{(x)}, e_i^{(x)}]$, $[b_i^{(y)}, e_i^{(y)}]$, and $[b_i^{(z)}, e_i^{(z)}]$ which represent the intervals spanned on the three coordinate axes by the ith bounding box. Thus, our first thought towards improving the efficiency of a discrete view algorithm for coherent situations would be to sort a list containing the $b_i^{(x)}$ and $e_i^{(x)}$ values, and similarly for the y and z axes. Again, such a step will involve $O(n+s)$ work, where now s is the total number of swaps involved in sorting all three lists. However, if we observe that checking two bounding boxes for overlap is a constant time operation, it follows that if we simply check bounding boxes i and j for overlap whenever an exchange is made between values indexed by i and j (on any coordinate axis), we will detect all changes in overlap status in $O(n+s)$ time.

Again, we can maintain a table of overlapping bounding boxes, and update it at each time step in $O(n+s)$ time. The extra work involved is again $O(s-k)$. For the three-dimensional case, extra work can occur if the extents of two bounding boxes

change on one coordinate axis without an actual change of their overlap status. In practice, the extra work done has been found to be completely negligible, and the algorithm runs essentially in time $O(n + k)$.

Chapter 6

Local Motion Constraints

In this chapter, the basic motion constraint that prevents inter-penetration from occurring near contact points is derived. The motion constraint at one contact point does not prevent inter-penetration at another contact point, and additional motion constraints are required. For this reason, the motion constraint at each contact point is said to be a *local* constraint, as it is responsible for preventing inter-penetration from occurring only in the immediate vicinity of the contact point, and only over some (possibly small) range of motion. This locality property, and the fact that the constraint is meant to apply only over some small range of motion is to be implicitly assumed in the rest of our discussion.

We will start by expressing the non-penetration constraint between bodies in terms of the spatial variables of the contacting bodies. By differentiating this constraint twice with respect to time, a constraint on the accelerations of the bodies is produced. Chapter 8 will show how workless constraint forces can be computed that satisfy this acceleration constraint for each contact point. The derivation of

these acceleration constraints is well known for certain special cases of contact, but as far as we know a formulation for the general case has been lacking in the literature.

6.1 Configuration Space

The problem of preventing inter-penetration at any one contact point between two bodies can be viewed as a more complicated version of the simple particle/surface problem of Chapter 2, by considering the contact in terms of the *configuration space*, or C-space, of the two bodies[1,12,31,32]. The C-space of a rigid body is a six-dimensional space, representing the degrees of freedom of the body. A single point in this C-space is specified by six coordinates; each separate coordinate represents one degree of freedom of the body. Thus, each point of this C-space represents a particular configuration (a position and orientation) of the rigid body. The C-space of a pair of rigid bodies is 12 dimensional; a single point of this C-space represents a configuration for each body. We will describe the motion constraint between two bodies at a single contact point using this 12 dimensional C-space.

Points in C-space will be denoted by a horizontal bar, such as $\overline{p}$, while points in world space ($\mathbf{R}^3$) will be written simply as p. Both points p in $\mathbf{R}^3$ and points $\overline{p}$ in C-space denote column vectors. Partial derivative operators $\partial/\partial p$ and $\partial/\partial \overline{p}$ yield column vectors when applied to functions.

At time t_0, consider two bodies A and B that contact at p_c, and whose configuration is represented by the point $\overline{x}_0$. By considering the contact in C-space, we can employ the standard technique of viewing the constraint that A and B remain in contact near p_c as a manifold S, containing $\overline{x}_0$, in some neighborhood of $\overline{x}_0$. The set of all points in C-space near $\overline{x}_0$ that represent configurations such that A and B remain in contact (near p_c) is a manifold, S, containing $\overline{x}_0$. Moreover S is a "hypersurface" in C-space; that is, S has an 11-dimensional tangent space and

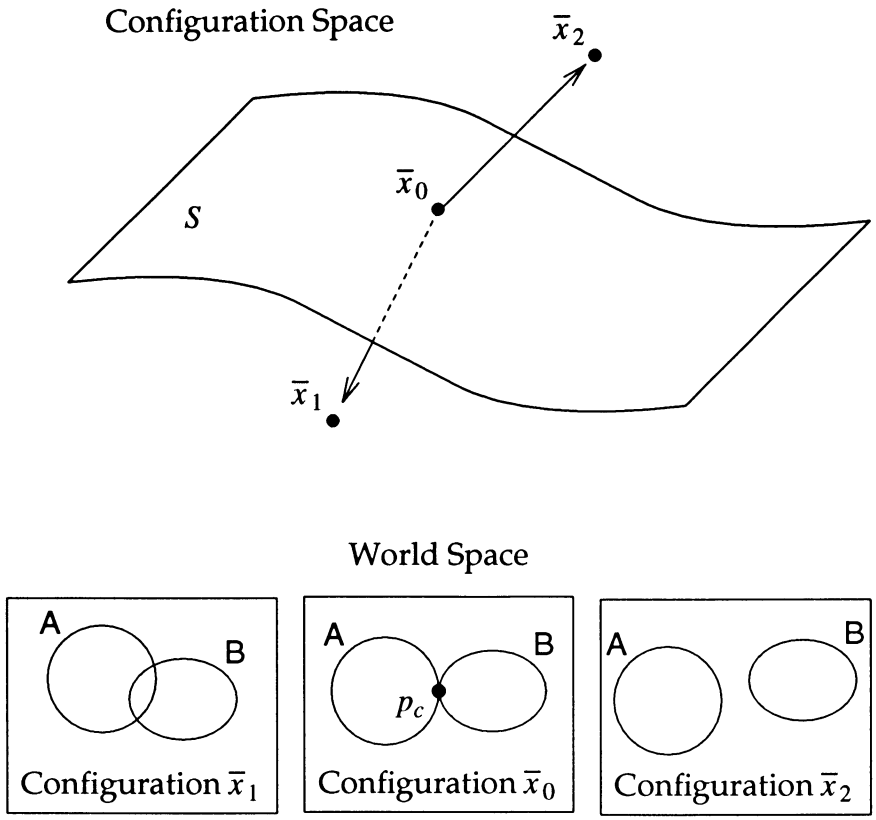


Figure 6.1: Three configurations in world space and their representation as points in C-space. The points $\overline{x}_1$ and $\overline{x}_2$ lie on opposite sides of the hypersurface S.

a single normal direction at each point.

The hypersurface S partitions C-space near $\overline{x}_0$ into two regions. Points near $\overline{x}_0$ on one side of S represent configurations where A and B are disjoint near p_c; points near $\overline{x}_0$ on the other side represent configurations where A and B have inter-penetrated near p_c (Figure 6.1).

Since we are interested in the behavior of A and B over time, let $\overline{x}(t)$ be the configuration of A and B at time t. Then $\overline{x}_0 = \overline{x}(t_0)$. We will describe S implicitly in terms of a scalar function C of C-space; that is,

$$S = \{\, \overline{p} \mid C(\overline{p}) = 0 \,\}. \tag{6–1}$$

We will adopt the convention that if $C(\overline{p}) < 0$, then $\overline{p}$ represents a configuration such that A and B inter-penetrate near p_c. Conversely, if $C(\overline{p}) > 0$, then $\overline{p}$ represents a configuration such that A and B are disjoint near p_c. Since A and B contact at p_c at time t_0, we know that

$$C(\overline{x}(t_0)) = 0. \tag{6–2}$$

In Figure 6.1, C is positive for configurations lying "above" S; thus, valid (non-penetrating) configurations are those that lie on or "above" S.

Given the above definitions, the C-space representation of the non-penetration constraint for an arbitrary contact has the same form as the non-penetration constraint for the point/surface problem of Section 2.3. To prevent inter-penetration, $\overline{x}(t)$ must be prevented from lying below S; thus

$$C(\overline{x}(t)) \geq 0 \tag{6–3}$$

enforces the non-penetration constraint. Comparing this equation with Equation (2–4), the analogy between the particle/surface non-penetration constraint and the non-penetration constraint for an arbitrary contact is obvious.

6.2 Projections onto S

Equation (6–3) expresses the non-penetration constraint in terms of spatial variables. We convert the non-penetration constraint into a constraint on acceleration by differentiating. Consider

$$\dot{C}(\overline{x}(t)) = \frac{\mathrm{d}}{\mathrm{dt}} C(\overline{x}(t)).$$

The function $\dot{C}(\overline{x}(t_0))$ measures relative velocity in the same manner as $\dot{C}(x(t_0))$ of Equation (2–10). If $\dot{C}(\overline{x}(t_0)) < 0$, then A and B are colliding (near p_c), while if

$\dot{C}(\overline{x}(t_0)) > 0$ then A and B are breaking contact at p_c; otherwise $\dot{C}(\overline{x}(t_0)) = 0$ and A and B are neither colliding nor separating at p_c. Since collisions are assumed to have been dealt with prior to this stage, and the non-penetration constraint would not be active if A and B were separating at p_c, we can assume $\dot{C}(\overline{x}(t_0)) = 0$.

Since $C(\overline{x}(t_0)) = \dot{C}(\overline{x}(t_0)) = 0$, inter-penetration is prevented by the condition

$$\ddot{C}(\overline{x}(t_0)) = \frac{\mathrm{d}^2}{\mathrm{dt}^2} C(\overline{x}(t_0)) \geq 0. \tag{6-4}$$

The quantity $\ddot{C}(\overline{x}(t_0))$ is a measure of the *acceleration* between A and B at p_c, in the normal direction. If $\ddot{C}(\overline{x}(t_0))$ is positive, then A and B are breaking contact, while if $\ddot{C}(\overline{x}(t_0))$ is zero, then A and B are remaining in contact.

Given any net forces and torques applied to A and B, these forces and torques can be translated into a net C-space applied force acting on the configuration point $\overline{x}_0$. If the contact between A and B is frictionless, then a C-space reaction force normal to the hypersurface S at $\overline{x}_0$ arises (if necessary) to prevent $\overline{x}(t)$ from moving downward.[1] The C-space reaction force is computed by projecting the C-space applied force onto the C-space normal, and reversing its direction (Figure 6.2).

Thus, it is assumed by those accustomed to working in C-space that computing the reaction force, in either C-space or world space, is a simple matter. However, the computation of the net C-space applied force can be difficult, because it must take into account inertial forces generated by the motion of A and B. Consider $\dot{C}(\overline{x}(t_0))$. Applying the chain rule,

$$\frac{\mathrm{d}}{\mathrm{dt}} C(\overline{x}(t_0)) = \frac{\partial C}{\partial \overline{p}} (\overline{x}(t_0))^T \dot{\overline{x}}(t_0). \tag{6-5}$$

[1] The normal direction is defined by the the natural inner product induced by the kinetic energy of the system. The projection of Figure 6.2 must also be performed using the natural inner product, but the intuition behind Figure 6.2 is valid. Erdmann[12] contains a more detailed discussion of the subject.

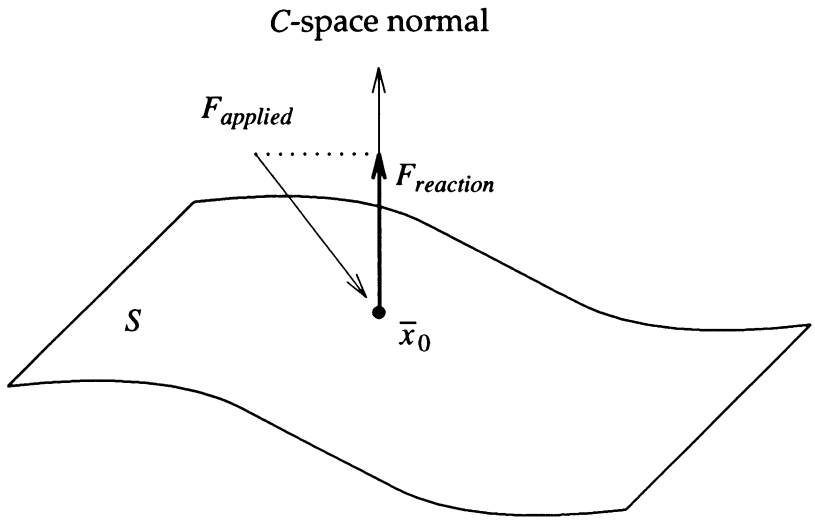


Figure 6.2: The reaction force $F_{reaction}$. The reaction force is computed by projecting the applied force $F_{applied}$ onto the C-space normal of S.

(Recall that both $\partial C/\partial \overline{p}$ and $\dot{\overline{x}}$ are column vectors.) Differentiating again,

$$\begin{aligned}\frac{\mathrm{d}^2}{\mathrm{dt}^2}C(\overline{x}(t_0)) &= \left(\frac{\partial}{\partial \overline{p}}\frac{\partial C}{\partial \overline{p}}(\overline{x}(t_0))\dot{\overline{x}}(t_0)\right)^T \dot{\overline{x}}(t_0) + \frac{\partial C}{\partial \overline{p}}(\overline{x}(t_0))^T \ddot{\overline{x}}(t_0) \\ &= \dot{\overline{x}}(t_0)^T \frac{\partial^2 C}{\partial \overline{p}^2}(\overline{x}(t_0))\dot{\overline{x}}(t_0) + \frac{\partial C}{\partial \overline{p}}(\overline{x}(t_0))^T \ddot{\overline{x}}(t_0) \qquad (6\text{–}6)\end{aligned}$$

with

$$\frac{\partial^2 C}{\partial \overline{p}^2}(\overline{x}(t_0))$$

the (symmetric) 12×12 matrix of second partial derivatives of C at $\overline{x}(t_0)$. The right term of the last sum in Equation (6–6) is the component of acceleration of $\overline{x}(t_0)$ normal to the C-space surface S. The left term of the last sum represents the inertial forces acting on A and B. In order to evaluate this term it is necessary to calculate the second partial derivatives of C at $\overline{x}_0$. This has been done only for the case of polyhedral contact[12], but appears not to have been done for the more general case of contact between curved surfaces. Thus, the computation of the

reaction force when either there is no initial motion ($\dot{\overline{x}}(t_0) = 0$) or the contacting bodies are polyhedral is simply accomplished. In the next section, the second partial derivatives of C are computed for the more general case of contacting surfaces in motion. The derivation for the second partial derivatives of C yields a result that is not particularly tractable for use in a simulation environment. As a result, in Section 6.4, a more tractable expression for Equation (6–6) is derived, that does not explicitly involve the computation of the second partial derivatives of C. However, an analytical expression for the second partial derivatives of C may be useful in some contexts, so a suitable expression is derived in the next section.

6.3 Calculating $\partial^2 C/\partial \overline{p}^2$

In order to create a constructive definition of C for curved surfaces, we will use the concepts of the extremal points between two curved surfaces, from Section 5.2.1. Since surfaces need not be convex in this chapter, the extremal points p_a and p_b can only be defined locally, in the vicinity of a contact point. The extremal points near p_c are defined as follows. If A and B are disjoint near p_c, then the extremal points minimize the distance between A and B near p_c. If A and B are in contact at p_c, then the extremal points are $p_a = p_b = p_c$. If A and B have inter-penetrated near p_c, then the extremal points maximize the distance distance between A and B (near p_c). The points p_a and p_b at a contact point vary according to the configuration $\overline{p}$ of A and B; thus, p_a and p_b may be considered as functions of configuration (Figure 6.3).

Again, as in Section 5.2, we will model the surface of A and B near p_c implicitly by functions F and G; however, F and G will now be functions of *configuration* as opposed to time. That is, if the configuration of A and B is $\overline{p}$, then a point p is on the surface of A if $F(p, \overline{p}) = 0$, and similarly for B. Again, A and B can be

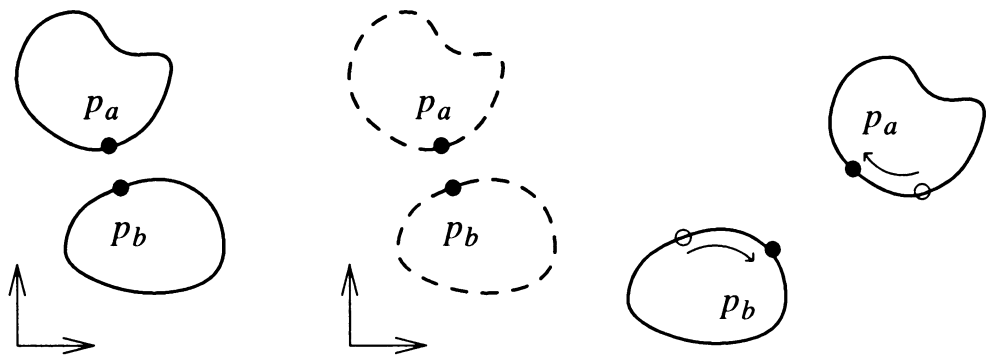


Figure 6.3: Change of the extremal points as a function of the configuration.

modeled equally well by parametric functions; Appendix A details the derivations for the parametric case. Partial derivatives of F and G with respect to $\overline{p}$ are easily computed, assuming partial derivatives of F and G with respect to p are known in some rest frame.

As in Section 5.2, the functions F and G are such that the surface normal of F,

$$\nabla F = \frac{\partial F}{\partial p},$$

is outwards pointing, and similarly for G. Thus, the function C can be written simply as

$$C(\overline{p}) = \nabla G(p_b, \overline{p}) \cdot (p_a - p_b) \tag{6–7}$$

where p_a and p_b are the extremal points for the configuration $\overline{p}$ (Figure 6.4). Note that $\nabla G(p_b, \overline{p})$ is always parallel to the vector $p_a - p_b$. Differentiating,

$$\frac{\partial C}{\partial \overline{p}}(\overline{p}) = \frac{\partial}{\partial \overline{p}} \nabla G(p_b, \overline{p}) \cdot (p_a - p_b) + \nabla G(p_b, \overline{p}) \cdot \left(\frac{\partial p_a}{\partial \overline{p}} - \frac{\partial p_b}{\partial \overline{p}} \right). \tag{6–8}$$

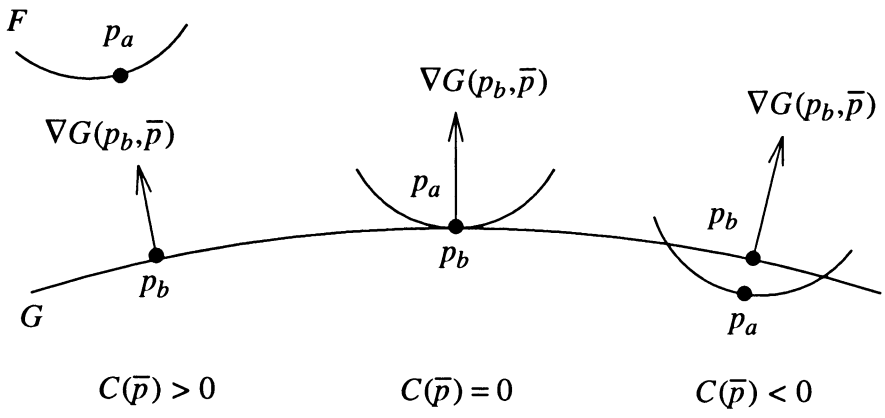


Figure 6.4: C expressed in terms of the extremal points.

Then

$$\begin{aligned}
\frac{\partial^2 C}{\partial \overline{p}^2}(\overline{p}) &= \frac{\partial}{\partial \overline{p}}\frac{\partial C}{\partial \overline{p}}(\overline{p}) \\
&= \frac{\partial^2}{\partial \overline{p}^2}\nabla G(p_b,\overline{p}) \cdot (p_a - p_b) + \frac{\partial}{\partial \overline{p}}\nabla G(p_b,\overline{p}) \cdot \left(\frac{\partial p_a}{\partial \overline{p}} - \frac{\partial p_b}{\partial \overline{p}}\right) \\
&\quad + \frac{\partial}{\partial \overline{p}}\nabla G(p_b,\overline{p}) \cdot \left(\frac{\partial p_a}{\partial \overline{p}} - \frac{\partial p_b}{\partial \overline{p}}\right) + \nabla G(p_b,\overline{p}) \cdot \left(\frac{\partial p_a}{\partial \overline{p}^2} - \frac{\partial p_b}{\partial \overline{p}^2}\right) \quad (6\text{–}9) \\
&= \frac{\partial^2}{\partial \overline{p}^2}\nabla G(p_b,\overline{p}) \cdot (p_a - p_b) + 2\frac{\partial}{\partial \overline{p}}\nabla G(p_b,\overline{p}) \cdot \left(\frac{\partial p_a}{\partial \overline{p}} - \frac{\partial p_b}{\partial \overline{p}}\right) \\
&\quad + \nabla G(p_b,\overline{p}) \cdot \left(\frac{\partial p_a}{\partial \overline{p}^2} - \frac{\partial p_b}{\partial \overline{p}^2}\right).
\end{aligned}$$

Clearly, to evaluate the second partial derivatives of C, we must be able to differentiate p_a and p_b in terms of the configuration $\overline{p}$. This requires a more constructive definition for p_a and p_b.

In Section 5.2.1, a set of necessary conditions was given for the extremal points. Because we are defining p_a and p_b only locally in this chapter, the conditions of Section 5.2.1 are both necessary and sufficient. That is, if we say that p_a and p_b

are the points near p_c that satisfy the conditions

$$\begin{cases} E_1: & \nabla F(p_a, \overline{p}) + \lambda_2 \nabla G(p_b, \overline{p}) = \mathbf{0} \\ E_2: & F(p_a, \overline{p}) = 0 \\ E_3: & G(p_b, \overline{p}) = 0 \\ E_4: & (p_b - p_a) + \lambda_1 \nabla G(p_b, \overline{p}) = \mathbf{0} \end{cases} \tag{6–10}$$

then this uniquely defines p_a and p_b in some neighborhood of p_c for a given configuration $\overline{p}$. Equation (6–10) is modified exactly as in Section 5.2.2 for the case of contact between a polyhedron and a curved surface. Constraint functions for contact between two polyhedra are easily formulated[12,13] if the contact involves either two non-parallel contacting edges, or a vertex in contact with the interior of a polygonal face; both these cases admit a well defined normal direction. However, contact between two vertices, a vertex and an edge, or two parallel edges do not yield a well defined normal and may be considered as *degenerate* contact situations. Palmer[42] discusses the complexity of simulations involving degenerate contacts; in general, degenerate contacts yield *NP*-hard simulation problems. We will assume that a normal direction is defined at all contact points.

We can use Equation (6–10) to differentiate p_a and p_b with respect to $\overline{p}$. Suppose we are given values for p_a and p_b for some configuration $\overline{p}$. (Typically, we wish to differentiate p_a and p_b for the configuration $\overline{x}_0$; for this configuration, $p_a = p_b = p_c$.) Given p_a and p_b, values for λ_1 and λ_2 are easily calculated. Using Equation (6–10), we can calculate $\partial p_a / \partial \overline{p}$ and $\partial p_b / \partial \overline{p}$ by making use of the implicit function theorem for simultaneous equations[49]. This theorem asserts that under proper conditions, p_a and p_b may be regarded as functions of the configuration $\overline{p}$; the theorem also gives a constructive expression for the derivatives of p_a and p_b. We will write the Jacobian determinant of a set of vector functions $H_1(x_1, \ldots, x_n)$

through $H_n(x_1, \ldots, x_n)$ as

$$\frac{\partial(H_1, \ldots, H_n)}{\partial(x_1, \ldots, x_n)} = \begin{vmatrix} \frac{\partial H_1}{\partial x_1} & \cdots & \frac{\partial H_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial H_n}{\partial x_1} & \cdots & \frac{\partial H_n}{\partial x_n} \end{vmatrix}.$$

Since p_a and p_b are points in $\mathbf{R}^3$, p_a and p_b are each three (scalar) functions of configuration: $p_{ax}(\overline{p})$, $p_{ay}(\overline{p})$ and $p_{az}(\overline{p})$ and similarly for p_b. The implicit function theorem, applied to Equation (6–10) yields

$$\frac{\partial p_{ax}}{\partial \overline{p}_i}(\overline{p}) = -\frac{K}{J} \tag{6–11}$$

where

$$\begin{aligned} K &= \frac{\partial(E_1, E_2, E_3, E_4)}{\partial(\overline{p}_i, p_{ay}, p_{az}, p_{bx}, p_{by}, p_{bz}, \lambda_1, \lambda_2)} \\ J &= \frac{\partial(E_1, E_2, E_3, E_4)}{\partial(p_{ax}, p_{ay}, p_{az}, p_{bx}, p_{by}, p_{bz}, \lambda_1, \lambda_2)} \end{aligned} \tag{6–12}$$

and $\overline{p}_i$ denotes the ith component of $\overline{p}$. Similar expressions give the derivatives for p_{ay}, p_{az}, p_{bx}, p_{by} and p_{bz}. The case when J is near-zero or zero is considered in Section 6.7.

Second partial derivatives for p_a and p_b can be calculated by differentiating Equation (6–11). For example,

$$\frac{\partial^2 p_{ax}}{\partial \overline{p}_i \, \partial \overline{p}_j}(\overline{p}) = -\frac{\partial}{\partial \overline{p}_j}\left(\frac{K}{J}\right). \tag{6–13}$$

Combining this result with Equation (6–9) yields an analytical expression for the second partial derivatives of C. Examining the dimensions of the equations involved however, both K and J are the determinants of 8×8 matrices. Differentiating K and J with respect to $\overline{p}$ requires computing the derivative of an 8×8 matrix determinant. This is extremely impractical. Although K and J have considerable block

structure, block structure cannot be exploited in computing matrix determinants. In its present form, the Jacobian determinants of K and J contain more than 1,000 terms of seven factors each; the derivatives contain far more terms. Additionally, each of the 72 different second partials of p_{ax}, p_{ay}, p_{az}, p_{bx}, p_{by}, and p_{bz} would need to be computed in this manner. Thus, although we have achieved an analytical expression for Equation (6–6), it is not practical.

6.4 Computing $\ddot{C}(\overline{x}(t_0))$ directly

In this section, we shall formulate C in a slightly different manner. Let us assume a rotated coordinate system in which $\nabla F(p_a, \overline{x}(t_0))$ and $\nabla G(p_b, \overline{x}(t_0))$ are colinear with the z axis, with $\nabla G(p_b, \overline{x}(t_0))$ directed positively along z. (We will employ the standard right-handed coordinate system used to depict functions $z = h(x, y)$, with z the vertical axis.) Next, we explicitly model A and B near p_c as scalar functions f and g of x and y and configuration $\overline{p}$. Where F was a function $F(x, y, z, \overline{p})$, f is a function $f(x, y, \overline{p})$ and similarly for g. The justification for the existence of f and g is the implicit function theorem. The formal definitions of f and g are given by

$$F(x, y, f(x, y, \overline{p}), \overline{p}) = 0 \qquad \text{and} \qquad G(x, y, g(x, y, \overline{p}), \overline{p}) = 0. \tag{6–14}$$

Thus, the point $(x, y, f(x, y, \overline{x}(t)))$ is a point on F at time t, and similarly for G. In this new coordinate system, the extremal points p_a and p_b share the same x and y coordinates at time t_0. At times t near t_0, A and B do not penetrate as long as the z value of p_a is greater or equal to the z value of p_b (Figure 6.5). We will use this to redefine C as

$$C(\overline{p}) = f(p_{ax}, p_{ay}, \overline{p}) - g(p_{bx}, p_{by}, \overline{p}) \tag{6–15}$$

where p_{ax}, p_{ay}, p_{bx}, and p_{by} are the extremal points for configuration $\overline{p}$.

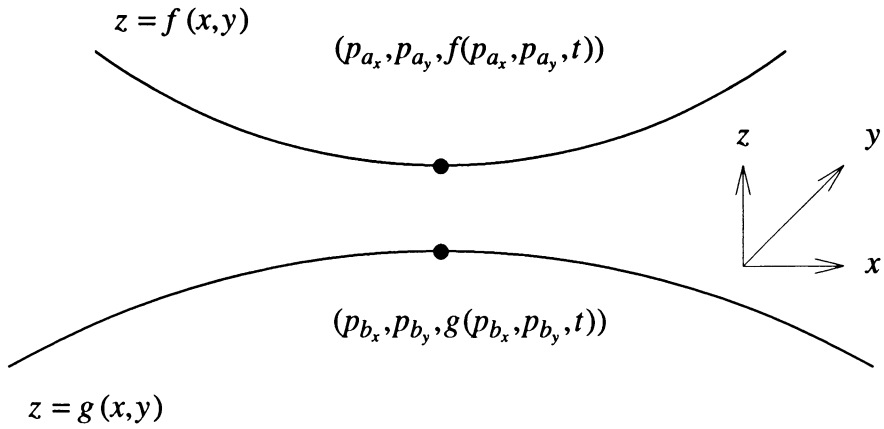


Figure 6.5: Side view of the implicit surfaces F and G expressed explicitly by functions f and g.

Since the z coordinates of the extremal points are no longer needed, let us rewrite Equation (6–10). Differentiating Equation (6–14) we obtain

$$\frac{\partial f}{\partial x} = -\frac{\dfrac{\partial F}{\partial x}}{\dfrac{\partial F}{\partial z}}, \quad \frac{\partial f}{\partial y} = -\frac{\dfrac{\partial F}{\partial y}}{\dfrac{\partial F}{\partial z}}, \qquad \text{and} \qquad \frac{\partial f}{\partial \overline{p}} = -\frac{\dfrac{\partial F}{\partial \overline{p}}}{\dfrac{\partial F}{\partial z}} \tag{6–16}$$

and similarly for g. Second derivatives of f and g are obtained by repeated differentiation of Equation (6–14). Using f and g, condition E_4 of Equation (6–10) may be written componentwise as

$$\begin{pmatrix} p_{bx} \\ p_{by} \\ g(p_{bx}, p_{by}, \overline{p}) \end{pmatrix} - \begin{pmatrix} p_{ax} \\ p_{ay} \\ f(p_{ax}, p_{ay}, \overline{p}) \end{pmatrix} + \lambda_1 \begin{pmatrix} \dfrac{\partial G}{\partial x}(p_b, \overline{p}) \\ \dfrac{\partial G}{\partial y}(p_b, \overline{p}) \\ \dfrac{\partial G}{\partial z}(p_b, \overline{p}) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \tag{6–17}$$

from which we obtain

$$\lambda_1 = \frac{f(p_{ax}, p_{ay}, \overline{p}) - g(p_{bx}, p_{by}, \overline{p})}{\dfrac{\partial G}{\partial z}(p_b, \overline{p})}. \tag{6–18}$$

Multiplying Equation (6–17) by -1 and using Equation (6–16) allows us to rewrite E_4 as

$$\begin{pmatrix} p_{ax} - p_{bx} \\ p_{ay} - p_{by} \end{pmatrix} + (f(p_{ax}, p_{ay}, \overline{p}) - g(p_{bx}, p_{by}, \overline{p})) \begin{pmatrix} \frac{\partial g}{\partial x}(p_{bx}, p_{by}, \overline{p}) \\ \frac{\partial g}{\partial y}(p_{bx}, p_{by}, \overline{p}) \end{pmatrix} = \mathbf{0}. \quad (6\text{–}19)$$

This new condition has one less equation than E_4 because of the reduction of variables from F to f. In a similar fashion, λ_2 of condition E_1 is eliminated and condition E_4 is rewritten as

$$\begin{pmatrix} \frac{\partial f}{\partial x}(p_{ax}, p_{ay}, \overline{p}) \\ \frac{\partial f}{\partial y}(p_{ax}, p_{ay}, \overline{p}) \end{pmatrix} - \begin{pmatrix} \frac{\partial g}{\partial x}(p_{bx}, p_{by}, \overline{p}) \\ \frac{\partial g}{\partial y}(p_{bx}, p_{by}, \overline{p}) \end{pmatrix} = \mathbf{0}. \quad (6\text{–}20)$$

Conditions E_2 and E_3 are no longer required since f and g give explicit definitions of the surfaces. At this point, it will be more useful to treat the extremal points as functions of time. We can define p_{ax}, p_{ay}, p_{bx} and p_{by} as the extremal points at time t by the system

$$\begin{cases} D_1: & \begin{pmatrix} \frac{\partial f}{\partial x}(p_{ax}, p_{ay}, \overline{x}(t)) \\ \frac{\partial f}{\partial y}(p_{ax}, p_{ay}, \overline{x}(t)) \end{pmatrix} - \begin{pmatrix} \frac{\partial g}{\partial x}(p_{bx}, p_{by}, \overline{x}(t)) \\ \frac{\partial g}{\partial y}(p_{bx}, p_{by}, \overline{x}(t)) \end{pmatrix} = \mathbf{0} \\ D_2: & (f(p_{ax}, p_{ay}, \overline{x}(t)) - g(p_{bx}, p_{by}, \overline{x}(t))) \begin{pmatrix} \frac{\partial g}{\partial x}(p_{bx}, p_{by}, \overline{x}(t)) \\ \frac{\partial g}{\partial y}(p_{bx}, p_{by}, \overline{x}(t)) \end{pmatrix} \\ & \qquad + \begin{pmatrix} p_{ax} - p_{bx} \\ p_{ay} - p_{by} \end{pmatrix} = \mathbf{0}. \end{cases} \quad (6\text{–}21)$$

Now p_{ax}, p_{ay}, p_{bx} and p_{by} can be differentiated as functions of time. The implicit function theorem for simultaneous equations gives the result

$$\dot{p}_{ax} = -\frac{\dfrac{\partial(D_1, D_2)}{\partial(t, p_{ay}, p_{bx}, p_{by})}}{J} \qquad \text{and} \qquad \dot{p}_{ay} = -\frac{\dfrac{\partial(D_1, D_2)}{\partial(p_{ax}, t, p_{bx}, p_{by})}}{J} \tag{6–22}$$

where

$$J = \frac{\partial(D_1, D_2)}{\partial(p_{ax}, p_{ay}, p_{bx}, p_{by})} \tag{6–23}$$

and

$$\frac{\partial f}{\partial t}(p_{ax}, p_{ay}, \overline{x}(t)) = \frac{\partial f}{\partial \overline{p}}(p_{ax}, p_{ay}, \overline{x}(t))^T \dot{\overline{x}}(t) \tag{6–24}$$

(and similarly for g). Similar expressions are obtained for $\dot{p}_{bx}$ and $\dot{p}_{by}$. The derivatives of p_{az} and p_{bz} are simply $\partial f/\partial t$ and $\partial g/\partial t$. Equation (6–23) redefines J, but we will not have any further use for the 8×8 Jacobian matrix in Equation (6–12). We will consider the case when J is near-zero or zero in Section 6.7.

Since D_1 and D_2 do not involve p_{az} or p_{bz}, let p_a and p_b henceforth denote the column vectors $(p_{ax}, p_{ay})^T$ and $(p_{bx}, p_{by})^T$ at time t. Similarly, let $\dot{p}_a = (\dot{p}_{ax}, \dot{p}_{ay})^T$ and $\dot{p}_b = (\dot{p}_{bx}, \dot{p}_{by})^T$ be the column vector derivatives of p_a and p_b at time t. The quantities $\partial f/\partial p$, $\partial f/\partial \overline{p}$ and $\dot{\overline{x}}(t)$ are also column vectors. From Equation (6–15), the condition $\ddot{C}(\overline{x}(t_0)) \geq 0$ is simply

$$\frac{\mathrm{d}^2}{\mathrm{dt}^2} f(p_a, \overline{x}(t_0)) - \frac{\mathrm{d}^2}{\mathrm{dt}^2} g(p_b, \overline{x}(t_0)) \geq 0. \tag{6–25}$$

Differentiating f with respect to time,

$$\frac{\mathrm{d}}{\mathrm{dt}} f(p_a, \overline{x}(t_0)) = \frac{\partial f}{\partial p}(p_a, \overline{x}(t_0))^T \dot{p}_a + \frac{\partial f}{\partial \overline{p}}(p_a, \overline{x}(t_0))^T \dot{\overline{x}}(t_0). \tag{6–26}$$

Then

$$\begin{aligned}
\frac{\mathrm{d}^2}{\mathrm{dt}^2} f(p_a, \overline{x}(t_0)) &= \left(\frac{\partial f}{\partial p^2}(p_a, \overline{x}(t_0))\dot{p}_a + \frac{\partial^2 f}{\partial p \partial \overline{p}}(p_a, \overline{x}(t_0))\dot{\overline{x}}(t_0) \right)^T \dot{p}_a \\
&\quad + \frac{\partial f}{\partial p}(p_a, \overline{x}(t_0))\ddot{p}_a \\
&\quad + \left(\frac{\partial^2 f}{\partial p \partial \overline{p}}(p_a, \overline{x}(t_0))\dot{p}_a + \frac{\partial f}{\partial \overline{p}^2}(p_a, \overline{x}(t_0))\dot{\overline{x}}(t_0) \right)^T \dot{\overline{x}}(t_0) \\
&\quad + \frac{\partial f}{\partial \overline{p}}(p_a, \overline{x}(t_0))^T \ddot{\overline{x}}(t_0) \qquad (6\text{–}27) \\
&= \dot{p}_a^T \frac{\partial f}{\partial p^2}(p_x, \overline{x}(t_0))\dot{p}_a + 2\dot{\overline{x}}(t_0)^T \frac{\partial^2 f}{\partial p \partial \overline{p}}(p_a, \overline{x}(t_0))\dot{p}_a \\
&\quad + \frac{\partial f}{\partial p}(p_a, \overline{x}(t_0))^T \ddot{p}_a + \dot{\overline{x}}(t_0)^T \frac{\partial f}{\partial \overline{p}^2}\dot{\overline{x}}(t_0) \\
&\quad + \frac{\partial f}{\partial \overline{p}}(p_a, \overline{x}(t_0))^T \ddot{\overline{x}}(t_0).
\end{aligned}$$

From Equation (6–16) and the fact that $\nabla F(p_{ax}, p_{ay}, \overline{x}(t_0))$ is colinear with the z axis,

$$\frac{\partial f}{\partial p}(p_a, \overline{x}(t_0)) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \qquad (6\text{–}28)$$

This yields

$$\begin{aligned}
\frac{\mathrm{d}^2}{\mathrm{dt}^2} f(p_a, \overline{x}(t_0)) &= \dot{p}_a^T \frac{\partial f}{\partial p^2}(p_a, \overline{x}(t_0))\dot{p}_a + 2\dot{\overline{x}}(t_0)^T \frac{\partial^2 f}{\partial p \partial \overline{p}}(p_a, \overline{x}(t_0))\dot{p}_a \\
&\quad + \dot{\overline{x}}(t_0)^T \frac{\partial f}{\partial \overline{p}^2}\dot{\overline{x}}(t_0) + \frac{\partial f}{\partial \overline{p}}\ddot{\overline{x}}(t_0). \qquad (6\text{–}29)
\end{aligned}$$

A similar expression is obtained for g. Thus, neither $\ddot{p}_a$ nor $\ddot{p}_b$ is required to calculate $\ddot{C}(\overline{x}(t_0))$. Equation (6–29) depends linearly on $\ddot{\overline{x}}(t_0)$, which in turn depends linearly upon the forces acting on A and B; this equation can be used to calculate the reaction force between A and B. Since $\dot{p}_a$ and $\dot{p}_b$ are simply calculated as the ratio of two 4×4 matrix determinants, the final expression is quite practical for use.

6.5 Predicting collision times

The primary use for the derivation of $\ddot{C}(\overline{x}(t_0))$ is to formulate the equations for the constraint forces at contact points. However, $\ddot{C}(\overline{x}(t_0))$ can also be used to help predict the time of a collision. When the collision/contact determination step detects inter-penetration between two bodies, an estimate must be made of when the two bodies collided. The work done by the simulator can be significantly decreased by increasing the accuracy of the prediction of the collision time.

Consider the case when inter-penetration is detected between two bodies at time t_1. Let $C(\overline{x}(t))$ represent the non-penetration constraint that has been violated in the vicinity of the inter-penetration.[2] Since the constraint is not satisfied at time t_1, we have $C(\overline{x}(t_1)) < 0$. However, if the previous time step (before the inter-penetration) occurred at time t_0, then $C(\overline{x}(t_0)) > 0$. Computing the collision time is thus a root finding problem, on some known interval: we are looking for a time t_c, with $t_0 < t_c < t_1$ such that $C(\overline{x}(t_c)) = 0$.

Since we can easily calculate derivatives of first and second derivatives of C, we model $C(\overline{x}(t))$ quadratically near t_0 as

$$C(\overline{x}(t)) = C(\overline{x}(t_0)) + \dot{C}(\overline{x}(t_0))(t - t_0) + \ddot{C}(\overline{x}(t_0))(t - t_0)^2 + O((t - t_0)^3). \quad (6\text{–}30)$$

Ignoring the $O((t - t_0)^3)$ term, we solve for $C(\overline{x}(t_c)) = 0$ to obtain the estimation

$$t_c = t_0 + \frac{-\dot{C}(\overline{x}(t_0)) \pm \sqrt{\dot{C}(\overline{x}(t_0))^2 - 2\ddot{C}(\overline{x}(t_0))C(\overline{x}(t_0))}}{\ddot{C}(\overline{x}(t_0))}. \quad (6\text{–}31)$$

[2] It is not always clear exactly which constraint this should be. For example, if a vertex from polyhedron A is inside polyhedron B, which face of B did it pass through to get there? In cases like this, the most likely non-penetration constraint should be chosen. For most situations, the correct non-penetration constraint is easy to determine. If several non-penetration constraints appear to have been violated, for example, if multiple vertices of A lie inside B, the non-penetration constraint that appears to have been violated the earliest should be chosen. For situations involving multiple constraint violations, an incorrect choice is not fatal—at worst, it will fail to yield a good estimate for the collision time.

The estimate t_c should be the smallest real root of Equation (6–31) greater than t_0; if no such root exists, or $\ddot{C}(\overline{x}(t_0))$ is near zero, t_c can be estimated using a linear approximation of $C(\overline{x}(t))$. The method is made robust by incorporating a bisection step when convergence is slow[15,44]; that is, predicting $t_c = (t_0 + t_1)/2$. For constant acceleration, Equation (6–31) gives an exact result for t_c, if $C(\overline{x}(t_0))$ is a linear measure of distance.

6.6 Rolling contact

As we have noted, $\ddot{C}(\overline{x}(t_0))$ is a measure of relative acceleration normal to the contact surfaces of A and B. In Chapter 9, when friction is discussed, we will need a description of both the relative slip velocity between A and B tangent to the contact surface, and the derivative of the slip velocity with respect to time. These quantities are expressed more naturally in world space than configuration space. The derivation of these quantities is straightforward for bodies of any geometry, and is certainly well known[4,41], but we will derive it here for completeness. We will also show why the constraint of pure rolling motion between bodies of arbitrary geometry is so much simpler to express than the non-penetration constraint we have just developed.

Let v_a, v_b, ω_a and ω_b denote the linear and angular velocities of A and B at time t_0. Let c_a and c_b denote the centers of mass of A and B. Then $\dot{c}_a = v_a$ and $\dot{c}_b = v_b$. Assume that A and B do not break contact near p_c. The slip velocity, s, between A and B at time t_0 is therefore

$$s = (v_a + \omega_a \times (p_c - c_a)) - (v_b + \omega_b \times (p_c - c_b)). \tag{6–32}$$

The derivative of the slip velocity is

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}s &= \frac{\mathrm{d}}{\mathrm{dt}}v_a + \frac{\mathrm{d}}{\mathrm{dt}}\omega_a \times (p_c - c_a) + \omega_a \times (\dot{p}_c - v_a) \\ &\quad - (\frac{\mathrm{d}}{\mathrm{dt}}v_b + \frac{\mathrm{d}}{\mathrm{dt}}\omega_b \times (p_c - c_b) + \omega_b \times (\dot{p}_c - v_b)). \qquad (6\text{–}33)\end{aligned}$$

Assuming that $s = \mathbf{0}$, that is, that A and B are not initially slipping, A and B can be made to maintain $s = \mathbf{0}$, and thus roll without slipping, by enforcing

$$\frac{\mathrm{d}}{\mathrm{dt}}s = \mathbf{0}. \qquad (6\text{–}34)$$

Since $\dot{p}_c$ is fairly easily calculated (by the previous section), the constraint that two bodies roll without slipping is easily derived as a constraint on the forces acting on A and B. Note that this constraint automatically enforces non-penetration between A and B. In order for A and B to inter-penetrate, they must attain some non-zero relative velocity normal to the contact surfaces; the constraint $\frac{\mathrm{d}}{\mathrm{dt}}s = \mathbf{0}$ prevents this from occurring. Why should this rolling without slipping constraint be so much simpler to express than simply the non-penetration constraint?

The motion of rolling without slipping between two bodies is a highly constrained motion. Equation (6–34) enforces three constraints on the relative acceleration, while Equation (6–4) enforces only one constraint. Thus, rolling without slipping has fewer degrees of freedom than simply moving without inter-penetrating. As a result, we have more knowledge about the motions of A and B, and the constraint is formulated much more easily.

6.7 Ill-conditioned Jacobians

As stated in Chapter 3, impact is assumed to have been dealt with prior to the computation of constraint forces. However, what has really been dealt with are situations where A and B have a non-zero approach velocity at p_c normal

to the contact surfaces. This is not the only case in which impact can arise. Featherstone[13] gives an example where discontinuities in the curvature of the contact surfaces require an impulse to be applied, even though the approach velocity normal to the contact surfaces is zero. Given the equations developed to compute $\ddot{C}$ for curved surfaces, we can add yet another type of impact that might occur between bodies.

Suppose we wish to simulate the frictionless oscillation of an ellipsoid A upon a plane surface B (Figure 6.6). Then the function F describing A is a simple quadratic function with non-zero curvature everywhere. B is described by a linear function G and has zero curvature everywhere. By solving Equation (6–25) for the reaction force between A and B, and considering Equation (6–29), it can be shown that for given initial conditions, the reaction force magnitude R is of the form

$$R(t) = O(\|\dot{p}_c(t)\| + 1) \tag{6–35}$$

over the simulation, treating p_c as a function of time.

Recall from Equation (6–22) that $\dot{p}_c$ is calculated as the ratio of two determinants. Let us calculate the denominator determinant J at a time t_0 when A and B are in contact (and thus $f(p_{ax}, p_{ay}, \overline{x}(t_0)) - g(p_{bx}, p_{by}, \overline{x}(t_0)) = 0$). Evaluating Equation (6–23), and since G (and thus g) have second partial derivatives everywhere,

$$J = \begin{vmatrix} f_{11} & f_{12} & 0 & 0 \\ f_{12} & f_{22} & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 \end{vmatrix} = f_{11}f_{22} - f_{12}^2 = \kappa \tag{6–36}$$

where

$$f_{11} = \frac{\partial f}{\partial x^2}(p_{ax}, p_{ay}, \overline{x}(t_0)), \quad f_{22} = \frac{\partial f}{\partial y^2}(p_{ax}, p_{ay}, \overline{x}(t_0)), \tag{6–37}$$

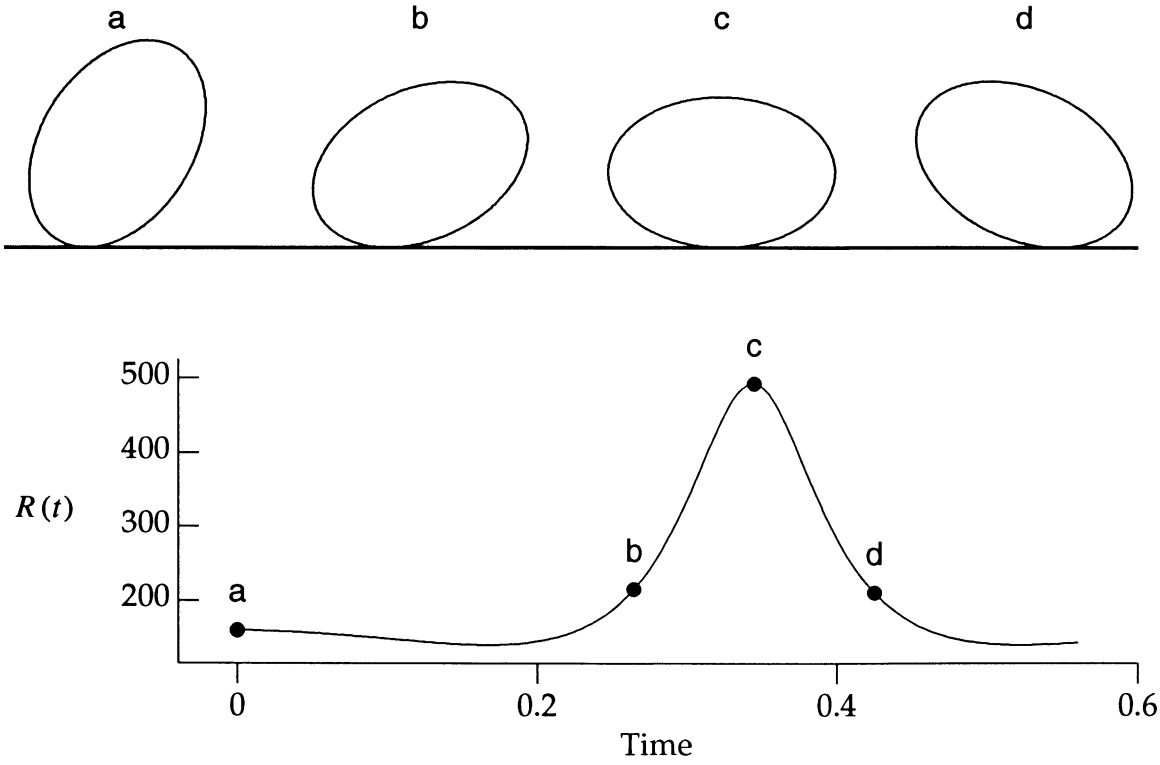


Figure 6.6: Frictionless oscillation of an ellipsoid A on a plane B.

and

$$f_{12} = \frac{\partial^2 f}{\partial x \partial y}(p_{ax}, p_{ay}, \overline{x}(t_0)). \tag{6–38}$$

This gives $J = \kappa$ where κ is the curvature of the ellipsoid at the contact point p_c. For the case of an ellipsoid, the curvature κ is always positive, so J is always non-zero. However, we can make κ arbitrarily close to zero by scaling the ellipsoid as shown in Figure 6.7. As the curvature at the contact point decreases, the reaction force increases, in a continuous fashion. The simulator must take small steps in order to accurately sample how the reaction force R changes over time. However, this is merely a practical problem of simulation. Mathematically speaking, nothing unusual occurs.

Suppose however that we enable A to have zero curvature at certain points. We can do this by letting A be a superquadric ellipsoid, defined in some rest frame as the set of points p satisfying

$$p_x^4 + p_y^4 + p_z^4 = 1. \tag{6–39}$$

Again, we will let A undergo frictionless oscillation in contact with B. Suppose that a point of zero curvature on A comes into contact with B at time t_0. As t approaches t_0, the value of J approaches zero, in a continuous fashion. As a result, $R(t)$ increases in a continuous fashion without bound, as t approaches t_0. In the limit, the reaction force $R(t)$ approaches infinity as the contact point approaches the zero curvature spot of A (Figure 6.8). Should this be considered as an impulsive reaction between A and B?

The answer to this is "almost". Clearly, the reaction force between A and B at t_0 cannot be considered to be a force, since it is infinite. However, in order for the reaction force to behave impulsively, it must cause a velocity discontinuity at time t_0. Even though the reaction force is unbounded, it does not cause such a discontinuity in velocity, although it does cause a discontinuity in acceleration.

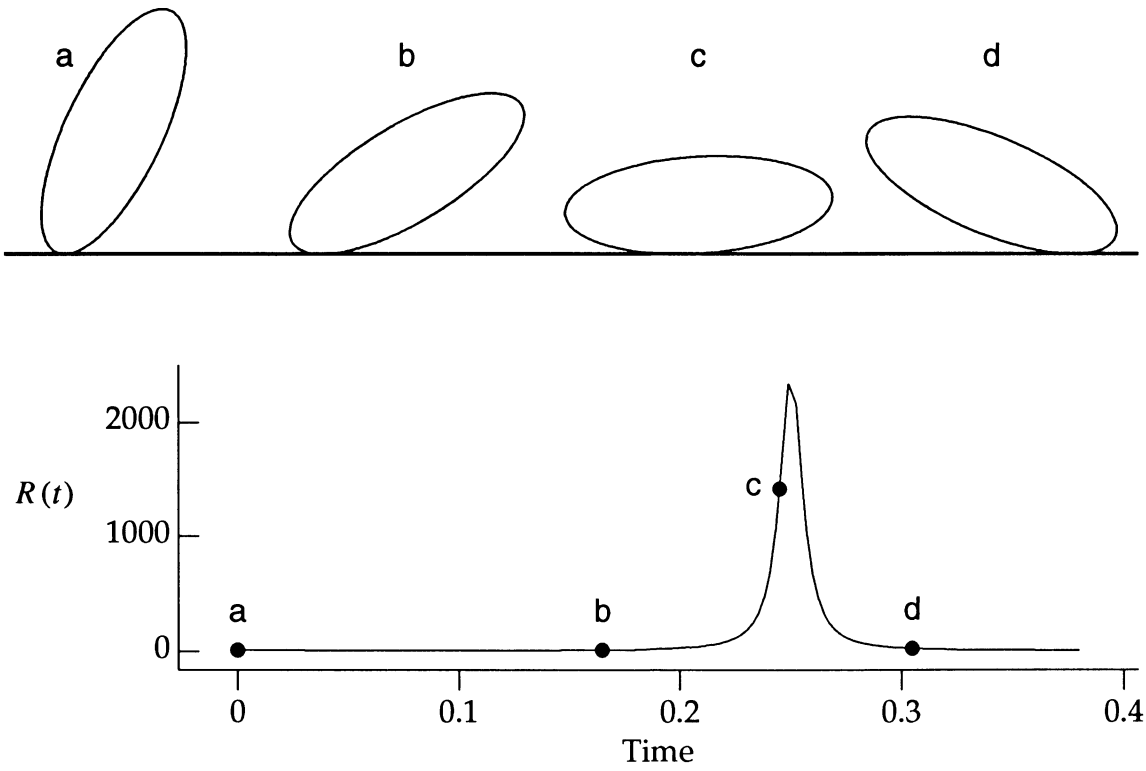


Figure 6.7: Frictionless oscillation of a more eccentric ellipsoid A on a plane B. The reaction force $R(t)$ is stronger and more sharply peaked the closer the curvature κ of A approaches zero.

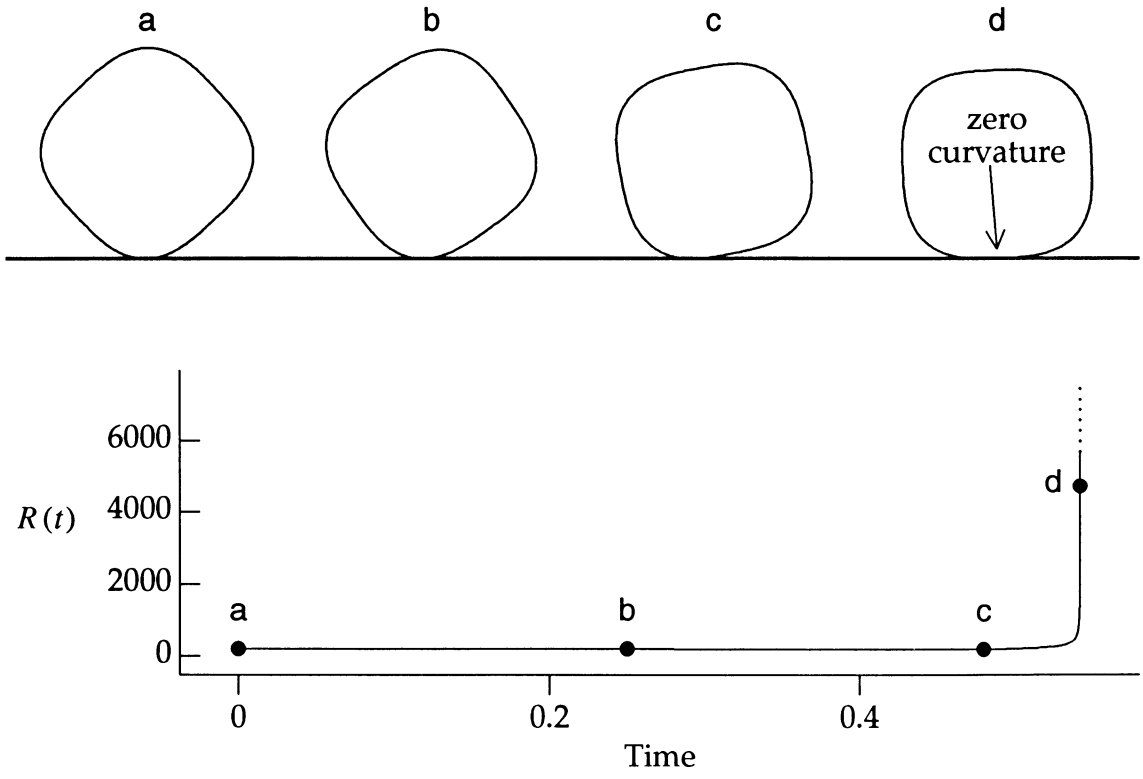


Figure 6.8: Frictionless oscillation of a superquadric ellipsoid A on a plane B. The reaction force $R(t)$ diverges as the contact point approaches a zero curvature spot on the superquadric ellipsoid A.

That is, the reaction force at time t_0 is an infinite force acting for an instant of time, but in such a way that the measured impulse is zero. To see this, consider the integral of the reaction force R over time. In order for the reaction force to behave as an impulse, it is necessary that

$$\lim_{\Delta t \to 0} \int_{t_0-\Delta t}^{t_0+\Delta t} R(t)\, dt > 0. \tag{6–40}$$

Since $R(t) = O(\|\dot{p}_c(t)\| + 1)$, consider

$$\lim_{\Delta t \to 0} \int_{t_0-\Delta t}^{t_0+\Delta t} O(\|\dot{p}_c(t)\| + 1)\, dt.$$

Since this is an integral over time involving $\dot{p}_c(t)$, by the fundamental theorem of calculus there exists a continuous function H of p_c such that

$$\int_{t_0-\Delta t}^{t_0+\Delta t} O(\|\dot{p}_c(t)\| + 1)\, dt = H(p_c(t_0 + \Delta t)) - H(p_c(t_0 - \Delta t)). \tag{6–41}$$

Since $p_c(t)$ is a spatial measure (the position of the contact point at time t), $p_c(t)$ is clearly continuous. Therefore,

$$\begin{aligned} \lim_{\Delta t \to 0} \int_{t_0-\Delta t}^{t_0+\Delta t} R(t)\, dt &= \lim_{\Delta t \to 0} \int_{t_0-\Delta t}^{t_0+\Delta t} O(\|\dot{p}_c(t)\| + 1)\, dt \\ &= \lim_{\Delta t \to 0} H(p_c(t + \Delta t)) - H(p_c(t - \Delta t)) \qquad (6–42) \\ &= H(p_c(t_0)) - H(p_c(t_0)) \\ &= 0. \end{aligned}$$

Thus, the reaction force is not impulsive. Viewed in another way, the reaction force behaves in the limit as an impulse of zero magnitude. For want of a better term, we could call such behavior a "zeroeth order" impact.

In practical terms, the problem of a zeroeth order impact is how to evaluate the integral of the force over a time period containing the impact. Current ordinary

differential equation solvers are not well suited for evaluating this integral, even though it is finite. From a sampling viewpoint, the reaction force must be sampled infinitely often near time t_0 to accurately evaluate the integral of the reaction force. One approach is to analytically evaluate the integral in a region of time bounding t_0, but it is unknown how to do this efficiently in a simulation environment. The simulation process described in this thesis does not deal with zeroeth order impacts.

Chapter 7

The Penalty Method

The remainder of this thesis will be devoted to the problem of computing the forces that arise at contact points for systems with multiple contact points. In this chapter, we will examine the penalty method for computing constraint forces between bodies, in the absence of friction. Since we already know that the penalty method usually performs poorly in practice, why bother to consider it at all? The main reason for considering this method is that in Chapter 12, we shall consider a model of frictional behavior that is based on the penalty method. Prior to this, we need to consider whether the penalty method for frictionless systems yields physically correct motion. Validating the penalty method for the frictionless case does not necessarily validate the model of frictional behavior proposed in Chapter 12, but it would clearly be unreasonable to extrapolate from the penalty method if it were known to be flawed for the case of frictionless systems. A secondary reason for considering the penalty method is to gain insights into its correctness as a simulation method; considering the large amount of simulation work that uses the penalty method, this is a topic worth pursuing.

7.1 Equality constrained motion

To simplify matters, we will first consider the penalty method as applied to equality constrained dynamics problems. This is actually not a large restriction, because systems with inequality constraints can be viewed as being subject to the corresponding equality constraints on any time interval during which no constraint changes from active to inactive or vice versa. For example, in the particle/surface example of Section 2.5, the motion of the particle, whenever it is in contact with the surface, is exactly the same as if it were constrained to always lie on the surface. Thus, we would first like to study the motion produced by the penalty method for equality constrained problems, such as the problem described in Section 2.4.

Rubin and Ungar[45] give an excellent proof that the penalty method does indeed yield (in the limit) physically correct behavior for frictionless systems with equality constraints. However, their proof assumes that the external forces acting on bodies are due to the potential energy of the system, rather than being arbitrary functions of time. Modifying Rubin and Ungar's proof to handle arbitrary time-dependent external forces does not seem overly difficult, but their proof is fairly lengthy and somewhat complicated, and proves a much stronger result than we are interested in. We will show that the penalty method produces (in the limit) physically correct motion by comparing differential descriptions of the physically correct motion and the motion produced by the penalty method. This is much simpler and more intuitive than Rubin and Ungar's approach; note however that Rubin and Ungar obtain a much stronger result about the limiting motion of the penalty method than can be achieved using the simple differential comparison.

Let us consider a system with equality constraints from time t_0 to t_1. In the previous chapter, the non-penetration constraint at a single contact point was represented by a hypersurface in C-space, and the configuration $\overline{x}(t)$ was required to

lie on or "above" this hypersurface. In restricting ourselves to equality constraints, we are requiring $\overline{x}(t)$ to lie on the hypersurface. For a system with k bodies and n contact points, the n non-penetration constraints are represented as n hypersurfaces in a $6k$-dimensional C-space. The configuration point $\overline{x}(t)$ must lie on all n hypersurfaces. Equivalently, $\overline{x}(t)$ must lie in the *intersection* of all the constraint hypersurfaces. Since the constraint hypersurfaces are manifolds, the intersection of the constraint hypersurfaces is itself a manifold, which can be expressed in terms of a scalar function $C_I(\overline{p})$ as $C_I(\overline{p}) = 0$. We shall call this manifold the *intersection manifold.*

7.1.1 The physically correct motion

We will use Lagrangian dynamics to derive the equation of motion for a system constrained to satisfy $C_I(\overline{x}(t)) = 0$ from time t_0 to t_1. The remainder of this chapter assumes a knowledge of Lagrangian dynamics and differentiable manifolds.

In order to describe the motion of the system (that is, the function $\overline{x}(t)$ from t_0 to t_1), we will have to describe the configuration state $\overline{x}(t)$, and velocity $\dot{\overline{x}}(t)$, in terms of a coordinate system on the intersection manifold. Let the intersection manifold have dimension d, and let $S(q)$ be a parameterization of the manifold. The function $S(q)$ invertibly maps points from some region of $\mathbf{R}^d$ to the intersection manifold. (If the manifold cannot be covered with a single parameterization, the motion of $\overline{x}(t)$ can be considered in suitably small intervals.) The motion of $\overline{x}(t)$, restricted to lie on the intersection manifold is then described in terms of a continuous path $q(t)$ in $\mathbf{R}^d$. Given initial values for $\overline{x}(t_0)$ and $\dot{\overline{x}}(t_0)$, the initial conditions $q(t_0)$ and $\dot{q}(t_0)$ must satisfy

$$S(q(t_0)) = \overline{x}(t_0) \qquad \text{and} \qquad \frac{\mathrm{d}}{\mathrm{dt}} S(q(t_0)) = dS_{q(t_0)}\dot{q}(t_0) = \dot{\overline{x}}(t_0) \qquad (7\text{–}1)$$

where

$$dS_{q(t_0)} = \frac{\partial S}{\partial q}(q(t_0)). \tag{7–2}$$

The function $dS_{q(t)}$ maps vectors from $\mathbf{R}^d$ to vectors that are tangent to the intersection manifold at $S(q(t))$.

We also need to describe the kinetic energy of the system. Given a system with configuration $\overline{p}$ in C-space and velocity $\overline{v}$, the kinetic energy T of the system can be written in the form

$$T = \tfrac{1}{2}\overline{v}^T M(\overline{p})\overline{v} \tag{7–3}$$

where $M(\overline{p})$ is a square matrix with size equal to the dimension of C-space. The matrix $M(\overline{p})$ is symmetric positive definite. The kinetic energy is used to induce an inner product on C-space; given vectors $\overline{u}$ and $\overline{w}$ at point $\overline{p}$ of C-space, the inner product of $\overline{u}$ and $\overline{w}$ is written $<\overline{u}, \overline{w}>_{\overline{p}}$ and is defined as

$$<\overline{u}, \overline{w}>_{\overline{p}} = <\overline{w}, \overline{u}>_{\overline{p}} = \overline{u}^T M(\overline{p})\overline{w}. \tag{7–4}$$

Using the inner product, T can be written as

$$T = \tfrac{1}{2}<\overline{v}, \overline{v}>_{\overline{p}}. \tag{7–5}$$

Expressing the kinetic energy T in terms of the parametric coordinates of $\overline{x}(t)$, we have

$$\begin{aligned} T &= \tfrac{1}{2}<\dot{\overline{x}}(t), \dot{\overline{x}}(t)>_{\overline{x}(t)} \\ &= \tfrac{1}{2}<\frac{\mathrm{d}}{\mathrm{dt}}S(q(t)), \frac{\mathrm{d}}{\mathrm{dt}}S(q(t))>_{S(q(t))} \\ &= \tfrac{1}{2}<dS_{q(t)}\dot{q}(t), dS_{q(t)}\dot{q}(t)>_{S(q(t))}. \end{aligned} \tag{7–6}$$

(Henceforth, all evaluations of dS will take place at $q(t)$, so we shall simply write dS in place of $dS_{q(t)}$. Likewise, all inner products are assumed to be evaluated at

$S(q(t))$ unless otherwise qualified.) Given any external C-space force $F(t)$ acting on the system, let $Q(t)$ be the equivalent generalized force on $q(t)$. If we assume that the potential energy of the system is always constant, since $Q(t)$ can include the effects of forces due to potentials, the equation of motion for $q(t)$ is given by

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial T}{\partial \dot{q}}\right) - \frac{\partial T}{\partial q} = Q. \tag{7–7}$$

Using Equation (7–6), this yields

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial}{\partial \dot{q}}\left(\tfrac{1}{2}<dS\dot{q}(t), dS\dot{q}(t)>\right)\right) - \frac{\partial}{\partial q}\left(\tfrac{1}{2}<dS\dot{q}(t), dS\dot{q}(t)>\right) = Q(t) \tag{7–8}$$

with initial conditions for $q(t_0)$ and $\dot{q}(t_0)$ as specified by Equation (7–1).

7.1.2 Motion due to the penalty method

Next, we will derive an equation of motion for the penalty method, which will then be compared with Equation (7–8). As defined in Section 2.4, the penalty method works by imposing a penalty force on the system when $\overline{x}(t)$ deviates from the intersection manifold. The penalty force can be viewed as the gradient of a potential energy function $P(\overline{p})$ that increases the farther $\overline{p}$ lies from the intersection manifold. Thus, the penalty force is conservative, and does not change the overall energy of the system. If the potential energy function P is made steep enough, then for a conservative system, $\overline{x}(t)$ can never deviate more than a certain distance from the intersection manifold; that distance is determined by the total energy of the system. Thus, for the case when the only external forces acting on the system are due to the potential energy of the system, increasing the penalty force (by increasing the steepness of P) binds $\overline{x}(t)$ closer to the intersection manifold.

However, if time-dependent, and thus, non-conservative, external forces act on the system, increasing the slope of P does not automatically limit the deviation of $\overline{x}(t)$ from the intersection manifold. If the external force varies over time, it is

possible that a steeper P may cause a "resonance", and allow $\overline{x}(t)$ to deviate more from the intersection manifold. For example, the amount of work done by a given external driving force on a linear spring system is not monotonic with respect to the spring constant. In general though, as the spring constant is increased, the driving force does less work. Similarly, in analyzing the penalty method, we will assume that the maximum deviation of $\overline{x}(t)$ from the intersection manifold can be made arbitrarily small, by increasing the steepness of P in such a way that resonance is avoided. We would like to show that as the maximum deviation decreases, the path traced in the vicinity of the intersection manifold converges to the physically correct path lying exactly on the intersection manifold.

In the previous section, the position of $\overline{x}(t)$ on the intersection manifold was described by the parametric coordinate $q(t)$. For the penalty method, we need to describe $\overline{x}(t)$ in terms of a coordinate $q(t)$ on the intersection manifold, and a displacement from the intersection manifold. To accomplish this, we will describe a coordinate system for the orthogonal complement of the intersection manifold. Let the orthogonal complement of the intersection manifold have dimension k. At each point $\overline{p}$ on the manifold, we define vectors $N_1(\overline{p})$ through $N_k(\overline{p})$ such that

$$<N_i(\overline{p}), N_i(\overline{p})>_{\overline{p}} = 1 \qquad (1 \leq i \leq k) \tag{7–9}$$

and

$$<N_i(\overline{p}), N_j(\overline{p})>_{\overline{p}} = 0 \qquad (1 \leq i < j \leq k). \tag{7–10}$$

If $\overline{u}$ is tangent to the intersection manifold at $\overline{p}$, then

$$<N_i(\overline{p}), \overline{u}>_{\overline{p}} = 0 \qquad (1 \leq i \leq k); \tag{7–11}$$

that is, each $N_i(\overline{p})$ is normal to the intersection manifold at $\overline{p}$, as defined by the inner product induced by the kinetic energy. Each $N_i(\overline{p})$ varies smoothly

over the intersection manifold; since we are considering the motion of $\overline{x}(t)$ locally, orientability of the intersection manifold is not a concern.

Given this orthogonal coordinate system, we will express points $\overline{x}(t)$ lying near the intersection manifold using the parametric coordinate $q(t)$ and a k-tuple of real numbers $(h_1(t), \ldots, h_n(t))$ as

$$\overline{x}(t) = S(q(t)) + \sum_{i=1}^{k} h_i(t) N_i(S(q(t))). \tag{7–12}$$

This coordinate system is one-to-one in some region of C-space properly containing the intersection manifold, as long as the curvature of the manifold is bounded. Since we can force $\overline{x}(t)$ to lie as close to the intersection manifold as necessary, Equation (7–12) is an acceptable way of defining the position of $\overline{x}(t)$.

Using the notation of Equation (7–12), it is simple to write an equation of motion for the penalty method. Since the values $h_i(t)$ describe the displacement of $\overline{x}(t)$ from the constraint manifold, we can write the potential energy of the penalty function in terms of $h_i(t)$. The assumption is that for any $\epsilon > 0$, a potential energy function $P(h_1(t), \ldots, h_k(t))$ can be chosen[1] so that $|h_i(t)| < \epsilon$ for all $1 \leq i \leq k$ and $t_0 \leq t \leq t_1$.

The kinetic energy T' of the system, as in Equation (7–6), is given by

$$T' = \tfrac{1}{2} < \dot{\overline{x}}(t), \dot{\overline{x}}(t) >. \tag{7–13}$$

From Equation (7–12),

$$\dot{\overline{x}}(t) = dS\dot{q}(t) + \sum_{i=1}^{k} \dot{h}_i(t) N_i(S(q(t))) + \sum_{i=1}^{k} h_i(t) \dot{N}_i(S(q(t))) \tag{7–14}$$

where $\dot{N}_i(S(q(t))) = \frac{\mathrm{d}}{\mathrm{dt}} N_i(S(q(t)))$. Then the kinetic energy can be written as

[1] This does not mean to imply that such a choice is simple—it's not. The choice of a suitably strong penalty forces usually requires great care. This is simply one of the penalty method's many problems.

$$
\begin{aligned}
T' &= \tfrac{1}{2}<\dot{\bar{x}}(t), \dot{\bar{x}}(t)> \\
&= \tfrac{1}{2}<dS\dot{q}(t) + \left(\sum_{i=1}^{k} \dot{h}_i(t) N_i(S(q(t)))\right) + \left(\sum_{i=1}^{k} h_i(t) \dot{N}_i(S(q(t)))\right), \\
&\qquad dS\dot{q}(t) + \left(\sum_{i=1}^{k} \dot{h}_i(t) N_i(S(q(t)))\right) + \left(\sum_{i=1}^{k} h_i(t) \dot{N}_i(S(q(t)))\right)> \\
&= \tfrac{1}{2}\Big(<dS\dot{q}(t), dS\dot{q}(t)> \qquad (7\text{–}15) \\
&\qquad + <\sum_{i=1}^{k} \dot{h}_i(t) N_i(S(q(t))), \sum_{i=1}^{k} \dot{h}_i(t) N_i(S(q(t)))> \\
&\qquad + <\sum_{i=1}^{k} h_i(t) \dot{N}_i(S(q(t))), \sum_{i=1}^{k} h_i(t) \dot{N}_i(S(q(t)))>\Big) \\
&\quad + <dS\dot{q}(t), \sum_{i=1}^{k} \dot{h}_i(t) N_i(S(q(t)))> + <dS\dot{q}(t), \sum_{i=1}^{k} h_i(t) \dot{N}_i(S(q(t)))> \\
&\quad + <\sum_{i=1}^{k} \dot{h}_i(t) N_i(S(q(t))), \sum_{i=1}^{k} h_i(t) \dot{N}_i(S(q(t)))>.
\end{aligned}
$$

We can simplify this expression for T' by noting that the vectors $N_1(S(q(t)))$ through $N_k(S(q(t)))$ are orthonormal with respect to the inner product. Thus

$$
<\sum_{i=1}^{k} \dot{h}_i(t) N_i(S(q(t))), \sum_{i=1}^{k} \dot{h}_i(t) N_i(S(q(t)))> = \sum_{i=1}^{k} \dot{h}_i(t)^2. \qquad (7\text{–}16)
$$

Also, since the vector $dS\dot{q}(t)$ is tangent to the intersection manifold at $q(t)$ for any $\dot{q}(t)$, the inner product $<dS\dot{q}(t), N_i(S(q(t)))>$ is also zero for any $1 \leq i \leq k$ and any $\dot{q}(t)$. Thus,

$$
<dS\dot{q}(t), \sum_{i=1}^{k} \dot{h}_i(t) N_i(S(q(t)))> = 0. \qquad (7\text{–}17)
$$

Using these facts, we may rewrite Equation (7–15) as

$$
\begin{aligned}
T' &= \tfrac{1}{2}\Big(<dS\dot{q}(t), dS\dot{q}(t)> + \sum_{i=1}^{k} \dot{h}_i(t)^2 \\
&\qquad + <\sum_{i=1}^{k} h_i(t) \dot{N}_i(S(q(t))), \sum_{i=1}^{k} h_i(t) \dot{N}_i(S(q(t)))>\Big) \qquad (7\text{–}18)
\end{aligned}
$$

$$+ <dS\dot{q}(t), \sum_{i=1}^{k} h_i(t)\dot{N}_i(S(q(t)))>$$
$$+ <\sum_{i=1}^{k} \dot{h}_i(t)N_i(S(q(t))), \sum_{i=1}^{k} h_i(t)\dot{N}_i(S(q(t)))>.$$

The equation of motion for $q(t)$ is

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial}{\partial \dot{q}}\left(T' - P\right)\right) - \frac{\partial}{\partial q}\left(T' - P\right) = Q, \tag{7–19}$$

with the same initial conditions of the physically correct motion specified by Equation (7–1). Since the penalty energy function P is independent of both $q(t)$ and $\dot{q}(t)$, we can express Equation (7–19) as

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial T'}{\partial \dot{q}}\right) - \frac{\partial T'}{\partial q} = Q. \tag{7–20}$$

This equation is independent of the penalty force $-\nabla P$; as expected, the penalty force imparts no "tangential" acceleration to $\overline{x}(t)$, along the intersection manifold.

All that remains is to show that the motion of $q(t)$ according to Equation (7–19) is the same as the motion according to Equation (7–8) in the limit as $h_i(t) \rightarrow 0$ for each $h_i(t)$. Since the physically correct motion and the penalty method motion both have the same initial conditions, all we need to do is show that as the $h_i(t)$s converges to zero, Equation (7–19) and Equation (7–8) yield the same differential equation.

First, consider $\partial T'/\partial \dot{q}$. Since we are not differentiating with respect to h_i, any terms containing $h_i(t)$ drop out as $h_i \rightarrow 0$. Differentiating Equation (7–18) with respect to $\dot{q}$, all that remains is

$$\begin{aligned} \frac{\partial T'}{\partial \dot{q}} &= \frac{\partial}{\partial \dot{q}}\left(\tfrac{1}{2}<dS\dot{q}(t), dS\dot{q}(t)> + \sum_{i=1}^{k} \dot{h}_i(t)^2\right) \\ &= \frac{\partial}{\partial \dot{q}}\left(\tfrac{1}{2}<dS\dot{q}(t), dS\dot{q}(t)>\right). \end{aligned} \tag{7–21}$$

Now, consider $\partial T'/\partial q$. Again, any terms containing $h_i(t)$ drop out, and we have

$$\begin{aligned}\frac{\partial T'}{\partial q} &= \frac{\partial}{\partial q}\left(\tfrac{1}{2}<dS\dot{q}(t), dS\dot{q}(t)> + \sum_{i=1}^{k} \dot{h}_i(t)^2\right) \\ &= \frac{\partial}{\partial q}\left(\tfrac{1}{2}<dS\dot{q}(t), dS\dot{q}(t)>\right). \end{aligned} \tag{7–22}$$

Thus, as the $h_i(t)$s converge to zero, the equation of motion

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial T'}{\partial \dot{q}}\right) - \frac{\partial T'}{\partial q} = Q \tag{7–23}$$

becomes

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial}{\partial \dot{q}}\left(\tfrac{1}{2}<dS\dot{q}(t), dS\dot{q}(t)>\right)\right) - \frac{\partial}{\partial q}\left(\tfrac{1}{2}<dS\dot{q}(t), dS\dot{q}(t)>\right) = Q(t). \tag{7–24}$$

Since this is precisely the same as Equation (7–8), the motion due to the penalty method converges to the physically correct motion as the maximum deviation of $\overline{x}(t)$ from the intersection manifold converges to zero. The proof by Rubin and Ungar establishes that the normal velocity, which is described by $\dot{h}_1(t)$ through $\dot{h}_k(t)$ in our parameterization, also converges to zero as the steepness of the penalty function P is increased.

7.2 Inequality constrained motion

The penalty method yields a correct result (in the limit) on time intervals during which no constraints change. However, what happens at the instant that a constraint *does* change? Specifically, how does the penalty method know when to "let go" of a contact point, and allow the contact to break? (The converse of this involves a constraint activating, which results in an impulse, but we are assuming that impulses have already been dealt with.) For non-penetration constraints, the penalty force

for a given non-penetration constraint is set to zero if the constraint has not been violated; this prevents penalty forces from becoming attractive. Unfortunately, this does not always give the correct effect, because in some situations the penalty force must act attractively to correctly maintain contact. Thus, one cannot simply release a constraint when the penalty force would have become attractive. It would be nice to speculate that the normal *accelerations*, that is, $\ddot{h}_1(t)$ through $\ddot{h}_k(t)$, converge to zero as the deviation from the intersection manifold goes to zero. If that were so, then it is possible that, given a very small step size Δt, examination of $\ddot{h}_1(t)$ through $\ddot{h}_k(t)$ would indicate whether a constraint should be made inactive; presumably, an increasing value of $\ddot{h}_i(t)$ would indicate that the ith contact point should be released. Unfortunately, Rubin and Ungar give an example that shows that in general $\ddot{h}_1(t)$ through $\ddot{h}_k(t)$ do not converge to zero.[2]

Thus, the penalty method must be used in conjunction with an algorithm that decides when constraints should be released. The algorithms in the next few chapters furnish precisely this information; but since they also calculate the constraint forces as well, they are clearly a better simulation method than the penalty method. The frictional model extrapolated from the penalty method in Chapter 12 is based on the behavior of the penalty method over an interval in which constraints do not change state.

[2] Actually, they don't converge to anything.

Chapter 8

Frictionless Systems

As we saw in the previous chapter, the problem of simulating non-penetration constraints on intervals when there is no change in constraint status is an equality constrained problem. Constraint forces for equality constrained problems can be computed by solving a system of linear equations. However, for the general problem of computing the constraint forces at time t_0 for a system with n contact points, we must allow for the possibility that some set of the constraints may become inactive after time t_0. The inability of the penalty method to determine which constraints should become inactive at time t_0 is a major shortcoming of the method.

Let us consider a system of n contact points without friction. At the ith contact point, the unit surface normal between bodies is denoted $\hat{n}_i$. If this contact occurs between bodies A and B, and $\hat{n}_i$ points outwards from B (towards A) as in Figure 6.4, the constraint force acting on A can be written as $f_{N_i}\hat{n}_i$ where $f_{N_i} \geq 0$. An equal and opposite force $-f_{N_i}\hat{n}_i$ acts on B. The magnitude of the ith constraint force is thus f_{N_i}. The n-vector collection of constraint force magnitudes is denoted $\boldsymbol{f}_{\mathbf{N}}$; the ith element of $\boldsymbol{f}_{\mathbf{N}}$ is f_{N_i}. (In general, matrices and vectors will be denoted by boldface type; for example, "$\boldsymbol{f}_{\mathbf{N}}$". Components of matrices and vectors are

written in regular type; for example, "f_{N_i}".)

If a constraint function $C_i(\overline{x}(t))$ is formulated for the ith contact point, then $\ddot{C}_i(\overline{x}(t_0))$ is a measure of the relative normal acceleration at the contact point, as described in Chapter 6. For simplicity, we will denote this normal acceleration $\ddot{C}_i(\overline{x}(t_0))$ by the quantity a_{N_i}. The n-vector collection of normal accelerations is denoted $\mathbf{a_N}$. The vector $\mathbf{a_N}$ can be considered a function of the unknown constraint force magnitudes $\boldsymbol{f}_\mathbf{N}$. Since acceleration is a linear function of force, $\mathbf{a_N}$ can be expressed, using the derivations of Chapter 6, as

$$\mathbf{a_N} = \mathbf{A}\boldsymbol{f}_\mathbf{N} - \mathbf{b} \tag{8–1}$$

for some $n \times n$ matrix $\mathbf{A}$ and n-vector $\mathbf{b}$.

The matrix $\mathbf{A}$ reflects both the inverse inertias of the contact bodies, and the contact geometry. The vector $\mathbf{b}$ reflects relative normal accelerations at contact points due to known forces; this includes both inertial effects and acceleration due to known external forces. The matrix $\mathbf{A}$ is well known to be positive semidefinite (PSD) and symmetric. The vector $\mathbf{b}$ is always in the range of $\mathbf{A}$, that is,

$$\mathbf{b} = \mathbf{A}\mathbf{c} \tag{8–2}$$

for some n-vector $\mathbf{c}$; situations in which this is not so represent unsatisfiable kinematic constraints on the system and are not of concern to us.

Since inter-penetration is prevented by requiring the relative normal accelerations at each contact point to be non-negative, the non-penetration constraint for the entire system can be written as

$$a_{N_i} \geq 0 \qquad (1 \leq i \leq n). \tag{8–3}$$

We denote this as $\mathbf{a_N} \geq \mathbf{0}$ or, using Equation (8–1), as

$$\mathbf{A}\boldsymbol{f}_\mathbf{N} - \mathbf{b} \geq \mathbf{0}. \tag{8–4}$$

Likewise, the constraint that each $f_{Ni} \geq 0$ is written as

$$\boldsymbol{f}_{\mathbf{N}} \geq \mathbf{0}. \tag{8–5}$$

Recall from Section 2.5.2 that a constraint force, in addition to being non-negative, also has to be zero when the constraint is being broken. As in Section 2.5.2, this can be expressed as the constraint

$$(f_{Ni})(a_{Ni}) = 0 \qquad (1 \leq i \leq n). \tag{8–6}$$

Since $\boldsymbol{f}_{\mathbf{N}} \geq \mathbf{0}$ and $\mathbf{a}_{\mathbf{N}} = \mathbf{A}\boldsymbol{f}_{\mathbf{N}} - \mathbf{b} \geq \mathbf{0}$ imply ${\boldsymbol{f}_{\mathbf{N}}}^T \mathbf{a}_{\mathbf{N}} \geq 0$, Equations (8–4) through (8–6) can be written as

$$\mathbf{a}_{\mathbf{N}} = \mathbf{A}\boldsymbol{f}_{\mathbf{N}} - \mathbf{b} \geq \mathbf{0}, \quad \boldsymbol{f}_{\mathbf{N}} \geq \mathbf{0}, \qquad \text{and} \qquad {\boldsymbol{f}_{\mathbf{N}}}^T \mathbf{a}_{\mathbf{N}} = 0. \tag{8–7}$$

Equation (8–7) defines what is known as a *linear complementarity problem* (LCP); a vector $\boldsymbol{f}_{\mathbf{N}}$ satisfying Equation (8–7) is a solution to the LCP. Thus, the constraint force magnitudes for a frictionless configuration can be computed by solving an LCP. We will call a set of normal forces satisfying Equation (8–7) a *valid* set of constraint forces.

LCP's are closely related to *quadratic programming.* A quadratic program (QP) has the general form

$$\underset{\mathbf{x}}{\text{minimize}}\ \mathbf{x}^T\mathbf{M}\mathbf{x} + \mathbf{c}^T\mathbf{x} \quad \text{subject to} \quad \left\{ \begin{array}{c} \mathbf{M}\mathbf{x} \geq \mathbf{d} \\ \mathbf{x} \geq \mathbf{0} \end{array} \right\}. \tag{8–8}$$

Any LCP can be written as a QP. Using

$${\boldsymbol{f}_{\mathbf{N}}}^T \mathbf{a}_{\mathbf{N}} = {\boldsymbol{f}_{\mathbf{N}}}^T(\mathbf{A}\boldsymbol{f}_{\mathbf{N}} - \mathbf{b}) = {\boldsymbol{f}_{\mathbf{N}}}^T\mathbf{A}\boldsymbol{f}_{\mathbf{N}} - {\boldsymbol{f}_{\mathbf{N}}}^T\mathbf{b}, \tag{8–9}$$

Equation (8–7), phrased as a QP, is

$$\underset{\boldsymbol{f}_{\mathbf{N}}}{\text{minimize}}\ {\boldsymbol{f}_{\mathbf{N}}}^T\mathbf{A}\boldsymbol{f}_{\mathbf{N}} - \mathbf{b}^T\boldsymbol{f}_{\mathbf{N}} \quad \text{subject to} \quad \left\{ \begin{array}{c} \mathbf{A}\boldsymbol{f}_{\mathbf{N}} \geq \mathbf{b} \\ \boldsymbol{f}_{\mathbf{N}} \geq \mathbf{0} \end{array} \right\}. \tag{8–10}$$

The objective function $\boldsymbol{f_N}^T\mathbf{A}\boldsymbol{f_N} - \boldsymbol{f_N}^T\mathbf{b}$ achieves zero for every $\boldsymbol{f_N}$ that satisfies Equation (8–7), and no others. Additionally, the QP formulation can directly represent equality constraints. If the ith constraint of the system is an equality constraint, the condition $f_{Ni} \geq 0$ in Equation (8–10) can be removed, and the condition $a_{Ni} \geq 0$ can be changed to $a_{Ni} = 0$. The QP in this case is still of size n. The same equality constraint can be indirectly represented in Equation (8–7) by a reformulation that adds the constraint $-a_{Ni} \geq 0$ and replaces the variable f_{Ni} with $f_{Ni} = f'_{Ni} - f''_{Ni}$ where $f'_{Ni} \geq 0$ and $f''_{Ni} \geq 0$. The resulting LCP is of size $n + 1$ and is positive semidefinite, but not positive definite.

In the general case, determining if an LCP has a solution or a QP can achieve a certain minimum value are *NP*-complete problems[40,52]. The corresponding problems of finding a solution to an LCP or finding a minimizer of a QP are *NP*-hard. However, when the coefficient matrices of the QP or LCP are PSD, then the problems have polynomial time solutions[40]. LCP's and QP's with PSD coefficient matrices are known as *convex* problems. In practice, convex QP's and LCP's are solved by efficient algorithms whose worst case behavior is exponential but whose expected running time is polynomial. We can use such an algorithm to compute $\boldsymbol{f_N}$ and determine the constraint forces for configurations without friction.

8.1 Existence and uniqueness of solutions

There are two basic questions about $\boldsymbol{f_N}$ that should be asked. First, does an $\boldsymbol{f_N}$ satisfying Equation (8–7) always exist? Furthermore, if an $\boldsymbol{f_N}$ exists, is it unique? (Equivalently: do valid constraint forces exist for every frictionless configuration? Are they unique?) For the case when the LCP of Equation (8–7) is convex, and $\mathbf{b}$ lies in the column-space of $\mathbf{A}$, an $\boldsymbol{f_N}$ satisfying the LCP always exists. If $\mathbf{A}$, in addition to being PSD, is non-singular, then $\mathbf{A}$ is positive definite (PD), and

$\boldsymbol{f}_\mathbf{N}$ is unique. Otherwise, although $\boldsymbol{f}_\mathbf{N}$ is not necessarily unique, $\mathbf{a}_\mathbf{N} = \mathbf{A}\boldsymbol{f}_\mathbf{N} - \mathbf{b}$ is unique; Cottle[7] gives an elegant proof of this fact. The non-uniqueness of $\boldsymbol{f}_\mathbf{N}$ arises in cases where the distribution of constraint forces among contact points is underdetermined. The canonical example is of a chair resting on the floor on four or more legs. Although the acceleration of the chair is unique, the distribution of force on the legs is not.

8.2 Implementation issues

For frictionless configurations, the problem of determining a valid set of constraint forces is well in hand. We will conclude this chapter by discussing two implementation issues in computing $\boldsymbol{f}_\mathbf{N}$.

The first issue involves avoiding computational effort by reusing information from previous time steps. It is well known in the mathematical programming literature that when Equation (8–7) has a solution, it has at least one solution called a *basic* solution. A basic solution may be quantified by a subset $L \subset \{1, 2, \ldots, n\}$ as follows. Let $\mathbf{B}$ denote the matrix obtained from $\mathbf{A}$ by deleting each row and column whose index is not in L; this yields a square sub-matrix of $\mathbf{A}$. Similarly, let $\mathbf{d}$ denote the vector obtained from $\mathbf{b}$ by deleting elements of $\mathbf{b}$ whose index is not in L.

The special properties of a basic solution are as follows. First, the matrix $\mathbf{B}$ is invertible. Second, the vector $\mathbf{B}^{-1}\mathbf{d}$ satisfies $\mathbf{B}^{-1}\mathbf{d} \geq \mathbf{0}$. Last, let us expand the vector $\mathbf{B}^{-1}\mathbf{d}$ by inserting zero elements. The zeroes are inserted into $\mathbf{B}^{-1}\mathbf{d}$ by inverting the process of deleting elements from $\mathbf{b}$ to form $\mathbf{d}$; the expanded vector has a zero for every index i not contained in L. The newly expanded vector is then a solution to Equation (8–7).

The concept of a basic solution also applies to the QP of Equation (8–10); a solution to Equation (8–10) can be obtained by inverting a matrix formed from the

coefficient matrices defining the QP. Most algorithms for solving LCP's and QP's find a basic solution. In a simulation environment, the concept of a basic solution can greatly reduce the amount of numerical work done at each time step. The major effort of computing the constraint forces lies in solving for $\boldsymbol{f}_{\mathbf{N}}$. However, given a basic solution for $\boldsymbol{f}_{\mathbf{N}}$ at one time step, the simulator can usually exploit coherence in L at the next time step. Although the coefficients given by $\mathbf{A}$ and $\mathbf{b}$ vary from one time step to the next, the index set L denoting a basic solution usually remains valid for some large number of time steps. Whenever L is known, $\boldsymbol{f}_{\mathbf{N}}$ is easily found by computing $\mathbf{B}^{-1}\mathbf{d}$. As a result, $\boldsymbol{f}_{\mathbf{N}}$ can be computed by solving only a linear equation for most time steps.

Using L from the previous time step, we may occasionally produce a singular $\mathbf{B}$ for the current time step or a non-positive vector $\mathbf{B}^{-1}\mathbf{d}$, indicating that L is invalid for the current time step. This usually happens when conditions change abruptly due to an impact. It also happens whenever an a_{Ni} becomes positive, indicating separation at a contact point. Since this may indicate a discontinuity in acceleration[30], the ordinary differential equation integrator is restarted at this point. Whenever L becomes invalid, a solution for $\boldsymbol{f}_{\mathbf{N}}$ must be computed from scratch. The extra cost of forming $\mathbf{B}$ and $\mathbf{d}$ and computing $\mathbf{B}^{-1}\mathbf{d}$ even when L is invalid is still much less than the cost of always computing $\boldsymbol{f}_{\mathbf{N}}$ from scratch. Gilmore and Cipra[20] discuss methods for predicting the change in L from one time step to the next.

The second implementation issue has to do with the formulation of $\boldsymbol{f}_{\mathbf{N}}$ in terms of a QP. In analysis, it is well known that a necessary, but not sufficient condition for a function $y(x)$ to achieve a minimum at x_0 is that the gradient of y vanish at x_0. This is known as a *first order* optimality condition. For a QP, analogous first order conditions called the *Karush-Kuhn Tucker* (KKT) conditions for optimality exist. In the case of a convex QP, the KKT conditions are both necessary and sufficient;

the global minimum is achieved if and only if the KKT conditions hold[40]. For a PSD symmetric matrix $\mathbf{A}$, as is the case for frictionless configurations, the QP

$$\underset{\boldsymbol{f}_\mathbf{N}}{\text{minimize}} \ \tfrac{1}{2}\boldsymbol{f}_\mathbf{N}{}^T\mathbf{A}\boldsymbol{f}_\mathbf{N} - \boldsymbol{f}_\mathbf{N}{}^T\mathbf{b} \quad \text{subject to} \quad \boldsymbol{f}_\mathbf{N} \geq \mathbf{0} \tag{8–11}$$

has KKT conditions

$$\mathbf{A}\boldsymbol{f}_\mathbf{N} - \mathbf{b} = \boldsymbol{\lambda}, \quad \boldsymbol{f}_\mathbf{N}{}^T\boldsymbol{\lambda} \geq 0 \qquad \text{and} \qquad \boldsymbol{\lambda} \geq 0. \tag{8–12}$$

(The elements of the n-vector $\boldsymbol{\lambda}$ are referred to as the *Lagrange multipliers* of the minimization problem.) A solution of Equation (8–11) is therefore a solution of Equation (8–10). The vector $\boldsymbol{f}_\mathbf{N}$ may be computed by solving either Equation (8–10) or Equation (8–11). The form of the two different QP formulations will become more important when we deal with static friction in Chapter 14. In actual practice, Equation (8–10) is computationally preferable in dealing with frictionless systems. In dealing with the less-than-perfect world of numerical solution and optimization, including the explicit constraint $\mathbf{A}\boldsymbol{f}_\mathbf{N} - \mathbf{b} \geq \mathbf{0}$ in the problem formulation increases the likelihood of finding an $\boldsymbol{f}_\mathbf{N}$ that satisfies the most important constraint of non-penetration.

Chapter 9

The Coulomb Friction Model

At this point, we will extend our treatment of contact between bodies to the case when frictional forces act at contact points. 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. In this chapter, the model of friction used throughout the rest of this thesis is defined. Following this chapter, we will consider the theoretical and practical complexity of computing both normal and friction forces. We will say that the contact forces acting on a configuration with friction are valid if the friction forces satisfy the conditions defined in this section, and the constraint forces satisfy the conditions of the previous chapter.

9.1 Coordinate geometry at a contact point

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. As in Chapter 8, we adopt the convention that $\hat{n}$ points from B towards A; for example, $\hat{n}$ would be $\nabla G(p_b, \overline{p})/\|\nabla G(p_b, \overline{p})\|$ in Figure 6.4. 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. Following the same convention as for normal forces, the friction force acting on A is $f_x\hat{t}_x + f_y\hat{t}_y$, with an equal and opposite friction force acting on B.

Since friction acts to oppose any slipping motion, let us consider the slip velocity tangent to the contact surface. In Equation (6–32) we derived s, the slip velocity between A and B at the contact point. 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:

$$v_x = \hat{t}_x \cdot s \qquad \text{and} \qquad v_y = \hat{t}_y \cdot s. \tag{9–1}$$

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

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

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

$$a_x = \hat{t}_x \cdot \frac{\mathrm{d}}{\mathrm{dt}}s \qquad \text{and} \qquad a_y = \hat{t}_y \cdot \frac{\mathrm{d}}{\mathrm{dt}}s. \tag{9–2}$$

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

$$f_x^2 + f_y^2 \leq (\mu f_N)^2 \tag{9–3}$$

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 non-zero, 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.)

9.2 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,

$$f_x\hat{t}_x + f_y\hat{t}_y = -\mu f_N \frac{v_x\hat{t}_x + v_y\hat{t}_y}{\sqrt{v_x^2 + v_y^2}} \tag{9–4}$$

or

$$f_x = -\mu f_N \frac{v_x}{\sqrt{v_x^2 + v_y^2}} \qquad \text{and} \qquad f_y = -\mu f_N \frac{v_y}{\sqrt{v_x^2 + v_y^2}}. \tag{9–5}$$

Thus, given μ and the tangential velocity, the dynamic friction force can be written

solely in terms of f_N. Since

$$\begin{aligned} f_N\hat{n} + f_x\hat{t}_x + f_y\hat{t}_y &= f_N\hat{n} + \frac{-\mu f_N v_x\hat{t}_x - \mu f_N v_y\hat{t}_y}{\sqrt{v_x^2 + v_y^2}} \\ &= f_N(\hat{n} + \frac{-\mu v_x\hat{t}_x - \mu v_y\hat{t}_y}{\sqrt{v_x^2 + v_y^2}}), \end{aligned} \tag{9–6}$$

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

9.3 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

$$a_x = a_y = 0 \qquad \text{and} \qquad f_x^2 + f_y^2 \leq (\mu f_N)^2, \tag{9–7}$$

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

$$f_x^2 + f_y^2 = (\mu f_N)^2 \qquad \text{and} \qquad (f_x\hat{t}_x + f_y\hat{t}_y) \cdot (a_x\hat{t}_x + a_y\hat{t}_y) \leq 0, \tag{9–8}$$

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.

9.4 Impulsive frictional forces

When two bodies collide at a contact point with friction, we can consider the collision as taking place over some small but non-zero 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[34,54] 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.

For simulation purposes, the simulator described in this thesis makes the simplification that the direction of the relative tangential velocity does not change during a collision. For planar collisions, this amounts to ignoring the possibility of reversed sliding, as considered by Wang and Mason. For three-dimensional collisions, this is simply an out-and-out approximation that is reasonable for some situations, and probably unreasonable for others. This model is no more than an *ad hoc* approximation that allows the simulator to produce a (hopefully) reasonable looking collision response behavior. However, the model, though incorrect physically, is computationally attractive because it produces precisely the same sort of constraint equations as are encountered for static friction.

The idea is the following. Given a collision at a contact point, let f_N denote the normal impulse magnitude, f_x and f_y the tangential frictional impulse, and v_N and v_N^+ the relative approach speeds in the the normal direction prior to the collision, and after the collision, respectively. Similarly, let v_x^+ and v_y^+ denote the tangential velocity components after the collision. The constraints on the normal quantities are $f_N > 0$, and $v_N^+ = -\epsilon v_N$ where ϵ is the coefficient of restitution of

the collision. Note that this kinematic condition, which is Newton's hypothesis, is known to violate energy conservation under some circumstances if both ϵ and μ are non-zero[3]. The standard condition $f_x^2 + f_y^2 \leq (\mu f_N)^2$ is imposed at all times on the frictional impulse.

If the bodies are sliding after application of the impulse, then the frictional impulse must achieve its upper bound, and must oppose the sliding velocity. This is written as

$$f_x^2 + f_y^2 = (\mu f_N)^2 \qquad \text{and} \qquad (f_x \hat{t}_x + f_y \hat{t}_y) \cdot (v_x^+ \hat{t}_x + v_y^+ \hat{t}_y) \leq 0. \tag{9–9}$$

Otherwise, the bodies are not sliding, and the frictional impulse need only satisfy its upper bound condition. This yields

$$v_x^+ = v_y^+ = 0 \qquad \text{and} \qquad f_x^2 + f_y^2 \leq (\mu f_N)^2. \tag{9–10}$$

Comparing these last two equations to Equations (9–8) and (9–7), it is clear that both sets of equations can be solved by the same techniques. Accordingly, we will be able to use the solution methods described in Chapter 14 to compute frictional impulses.

If we are dealing with a number of contact points simultaneously, we will need to add the following kinematic constraint. If at some colliding contact $v_N^+ > -\epsilon v_N$, then the normal impulse at that contact must be zero. In other words, if bodies bounce away at a contact point with a speed in excess of that predicted by the coefficient of restitution, the normal impulse at that point must be zero[13]. Satisfying these conditions is still exactly analogous to satisfying the static friction conditions for a collection of non-colliding contact points, so again, we can compute frictional impulses using the static friction methods described in Chapter 14.

Chapter 10

Dynamic Friction

Having defined a model of friction, we now consider the complexity of computing valid contact forces in the presence of friction. Most of the complexity results in this thesis apply to configurations with dynamic friction only. Complexity results for static friction are presented in Chapter 14. Until then, only configurations of bodies with dynamic friction are considered.

Given that no static friction forces act at any contact points, the contact force direction at each contact point is given by Equation (9–6), since μ and the tangential velocity are known quantities. Thus, as in Chapter 8, we are interested in computing only the vector $\boldsymbol{f}_{\mathbf{N}}$ of constraint force magnitudes. Solutions for $\boldsymbol{f}_{\mathbf{N}}$ for configurations with just dynamic friction can still be expressed as the LCP of Equation (8–7). However, as a result of the contact force having a tangential component, the matrix $\mathbf{A}$ in Equation (8–7) is no longer necessarily symmetric PSD and Equation (8–7) is no longer a convex LCP.

As a result, several nice properties are lost. Although solutions to convex LCP's can be found in polynomial time, non-convex LCP's do not share this property; finding solutions for non-convex LCP's is *NP*-hard. What then, is the complexity

of solving the non-convex LCP's corresponding to configurations of bodies with dynamic friction? Also, the uniqueness and existence properties associated with the convex LCP's of Chapter 8 no longer always hold. With dynamic friction, it is possible that there may be no solution $\boldsymbol{f_N}$, and thus no valid contact forces. We call a configuration exhibiting this behavior *inconsistent.*[1] It is also possible that the acceleration produced by solutions of the LCP is not unique; such a configuration is called *indeterminate.*

Configurations with one contact point that exhibit inconsistency and indeterminacy have been discussed by Erdmann[12], Lötstedt[28], and Mason and Wang[35]. Lötstedt[30], realizing that indeterminacy and inconsistency major present 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. We will discuss Lötstedt's modification in Chapter 14, when we consider static friction. In this chapter, we discuss two well known one contact point configurations that exhibit inconsistency and indeterminacy. Chapter 11 uses these configurations as building blocks in a proof about the computational complexity of solving dynamic friction problems.

10.1 Indeterminacy

In Figure 10.1, body A is a thin rod of length two with a symmetric mass distribution that contacts body B at a single contact point. Body B, the "base", is fixed. The

[1] The use of the word "inconsistent" throughout this thesis requires some qualification. The fact that configurations exist which have no valid contact forces is counter-intuitive to most people at first. These configurations are "inconsistent" with the initial expectation that a valid set of (finite) contact forces must exist for any configuration that can be physically realized. However, we do not use "inconsistent" to mean "unnatural", nor do we mean to imply that inconsistent configurations are in any way extraordinary physical occurrences. Readers who are familiar with "self-locking" behavior of assemblies due to friction will find "inconsistent" configurations to be nothing special at all.

example is two-dimensional, so the tangent direction is written simply as $\hat{t}$. The particular values of I, θ and μ given in Figure 10.1 are chosen to simplify later computations. A gravity force of $-mg\hat{n}$ acts on A. Let the angular velocity ω and gravity constant g be such that A is rotating clockwise and

$$|\omega|^2 \sin\theta - g = 1. \tag{10–1}$$

Given these values, let the linear velocity of A be chosen so that the point p_a of the rod is sliding tangent to B as shown. Since A and B are neither colliding nor separating at p_a, a contact force is presumed to act. Since the tangential velocity is non-zero, a dynamic friction force acts on A in the $\hat{t}$ direction. From Chapter 9, the net contact force acting on A is

$$f_N\hat{n} + \mu f_N\hat{t} = f_N(\hat{n} + \mu\hat{t}). \tag{10–2}$$

Because B is fixed and A is thin, the normal acceleration a_N is simply

$$a_N = \hat{n} \cdot \ddot{p}_a. \tag{10–3}$$

To compute a_N, let a and α denote the linear and angular acceleration of A, ω, the angular velocity of A, and r, the displacement of p_a from the center of mass of A. Although this configuration is two-dimensional, we will regard a, p_a, $\ddot{p}_a$, $\hat{n}$ and r as 3-space vectors lying in the xy plane, and ω and α as 3-space vectors parallel to the z axis. From Figure 10.1, $r = (-\cos\theta, -\sin\theta, 0)$. The acceleration $\ddot{p}_a$ may be expressed as the sum of three terms: the linear acceleration a, the tangential acceleration $\alpha \times r$, and the centripetal acceleration $\omega \times (\omega \times r)$. The linear acceleration, a, is

$$\begin{aligned} a &= \frac{f_N\hat{n} + \mu f_N\hat{t} + (-mg)\hat{n}}{m} \\ &= \frac{f_N\hat{n} + \mu f_N\hat{t}}{m} - g\hat{n}. \end{aligned} \tag{10–4}$$

Variables

p_a	contact point	$\dot{p}_a$	contact point velocity
$\hat{n}$	unit surface normal	$\hat{t}$	unit surface tangent
m	A's mass	I	A's moment of inertia
ω	A's angular velocity	μ	coefficient of friction
g	gravitational acceleration		

Relations

$$I = \frac{m}{16} \qquad \theta = 72° \qquad 16(\cos^2\theta - \mu\cos\theta\sin\theta) = -2 \qquad (\mu \approx 3/4)$$

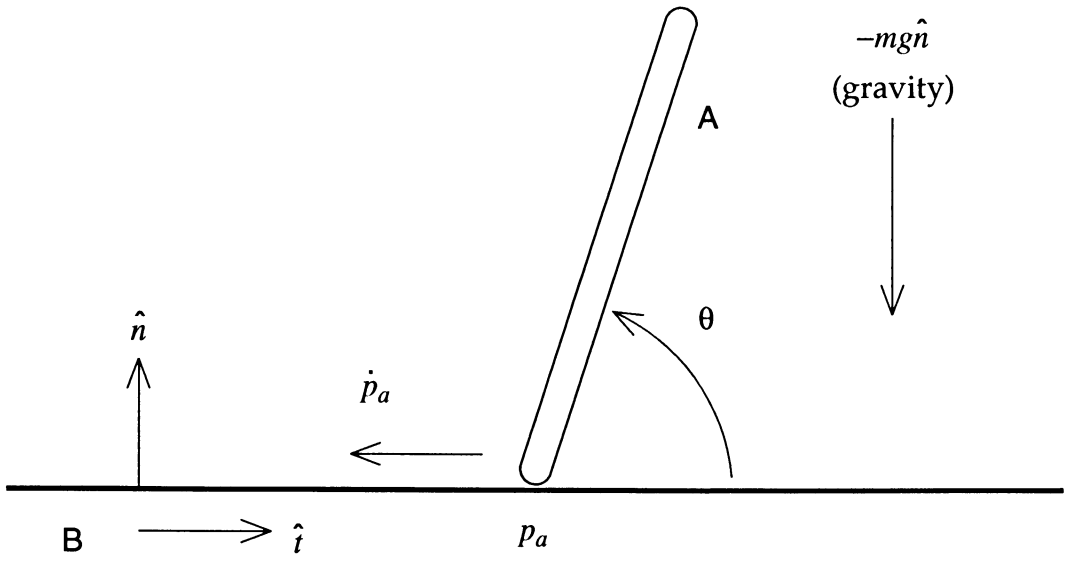


Figure 10.1: A one contact point configuration between a thin rod A and a fixed base B.

The torque on A about its center of mass is

$$r \times (f_N \hat{n} + \mu f_N \hat{t})$$

which yields an angular acceleration of

$$\alpha = \frac{r \times (f_N \hat{n} + \mu f_N \hat{t})}{I}. \tag{10–5}$$

Then

$$\begin{aligned} \ddot{p}_a &= a + \alpha \times r + \omega \times (\omega \times r) \qquad (10–6) \\ &= \frac{f_N \hat{n} + \mu f_N \hat{t}}{m} - g\hat{n} + \frac{r \times (f_N \hat{n} + \mu f_N \hat{t})}{I} \times r + \omega \times (\omega \times r). \end{aligned}$$

Taking the dot product of Equation (10–4) with $\hat{n}$,

$$\hat{n} \cdot a = \frac{f_N \hat{n} \cdot \hat{n} + \mu f_N \hat{n} \cdot \hat{t}}{m} - g\hat{n} \cdot \hat{n} = \frac{f_N}{m} - g. \tag{10–7}$$

Taking the dot product of the tangential acceleration $\alpha \times r$ with $\hat{n}$ and using the geometry of Figure 10.1,

$$\hat{n} \cdot \omega \times (\omega \times r) = |\omega|^2 \sin\theta \tag{10–8}$$

and

$$\begin{aligned} \hat{n} \cdot (\alpha \times r) &= \hat{n} \cdot \left(\frac{r \times (f_N \hat{n} + \mu f_N \hat{t})}{I} \times r \right) \\ &= \frac{f_N(\cos^2\theta - \mu \cos\theta \sin\theta)}{I}. \qquad (10–9) \end{aligned}$$

Then, from the relations in Figure 10.1,

$$\begin{aligned} \hat{n} \cdot \ddot{p}_a &= \frac{f_N}{m} - g + \frac{f_N(\cos^2\theta - \mu \cos\theta \sin\theta)}{I} + |\omega|^2 \sin\theta \\ &= \frac{f_N}{m}(1 + 16(\cos^2\theta - \mu \cos\theta \sin\theta)) + |\omega|^2 \sin\theta - g \\ &= \frac{f_N}{m}(1 - 2) + |\omega|^2 \sin\theta - g \qquad (10–10) \\ &= -\frac{f_N}{m} + (|\omega|^2 \sin\theta - g). \end{aligned}$$

Finally, from Equation (10–1),

$$\begin{aligned} \hat{n} \cdot \ddot{p}_a &= -\frac{f_N}{m} + (|\omega|^2 \sin\theta - g) \\ &= -\frac{f_N}{m} + 1. \end{aligned} \tag{10–11}$$

From Equation (8–7), the constraints on f_N are the LCP

$$f_N \geq 0, \quad -\frac{f_N}{m} + 1 \geq 0 \qquad \text{and} \qquad f_N(-\frac{f_N}{m} + 1) = 0. \tag{10–12}$$

This LCP yields two solutions for f_N; one solution is $f_N = 0$, and the other is $f_N = m$. For the $f_N = 0$ solution, $\hat{n} \cdot \ddot{p}_a = 1$. In this solution, the centripetal acceleration of $\ddot{p}_a$ is stronger than the force of gravity pulling A down; thus, A merely continues its rotation and the point p_a moves off of B (Figure 10.2a).

In the second solution, $f_N = m$ and $\hat{n} \cdot \ddot{p}_a = 0$. A normal force of $m\hat{n}$ and a friction force of $\mu m\hat{t}$ act on A. The torque generated by friction balances the centripetal acceleration of p_a; as a result, A and B do not break contact (Figure 10.2b). Note that the *only* valid values of f_N for this configuration are $f_N = 0$ or $f_N = m$. Since the solutions produce different accelerations for A, the configuration is indeterminate.

The fact that there is more than one possible behavior for A should not be of great concern. We have already seen that the rigid body assumption allows for non-unique distributions of contact forces, as in the case of a chair resting on four legs. When the rigid body assumption is relaxed to allow deformation, the indeterminacy in the force distribution is resolved. In the case of the sliding rod, sufficient knowledge about the flexibility and compressibility of the rod would serve to determine the actual behavior of the rod. Since we have chosen to treat bodies as perfectly rigid, we are in essence stating that this knowledge is not available to us.

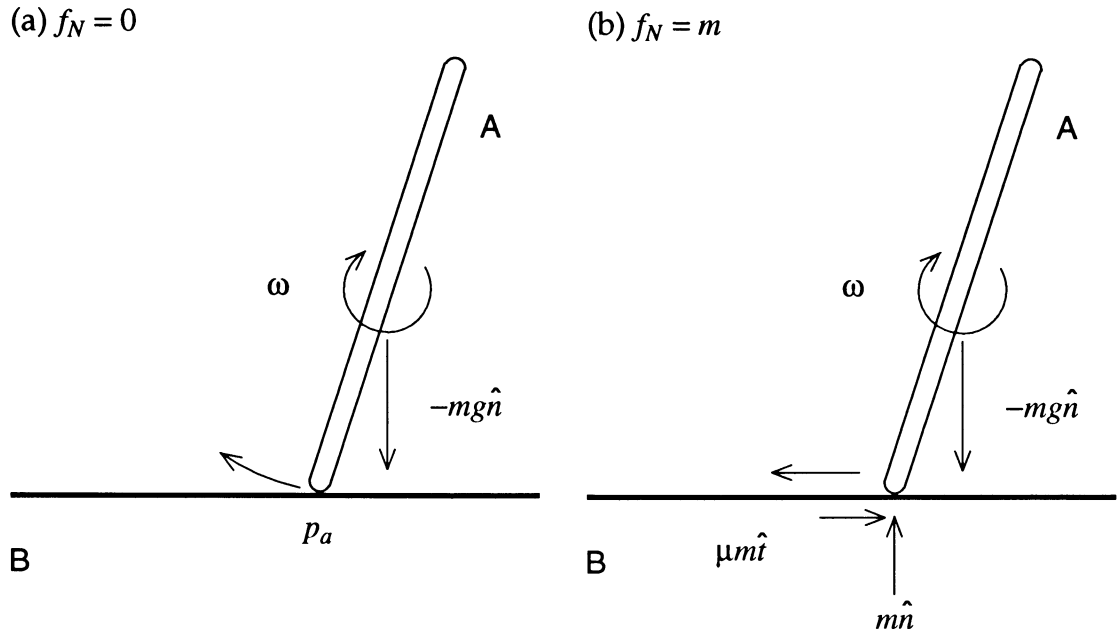


Figure 10.2: An indeterminate configuration. (a) The contact force between A and B is zero. The point p_a rotates to the left and up, breaking contact with B. (b) The normal and friction forces balance gravity and centripetal acceleration; p_a moves horizontally and maintains contact with B.

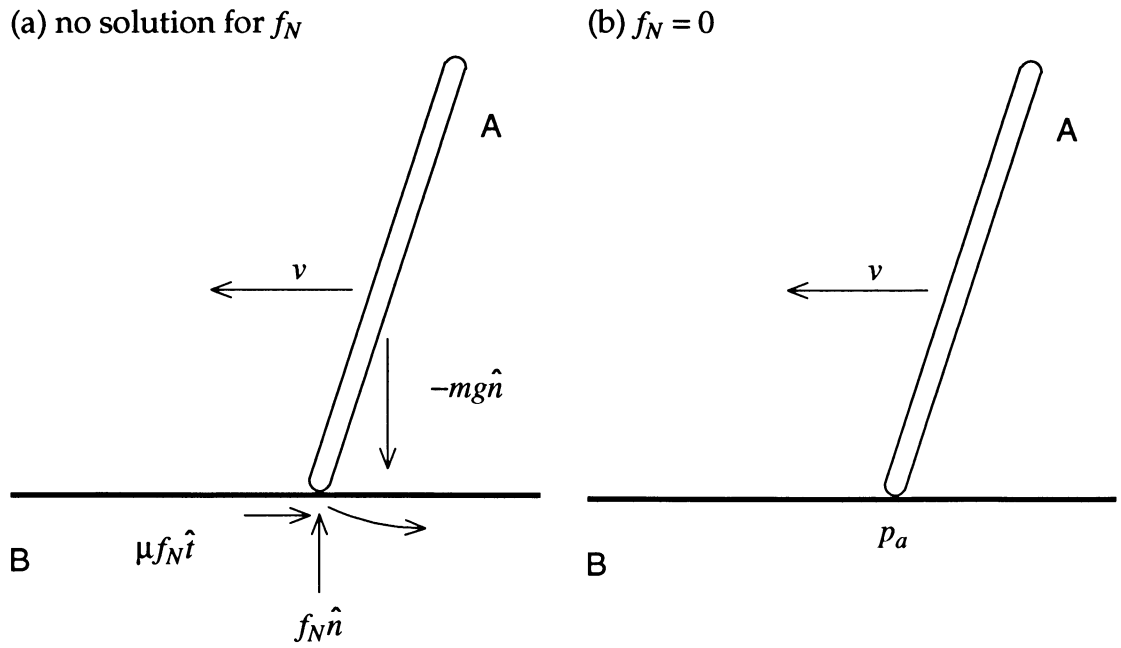


Figure 10.3: A potentially inconsistent configuration. (a) For any $f_N \geq 0$, the point p_a is accelerated downwards into B. This configuration is inconsistent. (b) The configuration has a unique solution of $f_N = 0$ when gravity is removed; A skims along the surface of B. This configuration is not inconsistent.

10.2 Inconsistency

To demonstrate inconsistency, consider the same sliding rod configuration, but with a different angular velocity. Suppose that A has an angular velocity of zero ($|\omega| = 0$), and a linear velocity v opposite $\hat{t}$, as shown in Figure 10.3. Then the condition $a_N = \hat{n} \cdot \ddot{p}_a \geq 0$ is

$$\hat{n} \cdot \ddot{p}_a = -\frac{f_N}{m} - g \geq 0. \tag{10–13}$$

However, if $g > 0$ (Figure 10.3a), then $f_N \geq 0$ implies that $\hat{n} \cdot \ddot{p}_a < 0$. Thus, no (positive) value of f_N is sufficient to prevent p_a from accelerating downwards and into B. This configuration is inconsistent.

Note that the value of g here is crucial. If $g = 0$, so that no external force acts

on A, then $f_N = 0$ becomes the (unique) valid contact force (Figure 10.3b). Any positive value of f_N for this configuration results in inter-penetration. Figure 10.3b corresponds to to p_a "skimming" horizontally over B, with neither a normal force nor a friction force exerted on A. If g becomes even slightly positive however, the configuration is inconsistent.

How can we interpret this phenomenon? Although our mathematical model admits no solution for this configuration, we know that there must be some physical answer. Still, how can we get around the fact that no matter what value we assume for f_N, the point p_a accelerates downwards? The resolution of the issue lies in how we view the subject of inconsistency. When two rigid bodies collide, no force, no matter how large, is sufficient to prevent some degree of inter-penetration. For rigid body mechanics, the need to prevent inter-penetration requires the extension of the rigid body model to include the concept of impulsive forces which can discontinuously change velocity to prevent inter-penetration. Impulsive forces are added to the rigid body model to avoid the inconsistency that non-impulsive forces are not always sufficient to prevent inter-penetration.

For the sliding inconsistent rod of Figure 10.3a, non-impulsive forces cannot prevent inter-penetration because the constraints of dynamic friction result in a fixed direction for the contact force, no matter how strong. In contrast, a contact impulse applied to A and B can prevent inter-penetration because the direction of the contact impulse is not fixed. After the impulse is applied, the inconsistency is removed and a non-impulsive contact force can be found. Mason and Wang[35] discuss the computation of this frictional impulse and show that it is consistent with the way in which frictional impulses are determined for collisions between rigid bodies.

Interestingly, what appears to disturb people most about the configuration of Figure 10.3a is that an impulse is necessary even though the relative normal velocity

between A and B is zero. Because of this, the configuration is seen as pathological, and the application of an impulse is seen as an action of last resort. In the next chapter, we will consider the computational complexity of finding non-impulsive solutions, so as to avoid applying impulsive solutions as much as possible.

Chapter 11

An *NP*-complete Class of Configurations

We define the *frictional consistency* problem as the problem of deciding if a given configuration with dynamic friction is consistent. In this section, we prove that the frictional consistency problem is *NP*-complete. We first show that the frictional consistency problem lies in *NP* and then establish the frictional consistency problem as *NP*-hard. Although the configurations constructed in this section seem contrived (and arguably are), the *NP*-hardness result has grave implications even when inconsistency is not a concern during simulation.

DEFINITION. *An instance of the frictional consistency problem is a configuration* C *of bodies that contact at* n *distinct contact points. The physical properties of each body (mass, moment of inertia, linear and angular velocity, position and orientation, and external forces) are described by rational numbers. The specifics of a contact point (position, coefficient of friction, surface normal) are also described by rational numbers. The relative motion between bodies at contact points with friction is non-zero in the direction tangent to the contact surface and zero in the direction normal to*

the contact surface. The notation $|\mathrm{C}| = \mathrm{k}$ *means that configuration* C *is describable in* k *bits. Clearly* $\mathrm{n} < \mathrm{k}$.

THEOREM 1. *The frictional consistency problem lies in NP.*

PROOF. Given an instance of C, an LCP of size n with the following two properties exists.

(1) If C is consistent, then an n-vector $\boldsymbol{f}_{\mathbf{N}}$ that is a solution to the LCP exists. The set of contact forces such that the magnitude of the normal force at the ith contact point is f_{Ni} is a valid set of contact forces for the configuration C.

(2) If C is inconsistent, the LCP has no solution.

The LCP is the LCP of Equation (8–7). The numerical quantities in the LCP are computed from the rational entries of C in a total of $O(n^3)$ arithmetical operations. The LCP can therefore be constructed in time polynomial to k. The problem of deciding if an LCP has a solution lies in *NP*[40]. It follows from this that deciding frictional consistency is also in *NP*. □

In order to show that deciding frictional consistency is *NP*-hard, we will reduce the *NP*-complete problem "subset sum" to the frictional consistency problem.

DEFINITION. *An instance of the subset sum problem is a pair* (A, S) *where* $\mathrm{A} = \{\mathrm{a}_1, \ldots, \mathrm{a}_\mathrm{n}\}$ *is a set of positive integers and* S *is a single positive integer. A subset sum instance* (A, S) *is* satisfiable *if there exists a subset* $\mathrm{A}' \subset \mathrm{A}$ *such that*

$$\sum_{\mathrm{a} \in \mathrm{A}'} \mathrm{a} = \mathrm{S}. \tag{11–1}$$

Deciding if an instance of the subset sum problem is satisfiable is an NP-complete problem[16].

To show that deciding frictional consistency is *NP*-hard we will take an arbitrary instance (A, S) of the subset sum problem and construct (in polynomial time) a configuration of bodies C. The configuration C will have the property that C is consistent if and only if (A, S) is satisfiable.

THEOREM 2. *Deciding frictional consistency is NP-hard.*

PROOF. We begin by presenting a slightly modified version of the configuration of Figure 10.3. In Figure 11.1, body B is free to move vertically, either up or down, but it is still not allowed to rotate or move horizontally. Let the mass M, of body B, be $M = 100m$ so that B is still massive compared to A. Assume that there is no external gravity force acting on A, but that B is subject to a vertical external force $f_b\hat{n}$, though its center of mass (Figure 11.1). A positive value for f_b indicates the force acts upwards, pushing B towards A.

Since B may now move vertically, Equation (10–13) is modified. Since there is no external force on A, $g = 0$. In place of $-f_N/m - g \geq 0$,

$$-\frac{f_N}{m} + \frac{f_N}{M} - \frac{f_b}{M} \geq 0. \tag{11–2}$$

The term f_N/M is the linear acceleration of B away from A due to the contact force. The term f_b/M is the upwards linear acceleration of B due to the external force. Since $M = 100m$,

$$-\frac{f_N}{m} + \frac{f_N}{M} - \frac{f_b}{M} = -\frac{99}{100}\frac{f_N}{m} - \frac{f_b}{M}. \tag{11–3}$$

Equation (10–13) is then rewritten as

$$-\frac{99}{100}\frac{f_N}{m} - \frac{f_b}{M} \geq 0. \tag{11–4}$$

As before, if $f_b > 0$, an inconsistency occurs. If $f_b = 0$, then $f_N = 0$ is the unique valid solution and the configuration is consistent.

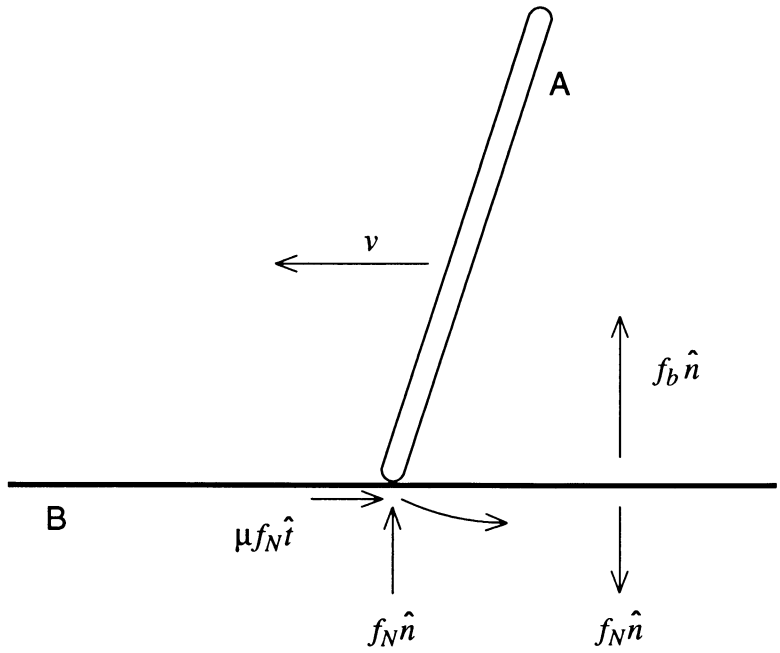


Figure 11.1: A possibly inconsistent configuration. Body B may move vertically, but is not allowed to move horizontally or rotate. If the external vertical force magnitude f_b is positive, the configuration is inconsistent. The contact force $f_N\hat{n}$ on A acts equally and opposite on B.

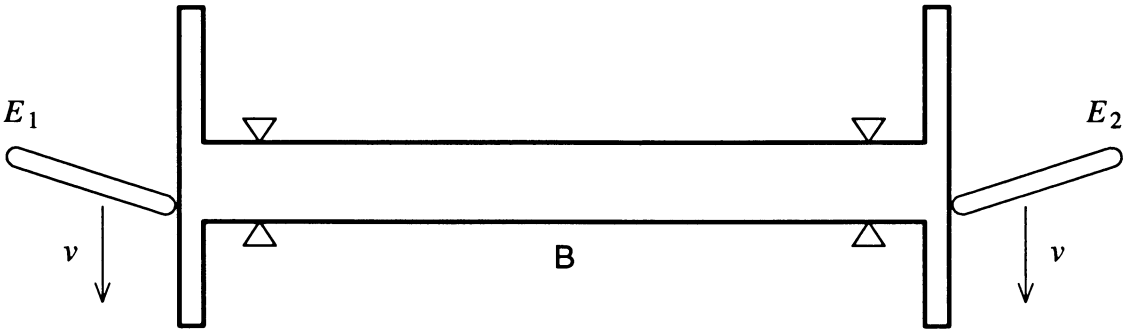


Figure 11.2: A configuration whose consistency depends on an external force. Body B is constrained by the fixed wedges and can only move horizontally. The configuration is consistent only if B is not subject to a net horizontal force.

Now, consider the configuration of Figure 11.2. Body B of Figure 11.2 is initially at rest and is positioned by four fixed triangular wedges that contact B without friction. Body B is therefore free to move horizontally, but can neither rotate nor move vertically. On either side of body B are thin rods E_1 and E_2 that have no angular velocity, and a linear velocity as indicated. The rods E_1 and E_2 contact B in the same manner as the configuration of Figure 11.1 except that the frames of reference for E_1 and E_2 are rotated by 90° with respect to Figure 11.1. In the configuration of Figure 11.1, inconsistency occurred if f_b was positive, pushing B towards A. The same holds true for Figure 11.2. If B has an acceleration leftwards (towards E_1), then inconsistency occurs. Likewise, if B has an acceleration rightwards (towards E_2), then inconsistency also occurs. Thus, the configuration of Figure 11.2 is consistent only if the net horizontal acceleration of B is zero. In this case, the rods E_1 and E_2 skim along the surface of B as in Figure 10.3b.

Now consider Figure 11.3, where a collection of thin rods $R_1, \ldots, R_n$ have been added. In addition, an external horizontal force with magnitude μS acts on B,

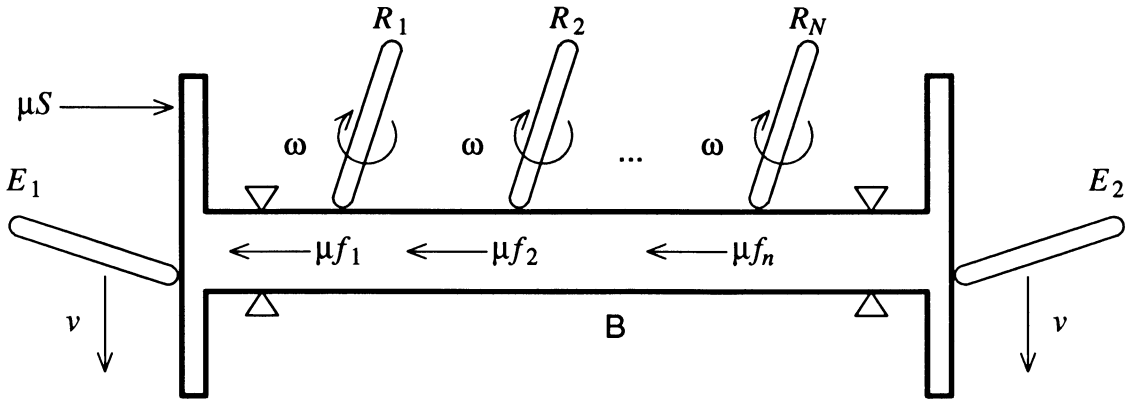


Figure 11.3: A configuration that is consistent if and only if the friction forces on B sum to μS.

trying to accelerate B to the right. Each rod R_i has mass m_i. The configuration between each rod R_i and B is the same as the configuration of Figure 10.2; thus each rod R_i has angular velocity ω and is subject to an external gravity force. Let f_{Ni} be the magnitude of the normal force between R_i and B. As in Figure 10.2, the only valid solutions for f_{Ni} are $f_{Ni} = 0$ and $f_{Ni} = m_i$. If $f_{Ni} = 0$, then no friction force acts between R_i and B. Otherwise, $f_{Ni} = m_i$ and a friction force of magnitude μm_i acts between R_i and B. The friction force pushes R_i to the right and B to the left, with magnitude μm_i. The friction force on B therefore acts to oppose the external force of magnitude μS.

In order for the configuration of Figure 11.3 to be consistent, B must have no net horizontal acceleration. This means that the friction forces exerted on B from the n rods must sum to μS, balancing the external force applied to B. Thus, the configuration is consistent if and only if

$$\sum_{i=1}^{n} \mu f_{Ni} = \mu S. \tag{11–5}$$

Since each f_{Ni} is either 0 or m_i, the configuration is consistent if and only if some subset of $\{m_1, \ldots, m_n\}$ sums to S.

We can now perform the reduction from subset sum to show *NP*-hardness. Given any set $A = \{a_1, \ldots, a_n\}$ and any target sum S, construct the configuration of Figure 11.3. The values chosen in the previous section for μ, θ and ω may introduce irrational quantities into the descriptions of the configurations. Since the frictional consistency problem must be described in terms of rational quantities, we may assume that μ, θ and ω are suitably perturbed so that all quantities are rational, without otherwise disturbing the solution properties of the configurations. This can be done since the set of rational numbers is dense with respect to the set of real numbers. In the constructed configuration, assign $m_i = a_i$ for $1 \leq i \leq n$, and let an external horizontal force of μS act on B as shown in Figure 11.3. By the above discussion, the configuration is consistent if and only if there exists a subset of $\{m_1, \ldots, m_n\}$ that sums to S. But since $A = \{m_1, \ldots, m_n\}$, the configuration is consistent if and only if (A, S) is satisfiable. We conclude that the problem of deciding frictional consistency is *NP*-hard. □

THEOREM 3. *Deciding frictional consistency is NP-complete.*

PROOF. The result follows immediately from Theorem 1 and Theorem 2. □

COROLLARY 1. *Computing contact forces (if they exist) for a configuration is NP-hard.*

PROOF. Since deciding if a set of contact forces exists is an *NP*-complete problem, computing the contact forces (if they exist) is an *NP*-hard problem. □

A reasonable response to this corollary is to question its practical applicability to the problem of computing valid contact forces. Suppose the possibility of an inconsistent configuration occurring in a simulation is so unlikely that it can be discounted completely. Can we construct an efficient algorithm that computes contact

forces only for the set of consistent configurations? The following corollary asserts that a polynomial time algorithm for computing contact forces, even restricted to consistent configurations, does not exist unless $P = NP$.

COROLLARY 2. *A polynomial time algorithm for computing valid contact forces for consistent configurations exists if and only if* $P = NP$.

PROOF. Suppose that $P = NP$. Since deciding if an LCP has a solution lies in NP, $P = NP$ implies a polynomial time algorithm for finding the solution to an LCP. Since valid contact forces for a consistent configuration of bodies can be found by solving an associated LCP, valid contact forces are computable in polynomial time if $P = NP$.

Conversely, suppose that contact forces for consistent configurations can be computed in polynomial time. Then there exists a machine M and a polynomial p with the following behavior. Whenever M is given a consistent configuration C as input, M outputs a valid set of contact forces within time $p(|C|)$. M's behavior when C is inconsistent is undefined. Given *any* configuration C, not necessarily consistent, M can be used to decide consistency in polynomial time as follows.

Let C be input to M and run for $p(|C|)$ time. If M fails to output within this time, then C is inconsistent. Otherwise, M has produced some output. Since deciding frictional consistency is in NP, the validity of M's output can be decided in an additional amount of time that is also a polynomial function of $|C|$. If M's output is a valid set of contact forces, then clearly C is consistent. If M's output is invalid, then C must be inconsistent (else M would have output a valid answer). In any event, the consistency of C has been decided in polynomial time.

Since deciding consistency is NP-complete, we conclude that the existence of a polynomial time algorithm for computing contact forces on consistent configurations

would imply that $P = NP$. □

From a simulation viewpoint, this seems to be a depressing result. However, there has been an implicit, unchallenged assumption in the above discussion. The assumption is that we must determine whether or not a configuration is consistent, so that we will know whether impulsive or non-impulsive forces should be applied. This involves a further assumption: it is invalid to apply impulsive forces to configurations that are consistent (and thus have a non-impulsive force solution). In the next chapter, the argument will be made that this second assumption is false. Based on this argument, we will reformulate the problem of computing contact forces so that it is neither necessary nor even desirable to determine whether or not a configuration is consistent. Following that, it will be shown how avoiding the need to determine consistency leads to a practical polynomial time algorithm for computing contact forces for dynamic friction.

Chapter 12

Physical Models

In this section, a physical model for both inconsistency and indeterminacy is presented. We will use this model to develop a computationally practical method of dealing with dynamic friction in Chapter 13. The model in this section was developed in order to understand the behavior of inconsistent and indeterminate configurations. The insights gained from considering this model also suggest a general principle regarding indeterminacy during simulation.

The motivation of a physical model stems from the need to answer the following basic questions. First, what should be the result of a simulation when inconsistency is encountered? As we have seen, the only resolution of inconsistent configurations, such as Figure 10.3a, is the application of an impulsive contact force. What is the physical process which produces such a force? Second, is there a simple physical explanation for the type of indeterminacy that the configuration of Figure 10.2 exhibits?

In the physical world, there is of course no such thing as a perfectly rigid body. For near rigid bodies, contact forces arise as a result of small elastic deformations in the neighborhood of the contact area. The penalty method, as described in Chap-

ter 7, is based on a mathematical method from numerical optimization. However, the penalty method can equally well be viewed (and is, by many) as an attempt to model elastic deformations in a very simple form. In the penalty method, small amounts of inter-penetration model the amount of deformation between bodies, and the penalty force accompanying the inter-penetration models the elastic restoring force. Typically, the normal force is modeled as a linear spring force $-Kd$, where K is the spring constant and d is the amount of inter-penetration. As Chapter 7 shows, the penalty method converges to a correct result as K is increased. It therefore makes sense to consider applying the penalty model, conceptually, to situations involving friction, and studying the limiting behavior. This thesis does not propose the penalty model as embodying a specific physical process. Rather, the penalty model is used as an intuition to help visualize why impulses are necessary in some situations, and to help guide our development of a definition of exactly when and how impulses can be applied to a system without colliding contact.

12.1 A model of inconsistency

Figure 12.1 shows the behavior of the inconsistent configuration of Figure 10.3a when the penalty method is applied. At time t_0, consider the tip of the rod, p_a, to be resting exactly on B, with zero inter-penetration. Since there is no inter-penetration, the normal force is zero. Even though p_a is sliding along B, the friction force is zero since the normal force is zero. Since the only force acting on A is the external gravity force $-mg\hat{n}$, p_a accelerates downwards. At time $t_0 + \Delta t$, p_a has inter-penetrated B by an amount d_1, so a normal force $Kd_1\hat{n}$ acts on A. Since p_a is still sliding, a friction force of $\mu Kd_1\hat{t}$ also acts on A. The net result, from Equation (10–11), is that this causes p_a to accelerate downwards (even faster than before). As the penalty force continues to increase, it causes more inter-penetration

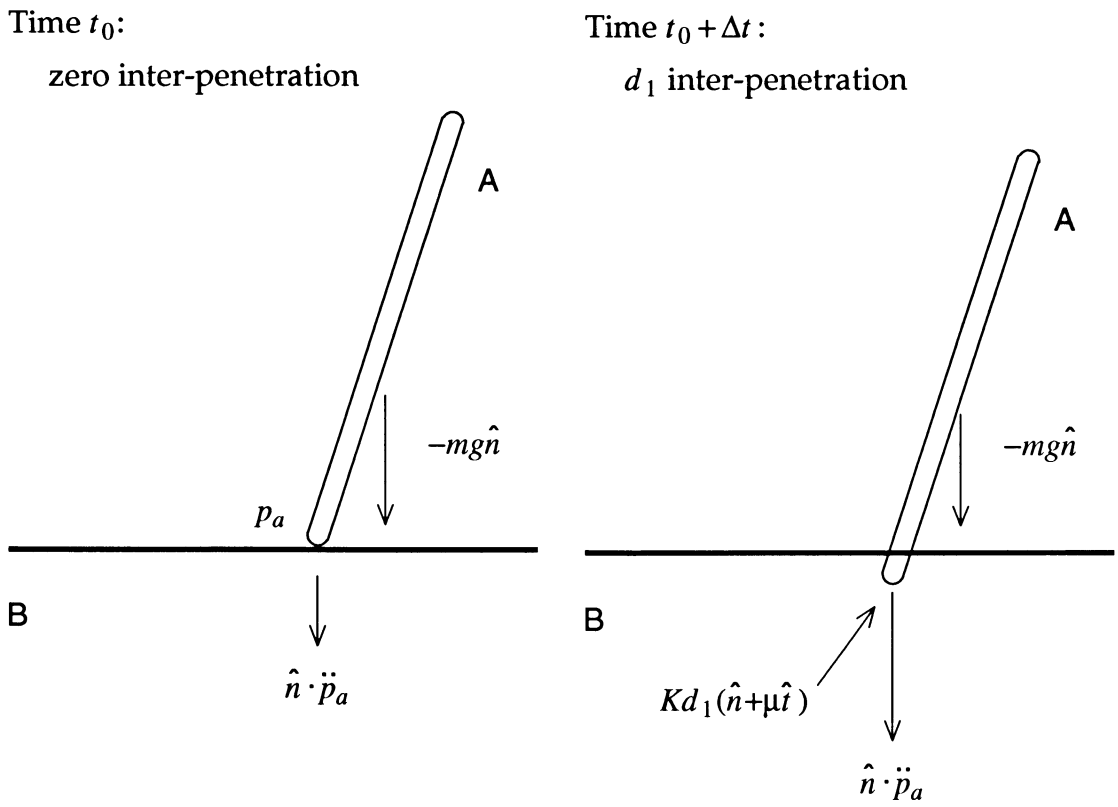


Figure 12.1: A model of indeterminacy. At time t_0, only gravity acts on A. At time $t_0 + \Delta t$, the inter-penetration distance is d_1 and both a penalty and a gravity force act on A, causing p_a's downwards acceleration to increase.

between A and B; a form of positive feedback. Accordingly, both the friction and the normal force increase, and the cycle continues. Since we are trying to model A and B as rigid bodies, the penalty constant K must be allowed to be arbitrarily large. The larger K is, the faster inter-penetration increases and the faster the normal and friction forces build.

Recall that the friction force opposes the sliding motion of A across B. By making K arbitrarily large, the friction force brings p_a to rest (horizontally) in an arbitrarily short time. For example, suppose K is adjusted so that p_a comes to rest within time Δt. Then the amount of inter-penetration is $O(\Delta t^2)$, since the distance traveled by p_a depends quadratically on the time for which it travels. In the limit as K goes to infinity, the contact force on A acts as an impulse and instantaneously brings p_a to rest horizontally, without inter-penetration occurring.

The impulse that brings p_a to rest horizontally produces a configuration in which A and B are colliding (in the normal direction) at p_a. This requires the application of a second impulse, to counter the downwards velocity acquired by p_a. Once the impulses have been applied and p_a is no longer sliding across B, dynamic friction is replaced with static friction. As a result, the net contact force direction is no longer fixed, and the inconsistency no longer exists.

12.2 A model of indeterminacy

Consider the indeterminate configuration of Figure 10.2, which has solutions $f_N = 0$ and $f_N = m$. Using the penalty method, the indeterminacy can be removed by assuming some amount of initial inter-penetration between A and B. If the initial inter-penetration between A and B is zero (Figure 12.2) then no normal force exists at time t_0 and the net acceleration of p_a is upwards. Contact is immediately broken; this behavior is shown in Figure 10.2a. However, if the initial inter-penetration

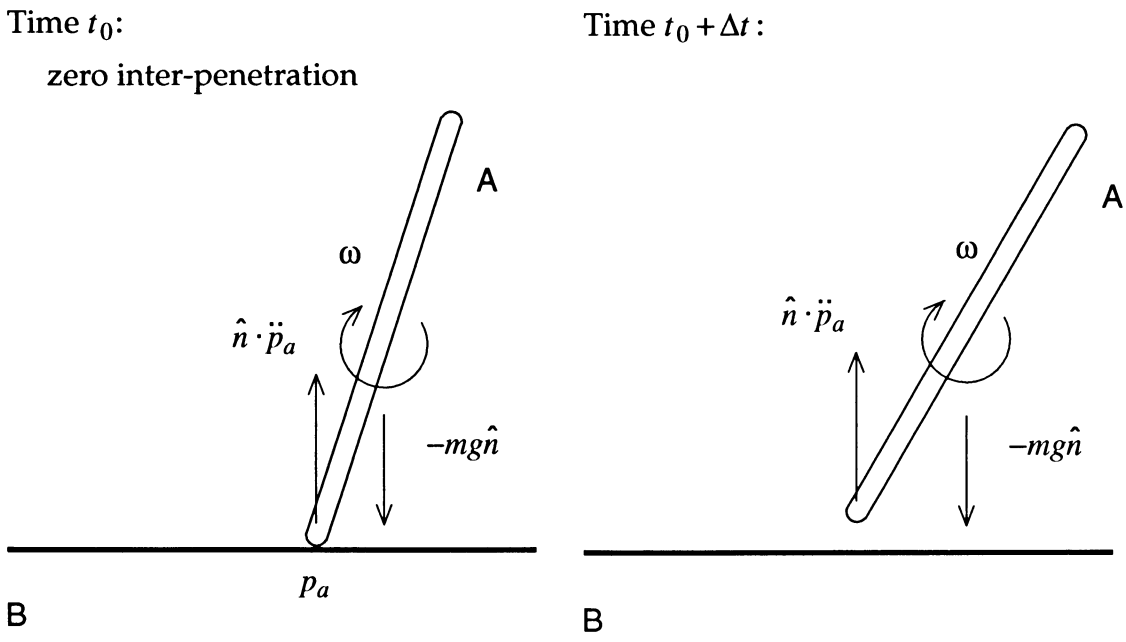


Figure 12.2: A configuration with no initial inter-penetration. Only gravity acts on A initially. The centripetal acceleration of A pulls p_a away from B and contact is broken.

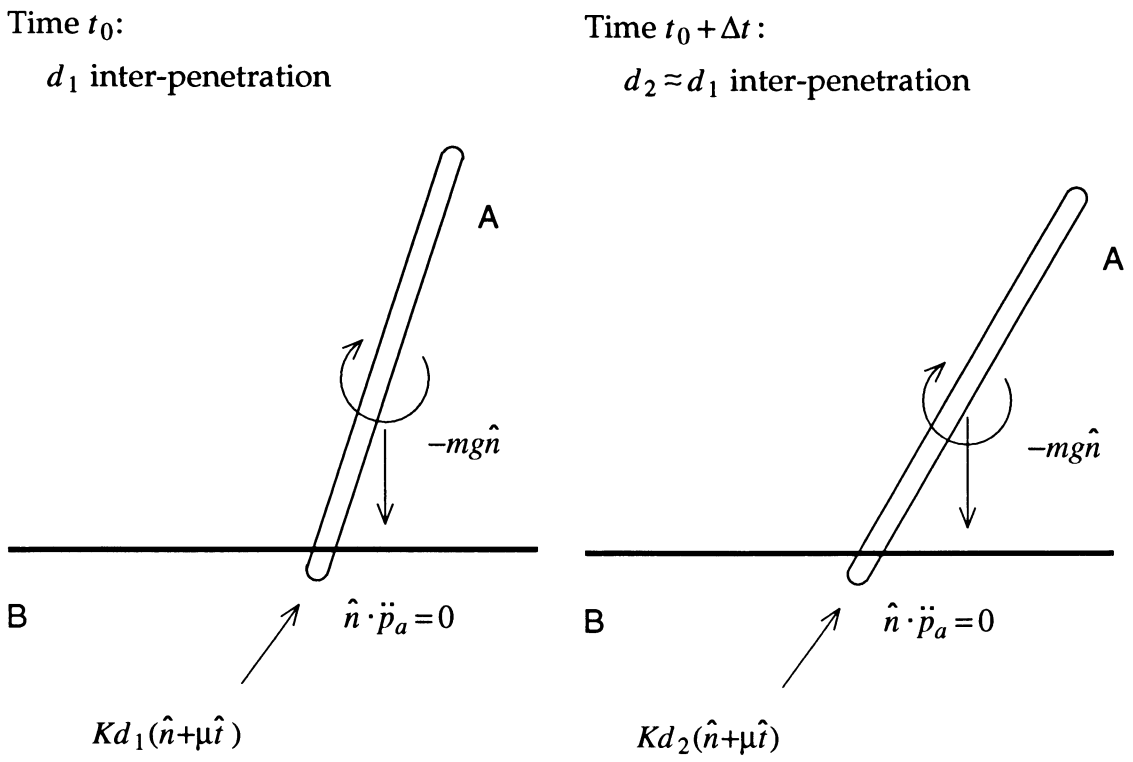


Figure 12.3: A configuration with an initial inter-penetration depth of d_1. Both gravity and a penalty force act on A. Body A slides and falls without breaking contact with B.

produces a normal force magnitude of m, then the normal and friction forces prevent A from breaking contact with B. In Figure 12.3, let the initial inter-penetration d_1 be m/K. Then the normal force magnitude at time t_0 is m. Since p_a is sliding on B, a friction force acts on A as shown. As A falls, maintaining contact with B, the inter-penetration varies smoothly, produce a varying normal force. At time $t_0+\Delta t$, body A still inter-penetrates B by an amount $d_1 \approx d_2$, and the behavior of the configuration is that of Figure 10.2b. Thus, the initial amount of inter-penetration determines which behavior occurs.

The numerical methods we will use to compute forces and impulses do not

model inter-penetration. Instead of determining behavior by initial choice of inter-penetration, we can consider the normal force that existed at a contact point at a time t immediately before t_0 as the initial conditions at time t_0. These initial conditions are used to determine the behavior of bodies. This brings up the obvious question of what initial conditions to use for the first time step of a simulation, or after a discontinuous velocity change of bodies caused by a collision. The choice of initial conditions depends upon the application; if there is no basis for preferring one set of initial normal forces over another, the solution found by the numerical routines solving the contact force equations arbitrarily determines the behavior simulated. We will consider, in a moment, the validity of arbitrarily choosing a path for the simulation to follow. A much deeper question that we will not attempt to address is whether or not following only this one path to the end of the simulation reveals useful information.

12.3 Indeterminacy and inconsistency combined

Let us now consider a configuration that combines indeterminacy and inconsistency so that it is indeterminate whether or not an impulsive solution is required. Consider Figure 12.4. Once again, the combination of gravity and the angular velocity of R_1 is the same as in Figure 11.2 and Figure 11.3. Similarly, a horizontal acceleration of B results in inconsistency. If E_1 is ignored for the moment, then both $f_N = 0$ and $f_N = m_1$ are valid solutions for f_N. The only valid solution for the configuration as a whole though, is $f_N = 0$; a magnitude $f_N = m_1$ pushes B to the left, causing inconsistency.

If we subscribe to the physical model presented in this chapter, then it is clear that whether or not inconsistency occurs depends solely on the initial inter-penetration between R_1 and B. The implication of our physical model then

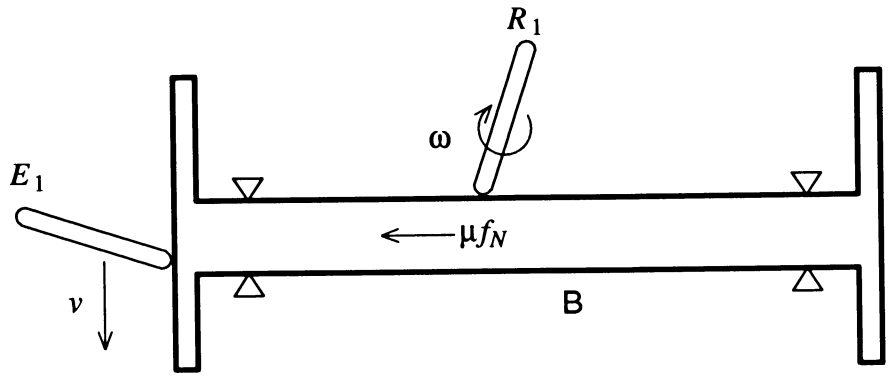


Figure 12.4: A configuration with a single impulse-free solution. The magnitude f_N must be either 0 or m_1 to be valid. However, $f_N = m_1$ causes inconsistency. The only valid solution is $f_N = 0$.

is the following: even though the configuration is consistent, that is, it has an impulse-free solution, there is no *a priori* reason to prefer this impulse-free behavior to the non-impulse-free behavior for this configuration. Stated differently, if we have no bias on the initial conditions of the configuration (that is, the initial inter-penetration distances), then the inconsistency resulting from $f_N = m_1$, and subsequent application of impulsive contact forces must be considered as acceptable a behavior as the application of non-impulsive contact forces resulting from $f_N = 0$. Even though the configuration in Figure 12.4 has only one valid solution of contact forces, ($f_N = 0$), it has two possible behaviors and is thus indeterminate.

12.4 Preferred solutions

By accepting the physical model based on the penalty method, we accept the fact that the two behaviors of Figure 12.4 must be considered equally valid. Let us contrast this model with a principle proposed by Kilmister and Reeve[25] called the *principle of constraints.* The principle states that when computing forces for

a configuration of bodies, impulsive forces should be used only if non-impulsive forces do not exist for the configuration. In essence, the principle of constraints is an assertion that non-impulsive solutions are preferable to solutions involving impulses. This principle seems (at first glance) sound; however, let us consider arguments against the principle of constraints.

From the discussion of the previous section, we can view the *NP*-hardness result of computing contact forces in a different light. By reasoning that solutions involving impulsive contact forces are acceptable even when a non-impulsive solution exists, we have extended the possible behaviors of configurations of bodies. (A formal definition of validity with regard to contact impulses has not been given yet, and will be presented in the next chapter.) If we subscribed to the principle of constraints however, we would first attempt to solve the LCP of Equation (8–7), in order to find a solution of non-impulsive contact forces. This is a search problem. Since some configurations (such as the configuration of Figure 11.3) may possess exponentially many impulsive solutions and only a single non-impulsive solution, it is not surprising that an attempt to search for this single solution leads to *NP*-hardness. Thus, the first argument we might advance against the principle of constraints is that by accepting a bias against solutions with impulsive forces, we are required to perform possibly an exponential amount of work. In contrast, disregarding this bias and accepting both impulsive and non-impulsive solutions turns out to require far less work (Chapter 13).

This is not really an argument, but more of a statement of the consequences of preferring non-impulsive solutions to impulsive solutions. A better argument is that the physical model proposed in this section, which is logically consistent with frictionless behavior, and explains both indeterminacy and inconsistency, has no such preference. In comparison, the principle of constraints is nothing more than a statement of the preference of non-impulsive behavior, without any justifying

explanation. Still, one might reject the physical model presented in this section, and argue from a rational mechanics points of view. That is, impulses have been added to our model of rigid body mechanics to deal with situations in which non-impulsive forces cannot prevent inter-penetration. The principle of constraints is therefore a good principle of constraints is preferable because it always seeks the "simplest" solution; that is, it only extends the axioms of rigid body mechanics to admit impulses when necessary.

One can refute this rational mechanics viewpoint very strongly with the following argument. Suppose that at time $t = 0$, we are faced with an indeterminate configuration, and suppose that there are two behaviors of the system past time zero. Let us call these behaviors of the systems *trajectories* A and B (Figure 12.5). The principle of constraints asserts that we must always avoid impulsive behavior, whenever possible. As we have seen, it is *NP*-hard to determine which trajectory (if either) of A or B avoids impulsive behavior at time zero. But suppose that neither trajectory A and B involves impulsive behavior at time zero, and that we arbitrarily choose A. If at time $t > 0$ along trajectory A we encounter inconsistency, what should we do? According to the principle of constraints, it would logical to back up our simulation to time $t = 0$ and consider trajectory B, for trajectory B might not require impulsive solutions for all $t > 0$. If so, then we see that in fact we really should have chosen trajectory B to start with, for by doing do, we avoid impulses forever, past time zero.

Clearly, this is enormously expensive, computationally, because it forces us to explore potential trajectories, which might themselves branch, to find the trajectory that is most free of impulses. The possible expense is for all practical purposes unbounded, since we can make the choice of the correct trajectory depend upon the outcome of some physical process which is arbitrarily hard to compute. Again, it can be argued that computational expense should not be a factor in rejecting

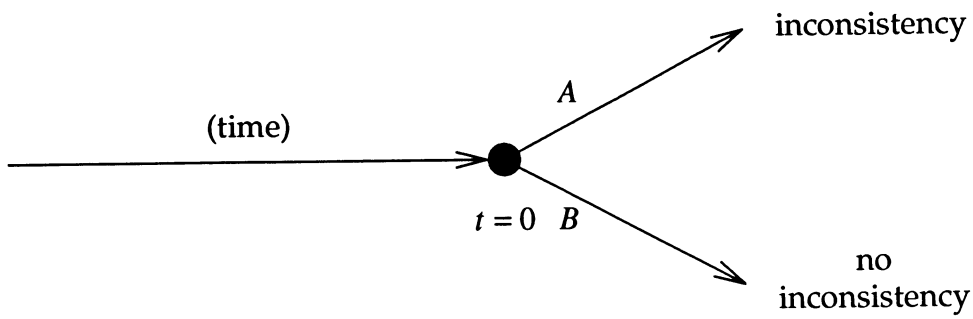


Figure 12.5: A configuration with two different non-impulsive solutions at time zero. If trajectory A is chosen, inconsistency is encountered for some later time $t > 0$, but if B is chosen, no inconsistency is encountered for $t > 0$.

the principle of constraints. However, suppose that an external time-varying force $F(t)$ acts during the course of the simulation. The decision as to whether trajectory A or B best avoids impulses past time zero would then require knowledge of the behavior of $F(t)$ for time $t > 0$. If we chose A or B based on $F(t)$ for $t > 0$, then we violate causality, since events at times $t > 0$ determine our course of action at time zero. In fact, we can create a paradox, by letting the force $F(t)$ be determined according to which trajectory is chosen, which is itself determined by $F(t)$. Thus, the principle of constraints in seeking to always use the smallest axiom set for rigid body mechanics, is an "unnatural" principle: it requires a physical system to be able to look forward in time to determine its action at the present instant.

The above argument clearly generalizes to any physical system with indeterminacy. Any principle that asserts that one sort of behavior is more preferable than another over the course of time leads to a violation of causality, and is thus an unnatural principle. That is not to say however that we cannot prefer one behavior for a particular *instant*. For instance, when we encounter an indeterminacy in a simulation, we might be told that one trajectory is more probable than another. If we regard indeterminacy as simply inadequate knowledge of a system's state to the

point that we cannot predict its actions, we might in fact be given a "tip" as to the correct behavior of the system by an oracular observer who has more knowledge of the system's state than we do. Although we should regard all trajectories as equally possible, they need not all be equally likely.

Given that we are willing to accept both impulsive and non-impulsive solutions as valid solutions for a configuration with dynamic friction, the following question arises. We know that searching for a particular type of solution, namely solutions without impulsive forces, is *NP*-hard. How difficult is it to find either a solution with or without impulsive forces? That is, how much time is required of an algorithm that computes either an impulsive or a non-impulsive solution for a configuration, without preference? Such an algorithm could not be used to decide if a configuration was consistent, since the algorithm could return an impulsive solution whether or not the configuration was consistent. The use of such an algorithm to compute contact forces (and thus choose an arbitrary trajectory for the simulation) would avoid the *NP*-complete problem of deciding consistency. In the next chapter, we will consider an algorithm with this property.

Chapter 13

Computing Valid Contact Forces and Impulses

Before computational methods can be considered, a definition of validity for contact impulses is needed. In order for a definition of validity to be useful, all inconsistent configurations should have a set of contact impulses satisfying the definition. An analysis of the computational method presented in Section 13.2 will show that the definition of validity presented satisfies this condition.

13.1 Defining valid contact impulses

In the penalty method interpretation of Figure 12.1 an impulse occurred because no matter how strong the normal force became, it was insufficient to prevent interpenetration. As a result, after the contact impulse was applied, the relative velocity of the bodies at the contact point was directed inwards. Since contact impulses may need to be applied to configurations involving more than one contact point, validity must be defined for a set of contact impulses. For example, in Figure 12.4, if the $f_N = m_1$ behavior is chosen, a contact impulse should occur between E_1 and B.

However, there should be no contact impulse between R_1 and B.

We will call a set of contact impulses valid under the following two conditions. The first condition is that the contact impulses must convert at least one of the contact points with dynamic friction to static friction. The reasoning here is that the impulse has occurred because the initial conditions of normal forces have caused inconsistency, as in Figure 12.4, and the inconsistency is removed by converting a contact point from dynamic to static friction.

The second condition is that every contact point at which a contact impulse occurs must end up with a non-positive relative normal velocity; that is, after the contact impulses are applied, bodies should *not* be separating wherever contact impulses occurred. The justification for this is that the contact impulses occur only when the normal force grows without bound to oppose inter-penetration. Intuitively, valid contact impulses are the limiting result of increasing normal forces without bound under the penalty method. If bodies are separating at a contact point after contact impulses are applied, then the normal force at the contact point should not have grown without bound into a contact impulse.[1]

[1] For planar configurations, this definition is valid, because a contact with dynamic friction cannot change its direction of sliding without having zero sliding velocity. For three-dimensional configurations, the sliding velocity direction can change without a contact changing from dynamic to static friction. Thus, for three-dimensional configurations, we would want to find contact impulses that satisfy the second condition. We would then need to differentially apply the contact impulses, and account for the change of sliding velocity direction; essentially the same problem as dealing with frictional impulses for three-dimensional collisions. Thus, the results of this chapter are valid for planar configurations, and are a starting point for three-dimensional systems. If one adopts the *ad hoc* assumption that the sliding velocity direction does not change during a collision for three-dimensional systems, then the results of this chapter can be applied without complication to three-dimensional systems.

13.2 Lemke's algorithm

Having defined validity for contact impulses, we can consider computational methods for computing contact forces and impulses. Given that computing contact forces alone is hard, how can either contact forces or impulses be computed efficiently? We begin by considering the LCP of Equation (8–7). One of the first algorithms for solving linear complementarity problems was introduced by Lemke[27] and is known as Lemke's algorithm. Lemke's algorithm is a pivoting method, similar to the simplex method of linear programming. The algorithm is exponential in the worst case, but has an expected running time polynomial in n. Lemke's algorithm progresses, like the simplex method, by trying various descent directions. If an LCP is PSD and has no solution then Lemke's algorithm will at some point encounter an *unbounded ray*; that is, a descent direction along which one can travel infinitely far without making any progress. Otherwise, if a PSD LCP has a solution, then no unbounded ray exists for that LCP, and Lemke's algorithm terminates by finding a solution to the LCP. The algorithm is viewed as a practical solution method to the problem of solving PSD LCP's.

However, for non-PSD LCP's, Lemke's algorithm is not guaranteed to terminate correctly (although it still takes only expected polynomial time to do so). For a non-PSD LCP, if there is no solution, Lemke's algorithm terminates by encountering an unbounded ray. Unfortunately, if there is a solution, the algorithm is not guaranteed to find it. For non-PSD LCP's with solutions, Lemke's algorithm terminates either by finding a solution or by encountering an unbounded ray.[2] As a result, Lemke's algorithm is not suitable for solving non-PSD LCP's.

However, whenever Lemke's algorithm terminates by encountering an unbounded

[2] Encountering an unbounded ray when there is a solution is roughly analogous to getting stuck at a non-global minimum in a non-convex minimization problem.

ray, it has found an n-vector $\mathbf{z}$ with the property

$$\mathbf{z} \geq \mathbf{0} \quad \text{and} \quad z_i > 0 \text{ implies } (\mathbf{Az})_i \leq 0 \quad (1 \leq i \leq n) \tag{13–1}$$

where $(\mathbf{Az})_i$ is the ith component of the vector $\mathbf{Az}$. Why is this property of interest?

Suppose we let $\mathbf{z}$ represent a set of contact impulses in the same way that $\boldsymbol{f}_\mathbf{N}$ represents a set of contact forces, and apply those contact impulses to the configuration. By the definition of impulse, the change in relative normal velocity at each contact point is $(A\mathbf{z})_i$. Since the relative normal velocity at each contact point is assumed to be zero before application of the impulse, the vector of relative normal velocities after applying the impulse is simply $\mathbf{Az}$.

Thus, the contact impulses indicated by $\mathbf{z}$ satisfy the second condition of validity: every contact point subject to a non-zero contact impulse $z_i > 0$ ends up with a non-positive relative normal velocity $(\mathbf{Az})_i \leq 0$. In order to completely satisfy the definition of validity, $\mathbf{z}$ must be scaled upwards from zero until it causes a contact point with dynamic friction to be converted to static friction. A subtle point to note is that if $\mathbf{z}$ specified contact impulses only at the *frictionless* contact points, then $\mathbf{z}$ could not satisfy the first condition condition for valid contact impulses (although it might satisfy the second). In order for $\mathbf{z}$ to represent valid contact impulses, $\mathbf{z}$ must have the property that for at least one $1 \leq i \leq n$, the ith contact point has dynamic friction and $z_i > 0$. A proof that Lemke's algorithm either solves the LCP or finds a vector $\mathbf{z}$ such that $\alpha\mathbf{z}$ represents valid contact impulses for some $\alpha > 0$ is given in Appendix B. After application of the contact impulse represented by $\alpha\mathbf{z}$, some real collisions (that is, collisions with positive normal approach velocity) occur and must be dealt with.

The behavior of Lemke's algorithm exactly matches our new view of the problem of computing contact forces. If the configuration has no valid contact impulse solutions, Lemke's algorithm cannot terminate with a ray $\mathbf{z}$ and must therefore

find a valid contact force solution. For inconsistent configurations, no valid contact force solution exists, so Lemke's algorithm must terminate with a ray $\mathbf{z}$, providing a contact impulse solution. For configurations with both a valid force and impulse solution, Lemke's algorithm will terminate by computing one or the other.

Note that the definition of valid contact impulses does not add any new solutions to frictionless systems. To see this, consider an arbitrary frictionless system, and suppose that a vector $\mathbf{z}$ satisfies the condition of Equation (13–1). Then

$$\mathbf{z}^T\mathbf{A}\mathbf{z} = \sum_{i=1}^{n} z_i(\mathbf{A}\mathbf{z})_i \leq 0. \tag{13–2}$$

But since the system is frictionless, $\mathbf{A}$ is PSD. Since a PSD matrix satisfies $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$ for any vector $\mathbf{x}$, Equation (13–2) implies that

$$\mathbf{z}^T\mathbf{A}\mathbf{z} = 0. \tag{13–3}$$

Since $\mathbf{A}$ is PSD, this in turn implies that $\mathbf{A}\mathbf{z} = \mathbf{0}$. Thus, any vector $\mathbf{z}$ that fits the first condition of valid contact impulses is a vector in the null-space of A. Since frictionless systems are determinate, given two vectors $\mathbf{x}$ and $\mathbf{y}$ such that $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{y}$, application of the impulses (or forces) indicated by $\mathbf{x}$ has the same effect as the application of the impulses (or forces) indicated by $\mathbf{y}$. Since $\mathbf{A}\mathbf{z} = \mathbf{0}$, this implies that the application of the impulses indicated by $\mathbf{z}$ has absolutely no effect at all on the system. Thus, the definition of valid contact impulses is such that it adds no new solutions to frictionless systems.

Although Lemke's algorithm runs, practically speaking, in polynomial time, this is not a proof that finding either valid contact forces or impulses is a polynomial time problem. From a practical standpoint, though, Lemke's algorithm provides an efficient algorithm for computing valid contact forces or impulses. The computational complexity of either solving an LCP or finding a vector $\mathbf{z}$ satisfying Equation (13–1) is unknown; but it would be surprising if this could not be done in polynomial time.

13.3 Continuity of solutions

One implementation issue we need to be concerned with is the continuity of the solutions chosen for $\boldsymbol{f}_{\mathbf{N}}$. Consider a configuration with two possible solutions $\boldsymbol{f}_{\mathbf{N}}$ that produce different accelerations of the bodies. Suppose that we (perversely) alternate between the two solutions at each time step, causing the accelerations of the bodies to be discontinuous. This will cause the ordinary differential equation solver to break down, since it expects to be integrating a continuous function.

In Section 8.1 where configurations were frictionless, this problem did not occur because all solutions of Equation (8–7) produced the same acceleration. When friction is introduced this is no longer true, and care must be taken to make $\boldsymbol{f}_{\mathbf{N}}$ (at least) piecewise continuous over time. Recall that Section 8.2 introduced the concept of computing $\boldsymbol{f}_{\mathbf{N}}$ by solving a non-singular linear system of equations as indicated by an index set L. This computation method for $\boldsymbol{f}_{\mathbf{N}}$ may be regarded as a time saving luxury for frictionless configurations, but it is essential in simulating configurations with friction. Since $\mathbf{A}$ and $\mathbf{b}$ depend on spatial and velocity variables, they are continuous functions of time (in the absence of impulses). As a result, the system of equations produced from L is continuous as long as L remains constant. Because of this, the $\boldsymbol{f}_{\mathbf{N}}$ vectors computed with this method are piecewise continuous, with discontinuities occurring whenever L changes. The integrator is of course informed whenever L changes.

Chapter 14

Static Friction

The results in this thesis concerning the complexity of static friction are much less comprehensive than the results for dynamic friction. We will indulge in some speculation concerning complexity results for static friction, and then discuss some methods for approximating static friction forces for simulation purposes. The approximation methods are suitable for some applications (such as computer animation), but may be unsuitable with regards to other applications.

14.1 Complexity of static friction

Consider once more the ith contact point of a configuration. The normal force and tangential friction force magnitudes at this point are f_{Ni}, f_{xi} and f_{y_i}. The corresponding acceleration magnitudes are a_{Ni}, a_{xi} and a_{y_i}. Since we already know that the presence of dynamic friction leads to *NP*-hardness, let us study configurations with only static friction and no dynamic friction.

In Chapter 9, the conditions for static friction were presented. These conditions

can be written as

$$\left\{\begin{array}{c} f_{Ni} \geq 0 \text{ and} \\ a_{Ni} \geq 0 \text{ and} \\ f_{Ni} a_{Ni} = 0 \text{ and} \\ f_{xi}^2 + f_{yi}^2 \leq (\mu f_{Ni})^2 \end{array}\right. \quad \text{and either} \quad \left\{\begin{array}{c} a_{xi} = a_{y_i} = 0 \text{ or} \\ f_{xi}^2 + f_{y_i}^2 = (\mu f_{Ni})^2 \text{ and} \\ f_{xi} a_{xi} + f_{y_i} a_{y_i} \leq 0. \end{array}\right. \tag{14–1}$$

The first questions we might ask are: does Equation (14–1) always have a solution? Does Equation (14–1) always have a unique solution? Lötstedt[28] gives an example of a one contact point configuration with static friction that does not have a unique solution. The example given is the sliding rod of Figure 10.2, but with the linear velocity adjusted so that the tip of the rod is motionless with respect to the ground. Once again, the rod is free to either rotate off the ground or remain in contact. Thus, Equation (14–1) does not necessarily have a unique solution. Whether Equation (14–1) always has even one solution is unknown. So far, no examples of configurations with static friction resulting in inconsistency are known (excepting the trivial case when the configuration has unsatisfiable kinematic constraints). This is particularly striking in light of the fact that inconsistency for dynamic friction can occur for one contact point configurations. For static friction, we can however prove that every one contact point configuration has a solution. The proof technique makes use of geometrical constructs based on work by Wang and Mason[34].

THEOREM 4. *All two-dimensional one contact point configurations with static friction have valid solutions.*

PROOF. For two dimensional configurations, we can describe the frictional force and the tangential accelerations in terms of just f_{xi} and a_{xi}. The analogue of

Equation (14–1) in two dimensions is therefore

$$\left\{\begin{array}{c} f_{Ni} \geq 0 \text{ and} \\ a_{Ni} \geq 0 \text{ and} \\ f_{Ni} a_{Ni} = 0 \text{ and} \\ -\mu f_{Ni} \leq f_{xi} \leq \mu f_{Ni} \end{array}\right. \quad \text{and either} \quad \left\{\begin{array}{c} a_{xi} = 0 \text{ or} \\ |f_{xi}| = \mu f_{Ni} \text{ and} \\ f_{xi} a_{xi} \leq 0. \end{array}\right. \tag{14–2}$$

For a one contact configuration, we can express the normal acceleration a_N and the frictional acceleration a_x in terms of the normal force f_N and the frictional force f_x as

$$\begin{aligned} a_N &= B f_N + C f_x + F \\ a_x &= C f_N + E f_x + G \end{aligned} \tag{14–3}$$

where B, C, D, and E satisfy[28]

$$B > 0, \quad E > 0 \qquad \text{and} \qquad BE > C^2. \tag{14–4}$$

Additionally, we can assume $C < 0$ by reversing the sense of the tangent direction along which f_x and a_x are measured, if necessary. (If $C = 0$ then Equation (14–3) decouples and it is trivial to show Equation (14–2) always has a solution.) From this, it is easily shown that

$$\frac{-E}{C} > \frac{-C}{B}. \tag{14–5}$$

Using these facts, we show that Equation (14–2) always has a solution.

If $F \geq 0$ then Equation (14–2) has the trivial solution $f_N = f_x = 0$. Otherwise, $F < 0$. In this case, we claim a solution with $f_N > 0$, $a_N = 0$ and one of the following three cases always exists:

(i) $a_x = 0$ and $-\mu f_N \leq f_x \leq \mu f_N$

(ii) $a_x > 0$ and $f_x = -\mu f_N$

(iii) $a_x < 0$ and $f_x = \mu f_N$.

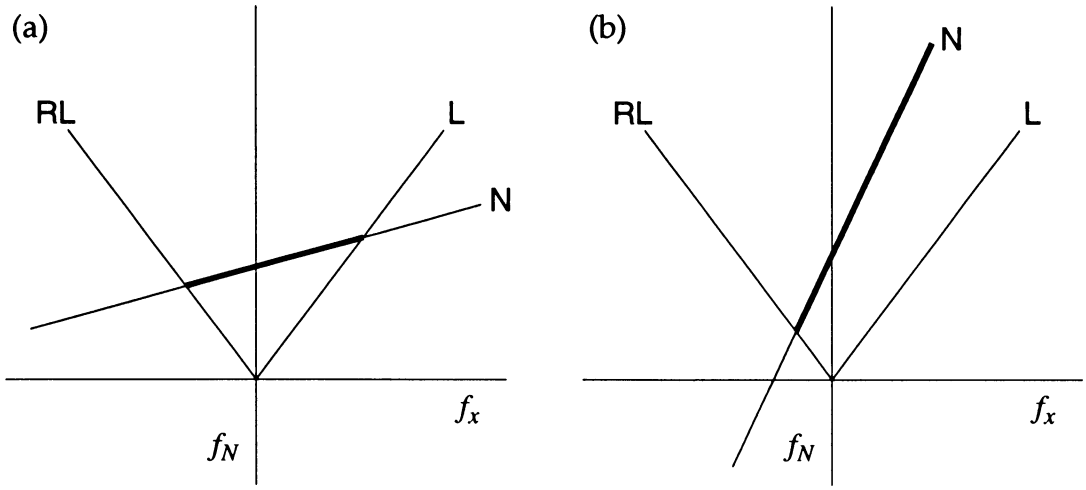


Figure 14.1: Force space. (a) Possible solutions for P lie on N between L and RL. (b) Possible solutions for P lie on N to the right of N's intersection with RL.

To verify this claim, we use a geometrical method based on the impulse space described by Wang and Mason[34].

Consider Figure 14.1 where f_N is plotted vertically and f_x is plotted horizontally. We are searching for a point $P = (f_N, f_x)$ in this space that satisfies both Equation (14–2), and $f_N > 0$ and $a_N = 0$. Several lines are identified in this space. First, the line $Bf_N + Cf_x + F = a_N = 0$ is labeled N. Next, the lines $f_x = \mu f_N$ and $f_x = -\mu f_N$ are labeled L and RL respectively. Since $F < 0$, $B > 0$ and $C \leq 0$, N is either horizontal or has positive slope and crosses the f_N axis above the fx axis. Because of this, N always intersects RL and may (Figure 14.1a) or may not (Figure 14.1b) intercept L. In order for a point P to satisfy $f_N > 0$, $a_N = 0$ and $-\mu f_N \leq f_x \leq f_N$, P must lie on N above the horizontal axis, and on or between L and RL. Points satisfying these conditions are indicated by the bold portion of line N in Figure 14.1.

Suppose now that N crosses both L and RL, and consider Figure 14.2. The line

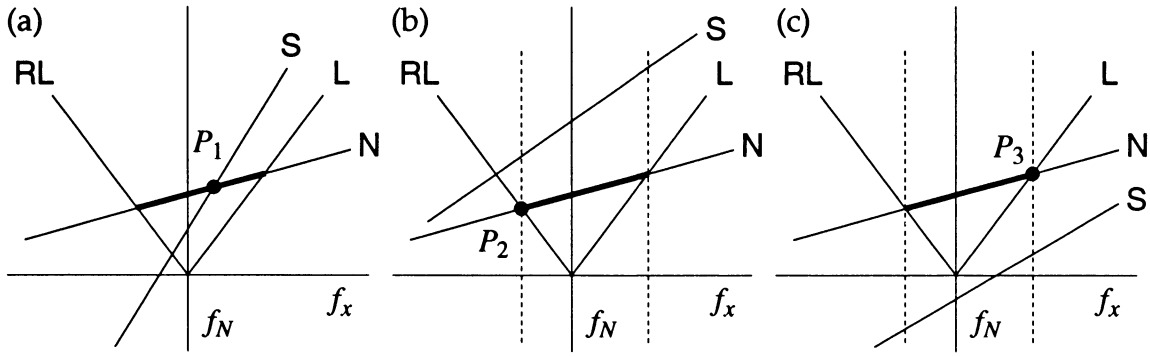


Figure 14.2: Line N intersects L and RL. (a) S intersects N and point P_1 satisfies case (i). (b) S lies above N between L and RL. Points lying below S have $a_x > 0$; point P_2 satisfies case (ii). (c) S lies below N between L and RL. Points lying above S have $a_x < 0$; point P_3 satisfies case (iii).

$Cf_N + Ef_x + G = a_x = 0$ is labeled S. Any point P on S results in zero tangential acceleration. From the definition of S and since $C < 0$, points lying *above* S satisfy $a_x < 0$ while points lying *below* S satisfy $a_x > 0$. Suppose S intersects N between L and RL (Figure 14.2a). Then the intersection point P_1 falls under case (i) and is a solution. If however S does not intersect N between L and RL, then either S lies above N as in Figure 14.2b, or below N as in Figure 14.2c. For Figure 14.2b, the point P_2 is a solution under case (ii) since it satisfies $f_x < 0$ and $a_x > 0$. In Figure 14.2c, P_3 satisfies $f_x > 0$ and $a_x < 0$ and is a solution under case (iii).

The only situation left to consider is if N intersects only RL and not L (Figure 14.3). Suppose that S lies above N's intersection with RL (Figure 14.3a). Since points below S satisfy $a_x > 0$, the point P_4 is a solution under case (ii). Otherwise, S lies below N's intersection with RL (Figure 14.3b). Since S has slope $-E/C$ and N has slope $-C/B$, from Equation (14–5), S has a steeper slope. Therefore, S must intersect N somewhere between L and RL at P_5; this solution

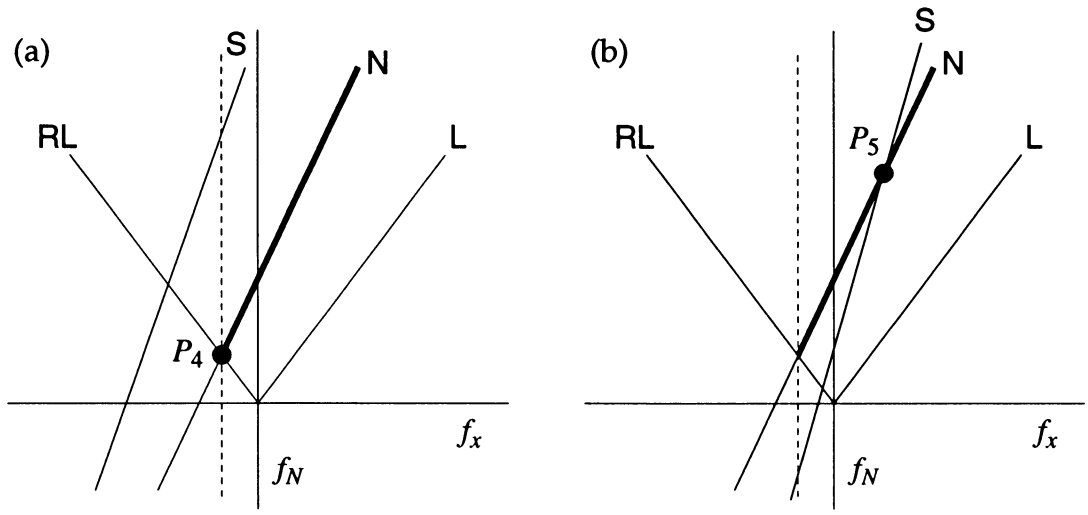


Figure 14.3: Line N intersects RL, but not L. (a) S lies above the intersection of N and RL; point P_4 satisfies case (ii). (b) S lies below the intersection of N and RL. Since S's slope is greater than N's, S must intersect N somewhere between L and RL. The intersection point, P_5 satisfies case (i).

falls under case (i).

Having verified our claim that that for $F < 0$ one of cases (i), (ii) or (iii) always holds, we conclude that all two-dimensional one contact point configurations with static friction have valid solutions. □

This result is easily extendible to three dimensions. The same geometric intuition is used, except that N and S are planes and the lines L and RL are replaced with the cone $f_x^2 + f_y^2 = (\mu f_N)^2$.

We can speculate that Equation (14–1) does in fact always have a solution. Having shown that all one contact point configurations with static friction have a solution, we can contemplate an inductive proof that all n contact point configurations with static friction have a solution. Theorem 4 would be the base case, and the inductive step might be as follows. Assume all n contact point configurations with friction have solutions, and consider an $n + 1$ contact point configuration.

Let one contact point be arbitrarily called the "new" contact point. We will insert an external force at this new contact point that acts equally and opposite on the contacting bodies at this point. Henceforth we ignore this new contact point, and consider the configuration to be simply an n contact point configuration. For any external force we insert at the new contact point, there is a valid solution of contact forces at the other n contact points, since all n contact point configurations have solutions.

The idea then is that we slowly increase the strength of the external force acting at the new contact point. As we do this, the contact forces at the other n contact points continually readjust to form a valid solution. Eventually, we should reach a point where the external force at the new contact point is just sufficient to prevent inter-penetration without being strong enough to cause separation at the new contact point. At this point, if we treat the external force as a contact force at the new contact point, we will have a valid solution for the $n+1$ contact point configuration. This will complete the inductive step. However, the above is only speculation; there are still some unresolved difficulties associated with this proof.

What can we actually say about the complexity of computing contact forces? First, note that the problem of computing contact forces for two-dimensional configurations with friction lies in *NP*. Just as any solution to the LCP of Equation (8–7) could be written in terms of the solution of a linear system, the same is true for a solution to Equation (14–2). As a result, any solution of Equation (14–2) must be rational and of polynomial length[52]. Thus, finding a solution to Equation (14–2) lies in *NP* since a solution to the equation can be verified in polynomial time.

Unfortunately, for three-dimensional configurations, the Coulomb friction constraint $f_{xi}^2 + f_{yi}^2 \leq (\mu f_{Ni})^2$ brings up the possibility that a solution might involve irrational quantities. We therefore cannot say that finding a solution to Equation (14–1) lies in *NP*. Practically speaking however, we would not expect

finding a solution to Equation (14–1) to be any harder than finding a solution to Equation (14–2). Laying aside the fact that solving Equation (14–1) might not lie in *NP*, can we at least express solutions to Equation (14–1) in terms of a convex minimization problem, as was done for frictionless configurations? Convex minimization problems, even when not lying in *NP*, usually admit practical solutions by some sort of descent procedure. Even if we could not exactly classify the difficulty of solving Equation (14–1), phrasing it as a convex minimization problem would go a long way towards describing the inherent complexity. The answer to this question is, unfortunately, "no". The configuration described by Lötstedt[28] that does not possess a unique solution has in fact exactly two distinct solutions. Since the solution set of a convex minimization problem is always a single connected component, the solution set of Equation (14–1) cannot be posed in terms of a convex minimization problem.

Should we believe then that computing contact forces for configurations with static friction is *NP*-hard? We cannot say, but we can again speculate. If we were able to find a configuration that had no solution, then we could probably form a reduction similar to the reduction of Chapter 11 to show that computing contact forces is *NP*-hard. Suppose however that our speculation that configurations with static friction always have a solution is correct. As previously mentioned, computing contact forces for two-dimensional configurations lies in *NP*. If all configurations with static friction have solutions, the decision problem of whether or not a two-dimensional configuration with static friction has a solution is trivially in P (polynomial time), since the answer is always "yes". Would this mean that contact forces could also be computed in polynomial time?

So far, an example of an *NP* decision problem lying in P that requires more than polynomial time to actually find a witness is unknown. Indeed the existence of such a problem would constitute a proof that $P \neq NP$, the reasoning being that

if $P = NP$, all NP witness-finding problems can be solved in polynomial time. If in fact all configurations with static friction had solutions, showing that contact forces could not be computed in polynomial time would be equivalent to showing that $P \neq NP$. Thus, if all configurations had solutions, it would probably be more fruitful to attempt to construct a polynomial time algorithm for computing contact forces than to attempt to disprove the existence of such an algorithm.

14.2 Approximation methods

We will conclude our discussion of static friction by considering three different methods for approximating static friction forces. The first method is due to Lötstedt[30] and appears to be the only published approximation method for contact forces with static friction that does not involve the penalty method. The second two methods are new. For all three methods, we express tangential and normal accelerations as linear functions of the contact forces by writing

$$\mathbf{a} = \mathbf{A}\boldsymbol{f} - \mathbf{b} \tag{14–6}$$

where

$$\mathbf{a} = (a_{N1}, a_{x1}, a_{y_1}, \ldots, a_{Nn}, a_{xn}, a_{y_n})^T \tag{14–7}$$

and

$$\boldsymbol{f} = (f_{N1}, f_{x1}, f_{y_1}, \ldots, f_{Nn}, f_{xn}, f_{y_n})^T. \tag{14–8}$$

The matrix $\mathbf{A}$ is a $3n \times 3n$ matrix and $\mathbf{b}$ is a $3n$-vector.

14.2.1 Lötstedt's approximation

Lötstedt's approximation is a very simple modification of the Coloumb law of friction, which very cleverly achieves two difficult goals. First, it allows dynamic

friction problems to be solved by convex QP's or LCP's. Second, it converts the static friction conditions into a form that is also solvable by quadratic programming; moreover, the quadratic program produced is convex. This is a very elegant transformation!

For dynamic friction, Lötstedt sets the dynamic friction force to have a magnitude of μ times the normal force magnitude *of the previous time step.* This makes the dynamic friction force a known external force at the current time step. Since only unknown normal forces and known external forces act on the bodies now, the configuration has a convex LCP. Thus, Lötstedt's modification does away with both indeterminacy and inconsistency, and produces an LCP that can be efficiently solved (since the LCP is convex).

For static friction, Lötstedt sets the upper bound on the magnitude of the static friction force to μ times the normal force magnitude of the previous time step. This allows static friction and normal forces to be solved by quadratic programming in the following manner. For a given time step, let $\tilde{f}_{Ni}$ denote the normal force at the ith contact point from the *previous* time step. Lötstedt's modification produces a system of equations of the form

$$\begin{cases} f_{Ni} \geq 0 \text{ and} \\ a_{Ni} \geq 0 \text{ and} \\ f_{Ni} a_{Ni} = 0 \text{ and} \\ -\mu \tilde{f}_{Ni} \leq f_{xi} \leq \mu \tilde{f}_{Ni} \text{ and} \\ -\mu \tilde{f}_{Ni} \leq f_{yi} \leq \mu \tilde{f}_{Ni} \text{ and} \\ a_{xi}(\mu \tilde{f}_{Ni} - |f_{xi}|) = 0 \text{ and } f_{xi} a_{xi} \leq 0 \text{ and} \\ a_{yi}(\mu \tilde{f}_{Ni} - |f_{yi}|) = 0 \text{ and } f_{yi} a_{yi} \leq 0. \end{cases} \tag{14–9}$$

The approximation is appealing because it still enforces a very important property of static friction: the friction force achieves its upper bound and is opposite the tangential acceleration whenever it is unable to prevent sliding. Still, the above

conditions still appear fairly complex, and certainly not the conditions of a quadratic program.

It turns out however that the QP

$$\underset{\boldsymbol{f}}{\text{minimize}}\ \tfrac{1}{2}\boldsymbol{f}^T\mathbf{A}\boldsymbol{f} - \mathbf{b}^T\boldsymbol{f} \quad \text{subject to} \quad \left\{ \begin{array}{c} f_{Ni} \geq 0 \\ -\mu\tilde{f}_{Ni} \leq f_{xi} \leq \tilde{f}_{Ni} \\ -\mu\tilde{f}_{Ni} \leq f_{yi} \leq \tilde{f}_{Ni} \end{array} \right\} \tag{14–10}$$

has KKT conditions that match the conditions of Equation (14–9), so a solution of Equation (14–10) satisfies the conditions of Equation (14–9). Thus, the fairly complex conditions of Equation (14–9) have the highly desirable property of being solvable by quadratic programming. (Note that Equation (14–9) does not appear to be solvable as an LCP though.) Unfortunately though, when $\mathbf{A}$ is not symmetric, Equation (14–10) has an entirely different set of KKT conditions that do not match Equation (14–9). For configurations with dynamic friction, even if $\mathbf{A}$ is still PSD (as often happens when the frictional coefficient μ is small) it is rarely (if ever) symmetric. This means that we cannot easily extend this approximation to handle the unmodified dynamic friction law, because dynamic friction causes $\mathbf{A}$ to become asymmetric.

Additionally, there is the question of the decoupling of the $\hat{t}_x$ and $\hat{t}_y$ axes, and the replacement of f_{Ni} with $\tilde{f}_{Ni}$. The effects of decoupling of the axes can possibly be minimized by a wise choice of vectors $\hat{t}_x$ and $\hat{t}_y$ for each contact point (so that one of f_{xi} or f_{yi} is near zero, for each contact point), but the replacement of f_{Ni} with $\tilde{f}_{Ni}$ is more serious. It is unclear how one produces a suitable $\tilde{f}_{Ni}$ for the initial time step (or after a discontinuity in the configuration's state). One possibility is to pick arbitrary initial values for the $\tilde{f}_{Ni}$ (such as zero), solve the system, and update $\tilde{f}_{Ni}$ with the newly computed f_{Ni}, and iterate. However, this is method will not necessarily converge; for the one-point configuration of Figure 10.1, the magnitude f_{Ni} will in fact diverge to infinity. Likewise, consider Figure 10.1 with a reversed

tangential velocity, so that the friction force is in effect accelerating the contact point p_a "upwards". Suppose $\tilde{f}_{N_i}$ is set to zero, and f_{N_i} is solved for; on the next iteration, we set $\tilde{f}_{N_i} = f_{N_i}$. Depending on the value of g and the value of θ, this new value of $\tilde{f}_{N_i}$ could be sufficiently large that it causes contact between the rod and the ground to break. If this happens, the new value of f_{N_i} will be zero, since contact is breaking, and the following iteration will start with $\tilde{f}_{N_i}$ set to zero as well. The iterative process then continues, with f_{N_i} and $\tilde{f}_{N_i}$ oscillating between two distinct values.

It is difficult to quantitatively describe the effects of the approximation. In situations where neither indeterminacy nor inconsistency occurs, perhaps the approximation has little effect on the outcome of a simulation. As a trivial example, if we set $\mu = 0$, then the approximation has no effect at all. We can imagine that as we slowly increase μ, the effects of the approximation might not be apparent until μ takes on a suitably large value. But for situations with inconsistency, Lötstedt's approximation produces a non-impulsive solution where such a solution did not exist previously. In this situation, the approximation must clearly have a large effect on the simulation.

14.2.2 Dynamic friction approximation

The second approximation method we consider has the following virtues. It is extremely simple to implement, very robust, and handles dynamic friction. The approximation is similar in concept to the penalty method of computing normal forces based on inter-penetration depth. It avoids some of the difficulties that make the penalty method unsuitable as a simulation method, but it does share the major flaw of all penalty methods: it is an *ad hoc* description of a physical process.

The second method is as follows. In order to determine whether static friction or

dynamic friction should occur at a contact point, the simulation program must have some threshold value ϵ. If the tangential velocity $(v_{xi}^2 + v_{y_i}^2)^{1/2} \geq \epsilon$, then dynamic friction occurs. Otherwise static friction occurs. Since the dynamic friction force is calculated by

$$f_{xi} = \mu f_{Ni} \frac{-v_{xi}}{\sqrt{v_{xi}^2 + v_{y_i}^2}} \qquad \text{and} \qquad f_{y_i} = \mu f_{Ni} \frac{-v_{y_i}}{\sqrt{v_{xi}^2 + v_{y_i}^2}} \tag{14–11}$$

we approximate static friction as

$$f_{xi} = \frac{\sqrt{(v_{xi}^2 + v_{y_i}^2)}}{\epsilon} \mu f_{Ni} \frac{-v_{xi}}{\sqrt{v_{xi}^2 + v_{y_i}^2}} = \mu f_{Ni} \frac{-v_{xi}}{\epsilon} \tag{14–12}$$

and similarly for f_{y_i}. Thus, we approximate static friction as a dynamic friction force that varies in magnitude from zero to an upper limit of μf_{Ni} as the tangential speed varies from 0 to ϵ. The method of Section 13.2 can be used to compute the contact forces, since this is now entirely a dynamic friction problem.

In this approximation however, static friction occurs only when the tangential velocity is non-zero. Bodies must acquire some small amount of "crawl" in order to maintain a static friction force. This is similar to the penalty method, where bodies must acquire some degree of inter-penetration for a sufficient normal force to exist. However, in the penalty method, it is necessary to increase the spring constant K (Chapter 12) without bound as the mass of bodies increases. Our approximation method does not suffer from this problem. If ϵ is made small enough, the "crawling" behavior of bodies is not visible, no matter what masses or forces or exist. If ϵ is made excessively small, the differential equations of motion may become stiff; otherwise, the method has a reasonable performance. The graphical simulations produced using this approximation are fairly realistic.

14.2.3 Quadratic programming approximation

The last method we present attempts to model static friction by quadratic programming. The basic idea is that by guessing the signs of the variables f_{xi} and f_{yi} in a solution, we can solve for $\boldsymbol{f}$ by solving a QP. Given a guess of the values

$$\operatorname{sgn}(f_{xi}) \qquad \text{and} \qquad \operatorname{sgn}(f_{yi}) \tag{14–13}$$

where $\operatorname{sgn}(x) = 1$ if $x \geq 0$ and -1 otherwise, we will try to find a solution $\boldsymbol{f}$ that agrees with these signs. Thus, we treat $\operatorname{sgn}(f_{xi})$ and $\operatorname{sgn}(f_{yi})$ as known and fixed quantities for now. We will describe later how we produce a guess of these values, and then discuss some shortcomings of the method.

Because we cannot represent the constraint $f_{xi}^2 + f_{yi}^2 \leq (\mu f_{Ni})^2$ in a QP, we decouple the $\hat{t}_x$ and $\hat{t}_y$ axes as Lötstedt does, replacing $f_{xi}^2 + f_{yi}^2 \leq (\mu f_{Ni})^2$ with

$$-\mu f_{Ni} \leq f_{xi} \leq \mu f_{Ni} \qquad \text{and} \qquad -\mu f_{Ni} \leq f_{yi} \leq \mu f_{Ni}. \tag{14–14}$$

Note however that for two-dimensional configurations, no decoupling of axes is needed and this method does not making *any* approximations of the laws of static friction.

We will also replace the condition $a_{xi} f_{xi} + a_{yi} f_{yi} \leq 0$ with the conditions

$$\begin{aligned} f_{xi} \operatorname{sgn}(f_{xi}) \geq 0 \qquad &\text{and} \qquad a_{xi} \operatorname{sgn}(f_{xi}) \leq 0 \\ f_{yi} \operatorname{sgn}(f_{yi}) \geq 0 \qquad &\text{and} \qquad a_{yi} \operatorname{sgn}(f_{yi}) \leq 0. \end{aligned} \tag{14–15}$$

This condition ensures that $a_{xi} f_{xi} + a_{yi} f_{yi} \leq 0$ (and is actually somewhat stronger). Note that the (seeming) tautologies $f_{xi} \operatorname{sgn}(f_{xi}) \geq 0$ and $f_{yi} \operatorname{sgn}(f_{yi}) \geq 0$ are actually constraints that f_{xi} and f_{yi} have the same signs as $\operatorname{sgn}(f_{xi})$ and $\operatorname{sgn}(f_{yi})$ (which we are regarding as known quantities). Next, we add the standard constraint that

$$f_{Ni} \geq 0 \qquad \text{and} \qquad a_{Ni} \geq 0. \tag{14–16}$$

Since we are treating $\mathrm{sgn}(f_{xi})$ and $\mathrm{sgn}(f_{y_i})$ as known quantities, and a_{Ni}, a_{xi} and a_{y_i} are linear functions of the contact forces, all of the above conditions can be expressed as linear constraints involving f_{xi}, f_{y_i} and f_{Ni}.

The condition that the static friction force attains its upper bound when slipping begins is written as

$$\begin{aligned}(\mu f_{Ni} - f_{xi}\,\mathrm{sgn}(f_{xi}))(-a_{xi}\,\mathrm{sgn}(f_{xi})) = 0 \\ (\mu f_{Ni} - f_{y_i}\,\mathrm{sgn}(f_{y_i}))(-a_{y_i}\,\mathrm{sgn}(f_{y_i})) = 0.\end{aligned} \tag{14–17}$$

Finally, we add the standard constraint on the normal forces that

$$(f_{Ni})(a_{Ni}) = 0. \tag{14–18}$$

Since each of f_{Ni}, a_{Ni}, $\mu f_{Ni} - f_{xi}\,\mathrm{sgn}(f_{xi})$, $\mu f_{Ni} - f_{y_i}\,\mathrm{sgn}(f_{y_i})$, $-a_{xi}\,\mathrm{sgn}(f_{xi})$, and $-a_{y_i}\,\mathrm{sgn}(f_{y_i})$ are constrained to be positive, we can express the fact that Equations (14–17) and (14–18) hold for all i as

$$\sum_{i=1}^{n} \begin{pmatrix} f_{Ni}a_{Ni} + (\mu f_{Ni} - f_{xi}\,\mathrm{sgn}(f_{xi}))(-a_{xi}\,\mathrm{sgn}(f_{xi})) \\ +(\mu f_{Ni} - f_{y_i}\,\mathrm{sgn}(f_{y_i}))(-a_{xi}\,\mathrm{sgn}(f_{y_i})) \end{pmatrix} = 0. \tag{14–19}$$

Since this is a quadratic function of f_{xi}, f_{y_i} and f_{Ni}, we can find a solution to Equation (14–19) satisfying the linear constraints of Equations (14–14), (14–15) and (14–16) by solving a QP.

We still must describe how to guess the signs of f_{xi} and f_{y_i}. Iterative methods for quadratic programming exist that are very similar to the Gauss-Seidel or Jacobi iterative methods used to solve linear systems[40]. These iterative routines are much simpler than routines that solve QP's by direct methods; in particular, it is extremely simple to modify these iterative routines to find a solution to Equation (14–1). The reason that we do not use these modified iterative routines is that they have poor convergence properties. However, we can run one of these modified iterative routines for some period of time and examine the (hopefully) partially converged variables

f_{x_i} and f_{y_i} to guess their signs. (Appendix C discusses how to formulate an iterative routine to solve Equation (14–1).) Using the signs of the variables, we then form and solve a quadratic program as described above. Coherence can be exploited (and must, to provide continuity over time steps) by using the same concept of basic solutions presented in sections 8.2 and 13.3.

This method can break down at any number of places. First, we still do not know that a solution for static friction always exists. Even if there is a solution, it is possible that the iterative algorithm may not converge to a solution, so our guess of the signs of f_{x_i} and f_{y_i} may not form a QP that leads to a solution. Furthermore, the QP may be non-convex, and therefore impractical to solve (even if only static friction occurs). Last, if we attempt to incorporate both dynamic and static friction, we still need to be concerned about the *NP*-hardness due to just the dynamic friction. Unfortunately, the form of the linear constraints of the QP do not allow us to solve it as an LCP using Lemke's method. No quadratic programming algorithm is known that has the property of Lemke's algorithm: returning a valid non-impulsive solution or indicating a valid impulsive solution. It is possible that whenever the iterative algorithm fails to converge, it does so in such a way that an analysis of the divergence indicates a valid set of contact impulses; it is not clear how to perform such an analysis.

However, this last method yields a very acceptable visual result, when it works. For large numbers of contact points ($n \approx 40$), the last method sometimes breaks down; either the iteration does not converge, or it converges to an incorrect solution. For configurations with approximately 40 contact points or less, the method is reasonably successful. Using this method, simulations with $n \approx 40$, and involving thousands of time steps have been run successfully. Images from several such simulations are shown in Appendix D.

Chapter 15

Future Directions

In this thesis, we have used a model of rigid body dynamics that is greatly simplified to examine the difficulties of simulation of complex physical systems. Even using a simplified model of rigid body dynamics, this thesis has shown that the problem of dynamic simulation with non-interpenetration constraints is a difficult problem. Beginning with the dynamics of frictionless configurations with a single point of contact, we have seen that producing a computationally tractable solution method for the contact force at the single contact point is non-trivial. Next, we considered properties and solution methods for configurations with multiple contact points and friction. The most difficult problems we encountered have centered on the behavior of configurations with friction. We have shown that the computational complexity of computing a non-impulsive solution for configurations without colliding contacts is *NP*-hard; more importantly, however, we have shown that it is not necessarily important to specifically compute such a solution, and that an absolute preference for such solutions over the course of a simulation violates causality and is thus unnatural. A model that allows expected polynomial time solution methods has been proposed, and iterative computational methods for computing contact forces

for configurations with and without collisions have been proposed.

A large number of graphical simulations have been produced using the methods described in this thesis, and images from several simulations are shown in Appendix D. Although these simulations yield results that seem visually correct, they are still simulations of a system subject to a number of simplifying assumptions, or restrictions. How severe have these restrictions been, and what is required to relax them?

The major restriction in the work described in this thesis has been the assumption that contact between bodies occurs at a finite number of contact points. For simulations with frictionless polyhedra, this restriction introduces no approximations into the simulation, since frictionless systems are determinate. However, this restriction prohibits simulations involving contact areas with curved boundaries, such as a cylinder standing upright on a plane. The problem with such a simulation is how to formulate the non-penetration constraint: since the cylinder could potentially be tipped over from any direction, every point on the circular boundary of the contact region (between the plane and the cylinder) must be included in a non-penetration constraint. Currently, it is unclear how one should write or attempt to solve the constraint equations for this case.

Interestingly, the same problem arises when considering the formulation of non-penetration constraints between flexible bodies that contact with a one- or two-dimensional contact region. Even if the contact region is polyhedral, inter-penetration is not necessarily prevented by constraints on the allowed motion of the vertices. A deformation that causes a flat surface to become curved could result in inter-penetration at an interior point of a contact region even though no inter-penetration had occurred on the boundary of that contact region[2]. Additionally, dealing with one- or two-dimensional contact regions also requires research into contact determination algorithms that can both detect and describe, in some fashion,

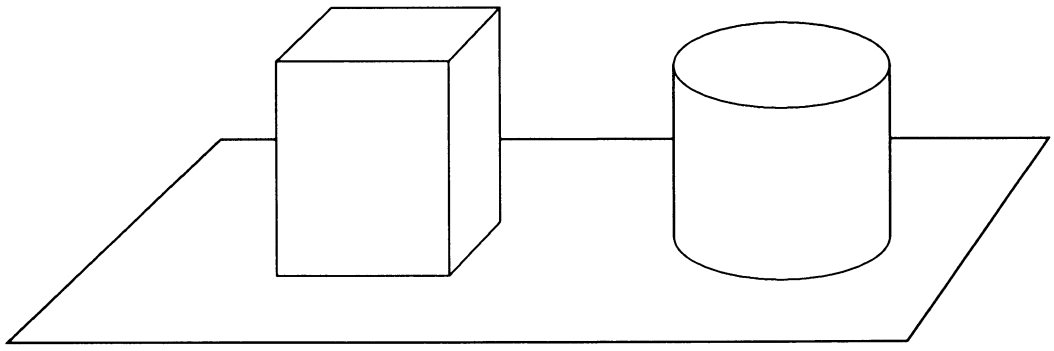


Figure 15.1: Non-polygonal contact region. The contact area between the cube and the plane is a polygon, while the contact area between the cylinder and the plane is a circle. The non-penetration constraint for the cube is posed in terms of the finitely many vertices describing the contact area boundary. For the cylinder, infinitely many points are necessary to describe the contact area boundary.

one- and two-dimensional contact regions. Finding collision/contact determination algorithms that can also deal with non-convex curved surfaces is also an interesting research problem.

For simulations with friction, the restriction to finitely many contact points is more severe. In essence, the tangential friction forces that should occur over a line or area of contact are restricted to act only at the vertices of the contact region. Again, the problem is how to formulate the Coulomb friction laws over an entire region of contact; this is a very open research problem.

Another problem is computing contact forces and impulses in the presence of friction. The assumption of constant slip velocity direction for three-dimensional collisions simplifies computations, but is unrealistic and must be dealt with. Also, we do not have a guaranteed solution method for configurations involving both static and dynamic friction. There is also the philosophical issue of what a simulation should do when faced with indeterminacy: this problem has been largely (if not completely) ignored by this thesis. There has been some speculation that both indeterminacy and inconsistency due to friction could be eliminated by discarding the restriction that bodies are perfectly rigid. Depending on the model of non-rigidity adopted, inconsistency and indeterminacy can either be partly or completely eliminated. For example, if a standard mass point and spring system is used to model deformable bodies, both inconsistency and indeterminacy are completely eliminated. In such a model, a given contact force affects the instantaneous acceleration of only a single mass point; the change in position of the mass point indirectly affects neighboring mass points via the spring forces. Since a mass point's instantaneous normal acceleration is unaffected by any tangential friction forces, neither indeterminacy nor inconsistency is possible. Note that this model of flexibility eliminates inconsistency and indeterminacy by giving the body an essentially unlimited number of degrees of freedom ($3n$ degrees of freedom for a body with n mass points). There also exists

a middle ground, in which bodies are allowed to deform, but with only a limited number of degrees of freedom. Witkin[56] and Baraff and Witkin[2] discuss flexible bodies that are limited to linear or quadratic deformations of a rest shape. Under this model, inconsistency and indeterminacy are only partly eliminated; the limited flexibility of the body eliminates inconsistency and indeterminacy in configurations with one contact point, but allows inconsistency and indeterminacy in configurations with two or more contact points. In general, one could speculate that the more degrees of freedom a model permits, the less likely indeterminacy and inconsistency can occur.

Appendix A

Extremal Point Conditions for Parametric Surfaces

The derivations in this thesis have all assumed implicit descriptions of curved surfaces. In this appendix, derivations for surfaces described parametrically are given. We will begin by defining the parametric analogue of Equation (5–2).

Let the surfaces A and B be modeled by the time-varying parametric functions $S(u,v,t)$ and $T(u,v,t)$. Componentwise,

$$S(u,v,t) = \begin{pmatrix} S_x(u,v,t) \\ S_y(u,v,t) \\ S_z(u,v,t) \end{pmatrix} \quad \text{and} \quad T(u,v,t) = \begin{pmatrix} T_x(u,v,t) \\ T_y(u,v,t) \\ T_z(u,v,t) \end{pmatrix}. \tag{A–1}$$

A normal vector $S_N(u,v,t)$ is given by

$$S_N(u,v,t) = \frac{\partial S}{\partial u}(u,v,t) \times \frac{\partial S}{\partial v}(u,v,t) \tag{A–2}$$

and similarly for T. If the extremal points p_a and p_b at time t are defined in terms of parametric coordinates (u_a, v_a) and (u_b, v_b) by

$$p_a = S(u_a, v_a, t) \quad \text{and} \quad p_b = T(u_b, v_b, t) \tag{A–3}$$

then the analogue of Equation (5–2) is

$$\begin{cases} E_1: & S_N(u_a, v_a, t) + \lambda_2 T_N(u_b, v_b, t) = \mathbf{0} \\ E_2: & (T(u_b, v_b, t) - S(u_a, v_a, t)) + \lambda_1 T_N(u_b, v_b, t) = \mathbf{0}. \end{cases} \tag{A–4}$$

The analogue of Equation (5–4) is

$$\begin{cases} S_N(u_a, v_a, t)^T r = 0 \\ (S(u_a, v_a, t) - (P_0 + c\, r)) + \lambda_1 S_N(u_a, v_a, t) = \mathbf{0}, \end{cases} \tag{A–5}$$

while the analogue of Equation (5–5) is

$$\begin{cases} (S(u_a, v_a, t) - P_0) + \lambda_1 S_N(u_a, v_a, t) = \mathbf{0}. \end{cases} \tag{A–6}$$

These equations can be solved to find the extremal points the during collision/ contact determination step.

In deriving an expression for $\ddot{C}$ for force determination, the functions S and T are treated as configuration dependent functions $S(u, v, \overline{p})$ and $T(u, v, \overline{p})$. Equation (A–4) then becomes

$$\begin{cases} S_N(u_a, v_a, \overline{x}(t)) + \lambda_2 T_N(u_b, v_b, \overline{x}(t)) = \mathbf{0} \\ (T(u_b, v_b, \overline{x}(t)) - S(u_a, v_a, \overline{x}(t))) + \lambda_1 T_N(u_b, v_b, \overline{x}(t)) = \mathbf{0}. \end{cases} \tag{A–7}$$

For force determination, it was also necessary to define functions $f(x, y, \overline{p})$ and $g(x, y, \overline{p})$ that explicitly describe the surfaces of A and B in a local neighborhood of the extremal points. For parametric functions, the analogue of Equation (6–14) is

$$f(S_x(u, v, \overline{x}(t)), S_y(u, v, \overline{x}(t)), \overline{x}(t)) = S_z(u, v, \overline{x}(t)) \tag{A–8}$$

and similarly for g and T. As in Section 6.4, the existence of f and g follows from the implicit function theorem. Partial derivatives of f and g can be expressed in terms of partial derivatives of S and T. If $\overline{p}$ has k coordinates, then differentiating

Equation (A–8) with respect to u, v, and $\overline{p}_1$ through $\overline{p}_k$ yields

$$\begin{array}{ccccccc}
\frac{\partial f}{\partial x}\frac{\partial S_x}{\partial u} & + & \frac{\partial f}{\partial y}\frac{\partial S_y}{\partial u} & & & = & \frac{\partial S_z}{\partial u} \\
\frac{\partial f}{\partial v}\frac{\partial S_x}{\partial v} & + & \frac{\partial f}{\partial y}\frac{\partial S_y}{\partial v} & & & = & \frac{\partial S_z}{\partial v} \\
\frac{\partial f}{\partial x}\frac{\partial S_x}{\partial \overline{p}_1} & + & \frac{\partial f}{\partial y}\frac{\partial S_y}{\partial \overline{p}_1} & + & \frac{\partial f}{\partial \overline{p}_1} & = & \frac{\partial S_z}{\partial \overline{p}_1} \\
\vdots & & \vdots & & \ddots & = & \vdots \\
\frac{\partial f}{\partial x}\frac{\partial S_x}{\partial \overline{p}_k} & + & \frac{\partial f}{\partial y}\frac{\partial S_y}{\partial \overline{p}_k} & & + \frac{\partial f}{\partial \overline{p}_k} & = & \frac{\partial S_z}{\partial \overline{p}_k}.
\end{array} \tag{A–9}$$

Thus, the first partials of f may be expressed in terms of the first partials of S by solving a simpler linear system (note that Equation (A–9) is mostly diagonal). Second partials of f are obtained by differentiating Equation (A–9) and solving another linear system. These derivatives are needed for Equation (6–29), along with the derivatives with respect to time of the extremal points: applying the implicit function theorem to Equation (A–4) yields

$$\dot{u}_a = -\frac{\frac{\partial(E_1, E_2)}{\partial(t, v_a, u_b, v_b)}}{J} \quad \text{and} \quad \dot{v}_a = -\frac{\frac{\partial(E_1, E_2)}{\partial(u_a, t, u_b, v_b)}}{J} \tag{A–10}$$

(and similarly for $\dot{u}_b$ and $\dot{v}_b$) where

$$J = \frac{\partial(E_1, E_2)}{\partial(u_a, v_a, u_b, v_b)}. \tag{A–11}$$

Given the derivatives of the parametric coordinates, we can calculate $\dot{p}_a$ and $\dot{p}_b$. From Equation (A–1), and since $p_a = S(u_a, v_a, \overline{x}(t))$

$$\dot{p}_{ax} = \frac{\partial S_x}{\partial u}(u_a, v_a, \overline{x}(t))\dot{u}_a + \frac{\partial S_x}{\partial v}(u_a, v_a, \overline{x}(t))\dot{v}_a + \frac{\partial S_x}{\partial \overline{p}}(u_a, v_a, \overline{x}(t))\dot{\overline{x}}(t) \tag{A–12}$$

and similarly for $\dot{p}_{ay}$, $\dot{p}_{bx}$ and $\dot{p}_{by}$.

Appendix B

Termination Conditions of Lemke's Algorithm

In this appendix, we will prove that when Lemke's algorithm fails to find a solution to Equation (8–7), it finds a vector $\mathbf{z}$ satisfying Equation (13–1); furthermore, for some $1 \leq i \leq n$ (where n is the size of the LCP of Equation (8–7)), the value z_i is positive, and the ith contact point of the system has dynamic friction. The proof is based on a proof by Murty[40, section 2.3, pages 86–88] that Lemke's algorithm, when applied to a copositive plus LCP,[1] either finds a solution, or terminates with an unbounded ray.

THEOREM 5. *When Lemke's algorithm fails to find a solution to the LCP of Equation* (8–7)*, it generates a vector* $\mathbf{z} \geq \mathbf{0}$ *with the following two properties.*

(1) *For all* $1 \leq i \leq n$*, if* $z_i > 0$ *then* $(\mathbf{Az})_i \leq 0$.

(2) *For at least one contact point of the system with dynamic friction, say the* i*th contact point,* $z_i > 0$.

[1] An LCP is copositive plus if for all $\mathbf{x} \geq \mathbf{0}$, the matrix $\mathbf{A}$ in the LCP satisfies $\mathbf{x}^T\mathbf{Ax} \geq 0$, and $(\mathbf{A}^T + \mathbf{A})\mathbf{x} = \mathbf{0}$ whenever $\mathbf{x}^T\mathbf{Ax} = 0$.

PROOF. Since the ordering of contact points is arbitrary, let the contact points of the system be ordered such that contact points 1 to k are all frictionless, and contact points $k+1$ to n have dynamic friction. Let the matrix $\mathbf{A}$ and constant vector $\mathbf{b}$ corresponding to this reordered system be partitioned by

$$\mathbf{A} = \begin{pmatrix} \mathbf{M} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} \mathbf{b_1} \\ \mathbf{b_2} \end{pmatrix} \tag{B–1}$$

where $\mathbf{M}$ is a $k \times k$ matrix, $\mathbf{B}$ is a $k \times (n-k)$ matrix, $\mathbf{C}$ is an $(n-k) \times k$ matrix, $\mathbf{D}$ is an $(n-k) \times (n-k)$ matrix, $\mathbf{b_1}$ is a k-vector, and $\mathbf{b_2}$ is an $(n-k)$-vector. Since the first k points are frictionless, the matrix $\mathbf{M}$ is PSD, and the vector $\mathbf{b_1}$ is in the range of $\mathbf{M}$. Thus, as established in Chapter 8, the LCP

$$\mathbf{a_N} = \mathbf{M}\boldsymbol{f_N} - \mathbf{b_1} \geq \mathbf{0}, \quad \boldsymbol{f_N} \geq \mathbf{0}, \qquad \text{and} \qquad \boldsymbol{f_N}^T\mathbf{a_N} = 0 \tag{B–2}$$

has a solution.

The proof that a vector satisfying property (1) exists is essentially a restatement of the termination conditions of Lemke's algorithm as detailed by Murty. For convenience, in this appendix the vector $\mathbf{e}$ is taken to be an appropriately sized vector of ones; that is, $e_i = 1$ for all i. When Lemke's algorithm fails to find a solution to the LCP of Equation (B–1), it finds vectors $\mathbf{z}^h$, $\mathbf{z}^k$, $\mathbf{w}^h$, $\mathbf{w}^k$ and scalars z_0^h and z_0^k satisfying

$$\begin{gathered} \mathbf{w}^k = \mathbf{A}\mathbf{z}^k - \mathbf{b} + z_0^k\mathbf{e} \\ \mathbf{w}^h = \mathbf{A}\mathbf{z}^h + z_0^h\mathbf{e} \\ \mathbf{w}^k \geq \mathbf{0}, \quad \mathbf{w}^h \geq \mathbf{0}, \quad \mathbf{z}^k \geq \mathbf{0}, \quad \mathbf{z}^h \geq \mathbf{0} \text{ (but } \mathbf{z}^h \neq \mathbf{0}), \\ z_0^h \geq 0, \quad z_0^k > 0 \end{gathered} \tag{B–3}$$

and

$$(\mathbf{w}^k + \lambda\mathbf{w}^h)^T(\mathbf{z}^k + \lambda\mathbf{z}^h) = 0 \qquad \text{for all } \lambda \geq 0. \tag{B–4}$$

From Equations (B–3) and (B–4), it follows that

$$(\mathbf{w}^h)^T\mathbf{z}^h = 0, \quad (\mathbf{w}^k)^T\mathbf{z}^k = 0, \quad (\mathbf{w}^k)^T\mathbf{z}^h = 0 \qquad \text{and} \qquad (\mathbf{w}^h)^T\mathbf{z}^k = 0. \tag{B–5}$$

From this, it follows that if $\mathbf{z}_i^h > 0$ then $\mathbf{w}_i^h = 0$. But if $\mathbf{z}_i^h > 0$, then

$$0 = \mathbf{w}_i^h = (\mathbf{A}\mathbf{z}^h)_i + (z_0^h\mathbf{e})_i \tag{B–6}$$

so

$$(\mathbf{A}\mathbf{z}^h)_i = -(z_0^h\mathbf{e})_i = -z_0^h e_i = -z_0^h \leq 0. \tag{B–7}$$

Thus, the vector $\mathbf{z}^h$ satisfies property (1) of the proof.

In order for $\mathbf{z}^h$ to satisfy property (2), at least one non-zero element of $\mathbf{z}^h$ must correspond to a contact point with dynamic friction. Let $\mathbf{z}^h$, $\mathbf{w}^k$ and $\mathbf{w}^h$ be partioned as follows:

$$\mathbf{w}^h = \begin{pmatrix} \mathbf{v}^h \\ \mathbf{u}^h \end{pmatrix}, \quad \mathbf{w}^k = \begin{pmatrix} \mathbf{v}^k \\ \mathbf{u}^k \end{pmatrix}, \qquad \text{and} \qquad \mathbf{z}^h = \begin{pmatrix} \mathbf{x}^h \\ \mathbf{y}^h \end{pmatrix}. \tag{B–8}$$

The vectors $\mathbf{v}^h$, $\mathbf{v}^k$ and $\mathbf{x}^h$ have dimension k, while $\mathbf{u}^h$, $\mathbf{u}^k$ and $\mathbf{y}^h$ have dimension $n - k$. To show that property (2) is true for $\mathbf{z}^h$, we use proof by contradiction. Suppose that $\mathbf{y}^h = \mathbf{0}$; that is, suppose that all elements of $\mathbf{z}^h$ that correspond to a contact point with dynamic friction are zero. Then $\mathbf{z}^h$ would have the form

$$\mathbf{z}^h = \begin{pmatrix} \mathbf{x}^h \\ \mathbf{y}^h \end{pmatrix} = \begin{pmatrix} \mathbf{x}^h \\ \mathbf{0} \end{pmatrix}. \tag{B–9}$$

Now suppose that $z_0^h > 0$. Then since $\mathbf{z}^h$ is non-zero, for some $1 \leq j \leq k$, we have $z_j^h > 0$. As before, if $z_j^h > 0$ then $w_j^h = 0$. But then

$$0 = w_j^h = (\mathbf{A}\mathbf{z}^h)_j + (z_0^h\mathbf{e})_j = (\mathbf{A}\mathbf{z}^h)_j + z_0^h \tag{B–10}$$

so

$$(\mathbf{A}\mathbf{z}^h)_j = -z_0^h < 0. \tag{B–11}$$

Since $\mathbf{z}^h$ satisfies property (1),

$$(\mathbf{z}^h)^T\mathbf{A}\mathbf{z}^h = \sum_{i=1}^{n} z_i^h(\mathbf{A}\mathbf{z}^h)_i \leq 0; \tag{B–12}$$

but $(\mathbf{A}\mathbf{z}^h)_j = -z_0^h < 0$ and $z_j^h > 0$ for some $1 \leq j \leq k$ implies that $(\mathbf{z}^h)^T\mathbf{A}\mathbf{z}^h < 0$. Now, since $(\mathbf{z}^h)^T = (\mathbf{x}^h, \mathbf{0})^T$,

$$0 > (\mathbf{z}^h)^T\mathbf{A}\mathbf{z}^h = ((\mathbf{x}^h)^T, \mathbf{0}^T)\begin{pmatrix} \mathbf{M} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}\begin{pmatrix} \mathbf{x}^h \\ \mathbf{0} \end{pmatrix} = (\mathbf{x}^h)^T\mathbf{M}\mathbf{x}^h. \tag{B–13}$$

But this contradicts the fact that $\mathbf{M}$ is PSD; thus $z_0^h \neq 0$ leads to a contradiction.

Conversely, suppose that $z_0^h = 0$. Since $(\mathbf{x}^h)^T\mathbf{M}\mathbf{x}^h = (\mathbf{z}^h)^T\mathbf{A}\mathbf{z}^h \leq 0$, and $\mathbf{M}$ is PSD, it must be that $(\mathbf{x}^h)^T\mathbf{M}\mathbf{x}^h = 0$. However, the fact that $\mathbf{M}$ is PSD means that $\mathbf{M}$ is also copositive plus; this in turn implies that $(\mathbf{M}^T + \mathbf{M})\mathbf{x}^h = \mathbf{0}$. Since $\mathbf{M}$ is symmetric as well, it follows that $\mathbf{M}\mathbf{x}^h = \mathbf{0}$.

Since

$$\mathbf{w}^k = \mathbf{A}\mathbf{z}^k - \mathbf{b} + z_0^k\mathbf{e},$$

from the partitioning of $\mathbf{A}$, $\mathbf{w}^k$, and $\mathbf{z}^k$,

$$\mathbf{u}^k = \mathbf{M}\mathbf{x}^k - \mathbf{b_1} + z_0^k\mathbf{e}. \tag{B–14}$$

Then since $(\mathbf{x}^h)^T\mathbf{M} = (\mathbf{M}\mathbf{x}^h)^T = \mathbf{0}^T$,

$$\begin{aligned} (\mathbf{x}^h)^T\mathbf{u}^k &= (\mathbf{x}^h)^T(\mathbf{M}\mathbf{x}^k - \mathbf{b_1} + z_0^k\mathbf{e}) \\ &= (\mathbf{x}^h)^T\mathbf{M}\mathbf{x}^k - (\mathbf{x}^h)^T\mathbf{b_1} + (\mathbf{x}^h)^T z_0^k\mathbf{e} \\ &= -(\mathbf{x}^h)^T\mathbf{b_1} + (\mathbf{x}^h)^T z_0^k\mathbf{e}. \end{aligned}$$

Since $\mathbf{w}^k$ and $\mathbf{z}^h$ satisfy $(\mathbf{w}_i^k)(\mathbf{z}_i^h) = 0$, it follows that $(\mathbf{u}^k)^T\mathbf{x}^h = 0$, so

$$(\mathbf{x}^h)^T\mathbf{u}^k = 0 = -(\mathbf{x}^h)^T\mathbf{b_1} + (\mathbf{x}^h)^T z_0^k\mathbf{e} \tag{B–15}$$

or equivalently

$$-(\mathbf{x}^h)^T\mathbf{b_1} = -(\mathbf{x}^h)^T z_0^k \mathbf{e}. \tag{B–16}$$

Since $\mathbf{x}^h \geq 0$, $\mathbf{e} > 0$ and $z_0^k > 0$, and $x_i^h > 0$ for some i,

$$-(\mathbf{x}^h)^T\mathbf{b_1} < 0. \tag{B–17}$$

Letting I denote the $k \times k$ identity matrix, and $(I \mid -M)$ the $k \times 2k$ matrix formed by catenating I and $-M$, we have

$$(\mathbf{x}^h)^T(I \mid -M) = (\mathbf{x}^h)^T - (\mathbf{x}^h)^T\mathbf{M} = (\mathbf{x}^h)^T > \mathbf{0} \quad \text{and}$$
$$-(\mathbf{x}^h)^T\mathbf{b_1} < 0. \tag{B–18}$$

By Farkas' lemma[40, appendix 1] this implies that the system

$$(I \mid -M)\begin{pmatrix} \mathbf{a_N} \\ \boldsymbol{f}_\mathbf{N} \end{pmatrix} = -\mathbf{b_1} \qquad \text{and} \qquad \begin{pmatrix} \mathbf{a_N} \\ \boldsymbol{f}_\mathbf{N} \end{pmatrix} \geq \mathbf{0} \tag{B–19}$$

is *infeasible*; that is, no $\boldsymbol{f}_\mathbf{N}$ and $\mathbf{a_N}$ satisfying Equation (B–19) exists. This means that

$$(I \mid -M)\begin{pmatrix} \mathbf{a_N} \\ \boldsymbol{f}_\mathbf{N} \end{pmatrix} = \mathbf{a_N} - \mathbf{M}\boldsymbol{f}_\mathbf{N} = -\mathbf{b_1} \qquad \text{and} \qquad \begin{pmatrix} \mathbf{a_N} \\ \boldsymbol{f}_\mathbf{N} \end{pmatrix} \geq \mathbf{0} \tag{B–20}$$

or equivalently

$$\mathbf{a_N} = \mathbf{M}\boldsymbol{f}_\mathbf{N} - \mathbf{b_1} \qquad \text{and} \qquad \begin{pmatrix} \mathbf{a_N} \\ \boldsymbol{f}_\mathbf{N} \end{pmatrix} \geq \mathbf{0} \tag{B–21}$$

is infeasible. However, this contradicts the fact that Equation (B–2) always has a solution, since it is the LCP of a frictionless configuration. Thus, $z_0^h = 0$ leads to a contradiction.

Since both $z_0^h = 0$ and $z_0^h \neq 0$ lead to contradictions, the original assumption that $\mathbf{y}^h = 0$ must in fact be false. Thus, at least one element z_i^h of $\mathbf{z}^h$ corresponding to a contact point with dynamic friction satisfies $z_i^h > 0$, and $\mathbf{z}^h$ satisfies both properties (1) and (2). □

Appendix C

Iterative Solution Methods for Static Friction

In this appendix, a Jacobi-like iterative method for solving Equation (14–1) is derived. As stated in Section 14.2.3, the iterative method is applied long enough to estimate the signs of the unknown variables. Gauss-Seidel, Jacobi, and successive overrelaxation (SOR) iterations have been developed for solving linear complementarity problems[9,33]. The Jacobi-like iteration for solving Equation (14–1) developed in this appendix can easily be made into either a Gauss-Seidel- or SOR-like iterative method.

Linear equations

Consider the linear equation

$$\mathbf{A}\mathbf{z} + \mathbf{q} = \mathbf{0} \tag{C–1}$$

where $\mathbf{A}$ is an $n \times n$ matrix, $\mathbf{q}$ is an n-vector, and $\mathbf{z}$ is the unknown n-vector being solved for. Iterative methods generate a sequence of solutions $\{\mathbf{z}^{(0)}, \mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots\}$ that (hopefully) converge to a vector $\mathbf{z}$ satisfying Equation (C–1).

The Jacobi iteration for computing $\mathbf{z}^{(k+1)}$ is the following.

$\mathbf{e}^{(k)} = \mathbf{A}\mathbf{z}^{(k)} + \mathbf{q}$
for $i = 1$ **to** n
    $z_i^{(k+1)} = z_i^{(k)} - e_i^{(k)}/A_{ii}$
done

Note that if $\mathbf{z}^{(k)}$ exactly solves Equation (C–1), then $\mathbf{e}^{(k+1)} = \mathbf{A}\mathbf{z} + \mathbf{q} = \mathbf{0}$, and $\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)}$. In practice, the iteration is stopped when $\mathbf{e}^{(k)}$ becomes suitably small. The Jacobi iteration is known to converge if $\mathbf{A}$ is positive definite (PD); in this case, the diagonal elements A_{ii} are all positive.

Linear complementarity

Now suppose that instead of solving Equation (C–1), we wish to find $\mathbf{z}$ such that

$$\mathbf{A}\mathbf{z} + \mathbf{q} \geq \mathbf{0}, \quad \mathbf{z} \geq \mathbf{0} \qquad \text{and} \qquad \mathbf{z}^T(\mathbf{M}\mathbf{z} + \mathbf{q}) = 0. \tag{C–2}$$

The Jacobi iteration for solving Equation (C–2) involves a *projection* of $\mathbf{z}^{(k)}$, to maintain the constraint $\mathbf{z} \geq \mathbf{0}$. The projection of a scalar x so that it is non-negative is written x_+ and is defined as

$$x_+ = \max(0, x). \tag{C–3}$$

The Jacobi iteration for solving Equation (C–2) is simply the Jacobi iteration for solving Equation (C–1), with the addition of a projection; that is,

$\mathbf{e}^{(k)} = \mathbf{A}\mathbf{z}^{(k)} + \mathbf{q}$
for $i = 1$ **to** n
    $z_i^{(k+1)} = (z_i^{(k)} - e_i^{(k)}/A_{ii})_+$
done

This is called the projected Jacobi iteration.

If $\mathbf{z}^{(k)}$ satisfies Equation (C–2), then $z_i^{(k)} > 0$ implies

$$e_i^{(k)} = (\mathbf{A}\mathbf{z}^{(k)} + \mathbf{q})_i = 0.$$

Thus $z_i^{(k+1)} = (z_i^{(k)})_+ = z^{(k)}$. Conversely, if $z_i^{(k)} = 0$, then

$$e_i^{(k)} = (\mathbf{A}\mathbf{z}^{(k)} + \mathbf{q})_i > 0, \tag{C–4}$$

so

$$z_i^{(k+1)} = (-e_i^{(k)}/A_{ii})_+ = 0$$

assuming $A_{ii} > 0$. (As a minimum, iterative methods generally assume the diagonal elements of $\mathbf{A}$ are positive.) Thus, if $\mathbf{z}^{(k)}$ satisfies Equation (C–2), then $\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)}$. If $\mathbf{A}$ is PD then the projected Jacobi iteration will always converge[33].

Static friction equations

We will use the notation of Equations (14–6), (14–7), and (14–8) in defining the vector of accelerations, $\mathbf{a}$, and the vector of forces, $\boldsymbol{f}$, along with the known matrix $\mathbf{A}$ and vector $\mathbf{b}$. (To be consistent, we will write $\mathbf{q} = -\mathbf{b}$, so that $\mathbf{a} = \mathbf{A}\boldsymbol{f} + \mathbf{q}$.) These equations define f_{xi}, f_{y_i}, a_{xi} and a_{y_i} for every contact point. If the configuration of bodies is such that the ith contact point does not have static friction, then the variables f_{xi}, f_{y_i}, a_{xi}, and a_{y_i} are left out of Equations (14–7) and (14–8); the corresponding steps for computing f_{xi} and f_{y_i} in the Jacobi-like iteration are ignored as well. For simplicity, if f_{Ni}, f_{xi} and f_{y_i} are the pth, qth and rth elements of $\boldsymbol{f}$, then the diagonal elements of $\mathbf{A}$ corresponding to these indices are denoted $D_{Ni} = A_{pp}$, and $D_{xi} = A_{qq}$ and $D_{y_i} = A_{rr}$. If the ith contact point has dynamic friction, it is possible that D_{Ni} may be zero or negative if the value of μ_i (the

coefficient of friction at the contact point) is non-zero. If this happens, then it is trivial to apply a valid contact impulse to the system by imposing an impulse at just the ith contact point, converting the dynamic friction to static friction. Otherwise, the ith contact has static friction and D_{Ni} is positive. Thus, without loss of generality, we may assume $D_{Ni} > 0$ in all cases. Additionally, both D_{xi} and D_{y_i} are positive[28]. The Jacobi-like iteration to solve Equation (14–1) is

$$\mathbf{a}^{(k)} = \mathbf{A}\boldsymbol{f}^{(k)} + \mathbf{q}$$

for $i = 1$ **to** n

$$f_{N_i}{}^{(k+1)} = (f_{N_i}{}^{(k)} - a_{N_i}{}^{(k+1)}/D_{Ni})_+$$

$$v_f = ((f_{x_i}{}^{(k)})^2 + (f_{y_i}{}^{(k)})^2)^{1/2}$$

$$p = f_{x_i}{}^{(k)} a_{x_i}{}^{(k)} + f_{x_i}{}^{(k)} a_{x_i}{}^{(k)}$$

if $v_f < \mu_i f_{N_i}{}^{(k)}$ **or** $p > 0$ **then**

$$f_{x_i}{}^{(k+1)} = f_{x_i}{}^{(k)} - a_{x_i}{}^{(k)}/D_{x_i}$$

$$f_{y_i}{}^{(k+1)} = f_{y_i}{}^{(k)} - a_{y_i}{}^{(k)}/D_{y_i}$$

end

$$v'_f = ((f_{x_i}{}^{(k+1)})^2 + (f_{y_i}{}^{(k+1)})^2)^{1/2}$$

if $v'_f > \mu_i f_{N_i}{}^{(k)}$ **then**

$$f_{x_i}{}^{(k+1)} = f_{x_i}{}^{(k+1)}(\mu_i f_{N_i}{}^{(k)}/v'_f)$$

$$f_{y_i}{}^{(k+1)} = f_{y_i}{}^{(k+1)}(\mu_i f_{N_i}{}^{(k)}/v'_f)$$

end

done

The assignment to $f_{N_i}{}^{(k+1)}$ is the same as in the linear complementarity case. After this, the iteration proceeds as follows. The value v_f is the magnitude of the friction force. The value p is the dot product of the friction force and the tangential acceleration. If the magnitude of the friction force, v_f, has not exceeded its upper bound of $\mu_i f_{N_i}{}^{(k)}$, or if the friction and the acceleration are not opposite pointing

$(p > 0)$, then the friction forces are modified. The assignments

$$\begin{aligned} {f_{x_i}}^{(k+1)} &= {f_{x_i}}^{(k)} - {a_{x_i}}^{(k)}/D_{x_i} \\ {f_{y_i}}^{(k+1)} &= {f_{y_i}}^{(k)} - {a_{y_i}}^{(k)}/D_{y_i} \end{aligned}$$

attempt to modify the friction forces so that the tangential accelerations will be zero. Following this, the magnitude v'_f of the (possibly) modified friction forces is computed. If the friction force magnitude exceeds its upper bound of $\mu_i {f_{N_i}}^{(k)}$, the friction forces are suitably scaled (by $\mu_i {f_{N_i}}^{(k)}/v'_f$) so that their magnitude exactly attains this upper bound. As in the previous two Jacobi algorithms, when $\boldsymbol{f}^{(k)}$ satisfies Equation (14–1), then $\boldsymbol{f}^{(k+1)} = \boldsymbol{f}^{(k)}$. As stated, the purpose of this algorithm is to examine the signs of the variables after some number of iterations, as a prediction to solve the QP of Section 14.2.3.

Appendix D

Simulation Examples

This appendix contains images from simulations produced using the methods described in this thesis. Figure D.1 shows a simulation of a jack rolling down stairs. Figure D.2 shows a simulation of superquadric dice falling through a lattice. Figure D.3 shows a simulation of a collapsing structure with friction.

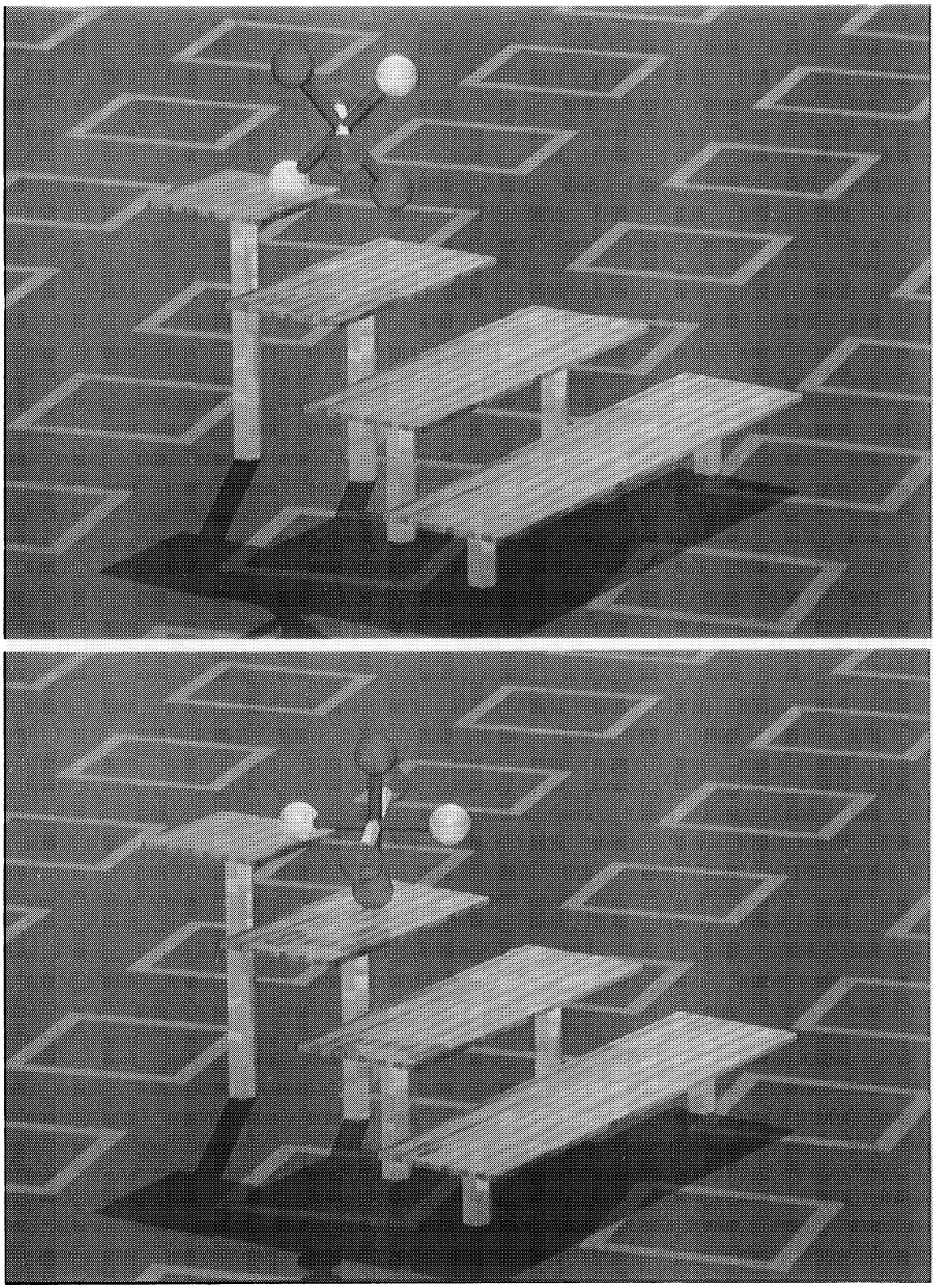
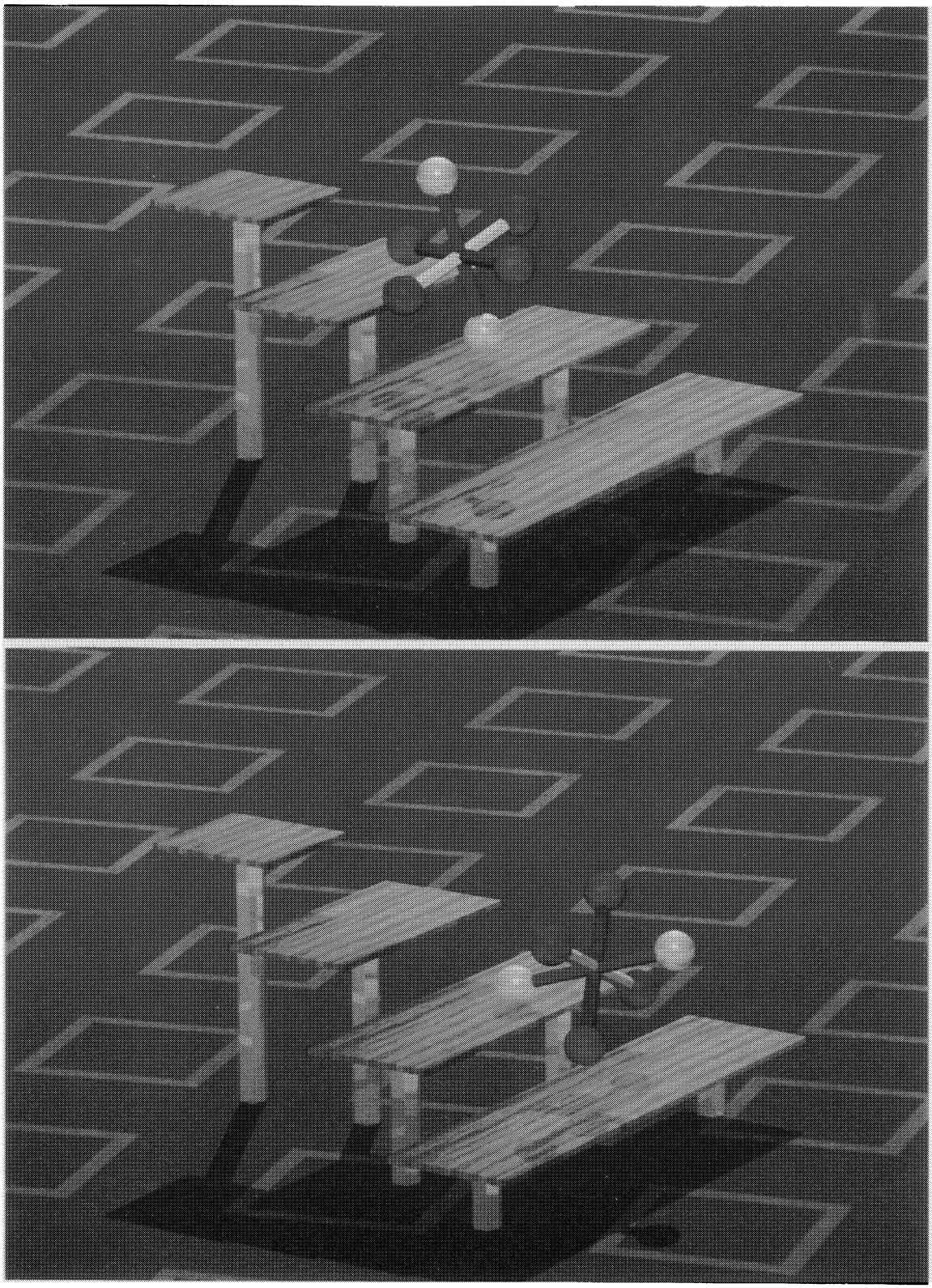


Figure D.1: Jack falling down stairs.

Figure D.1 (Continued)



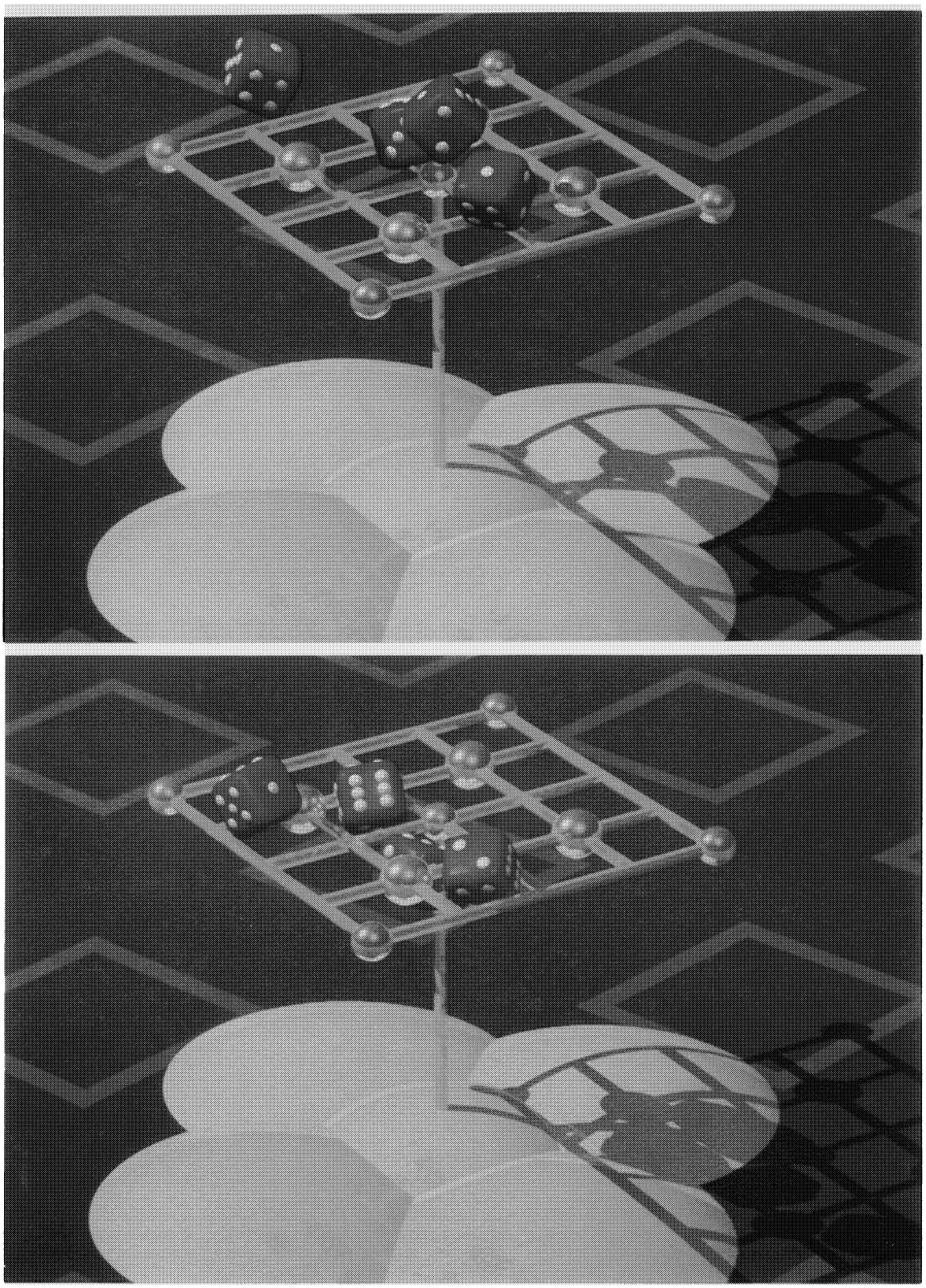
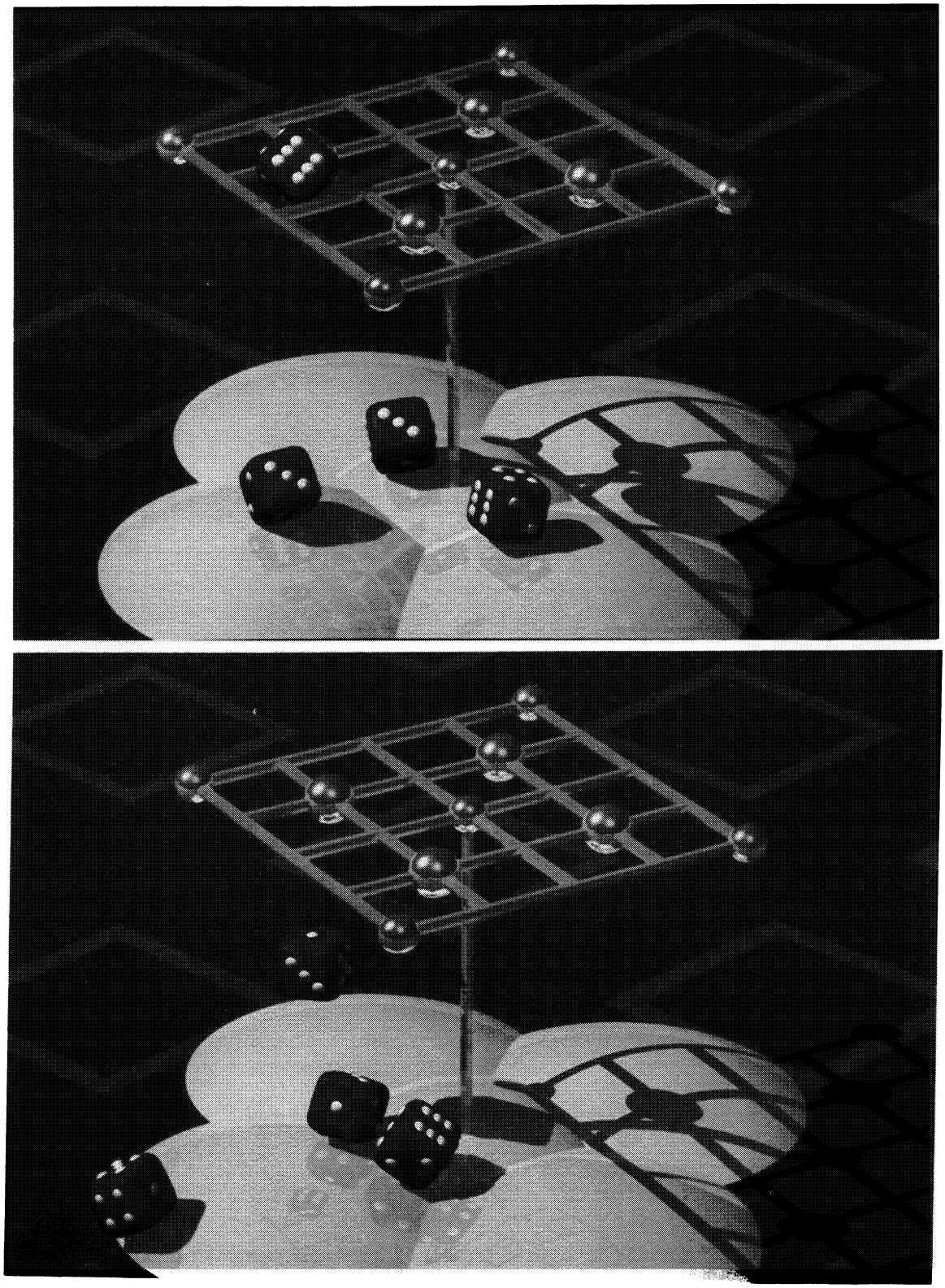


Figure D.2: Falling superquadric dice.

Figure D.2 (Continued)



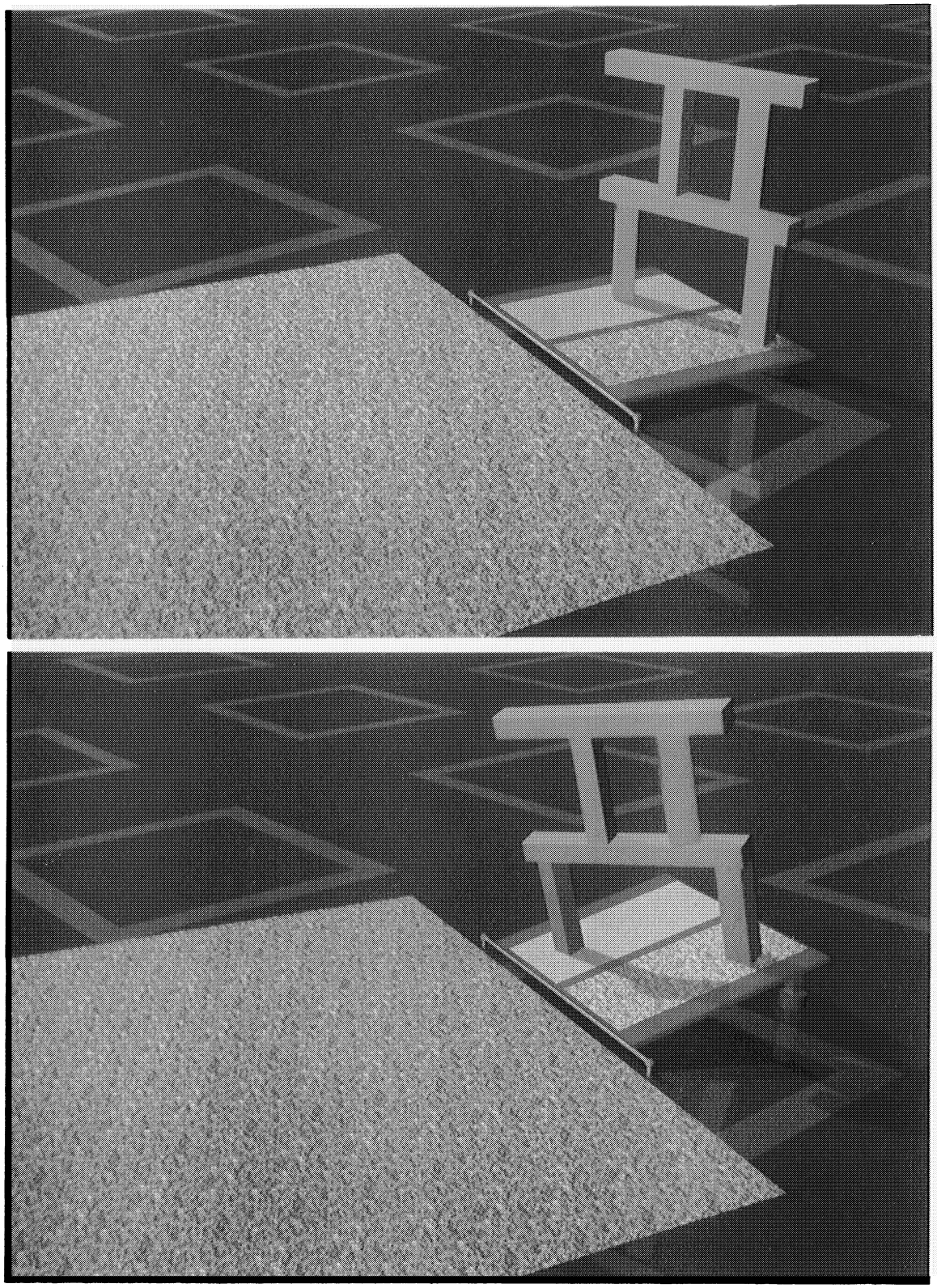
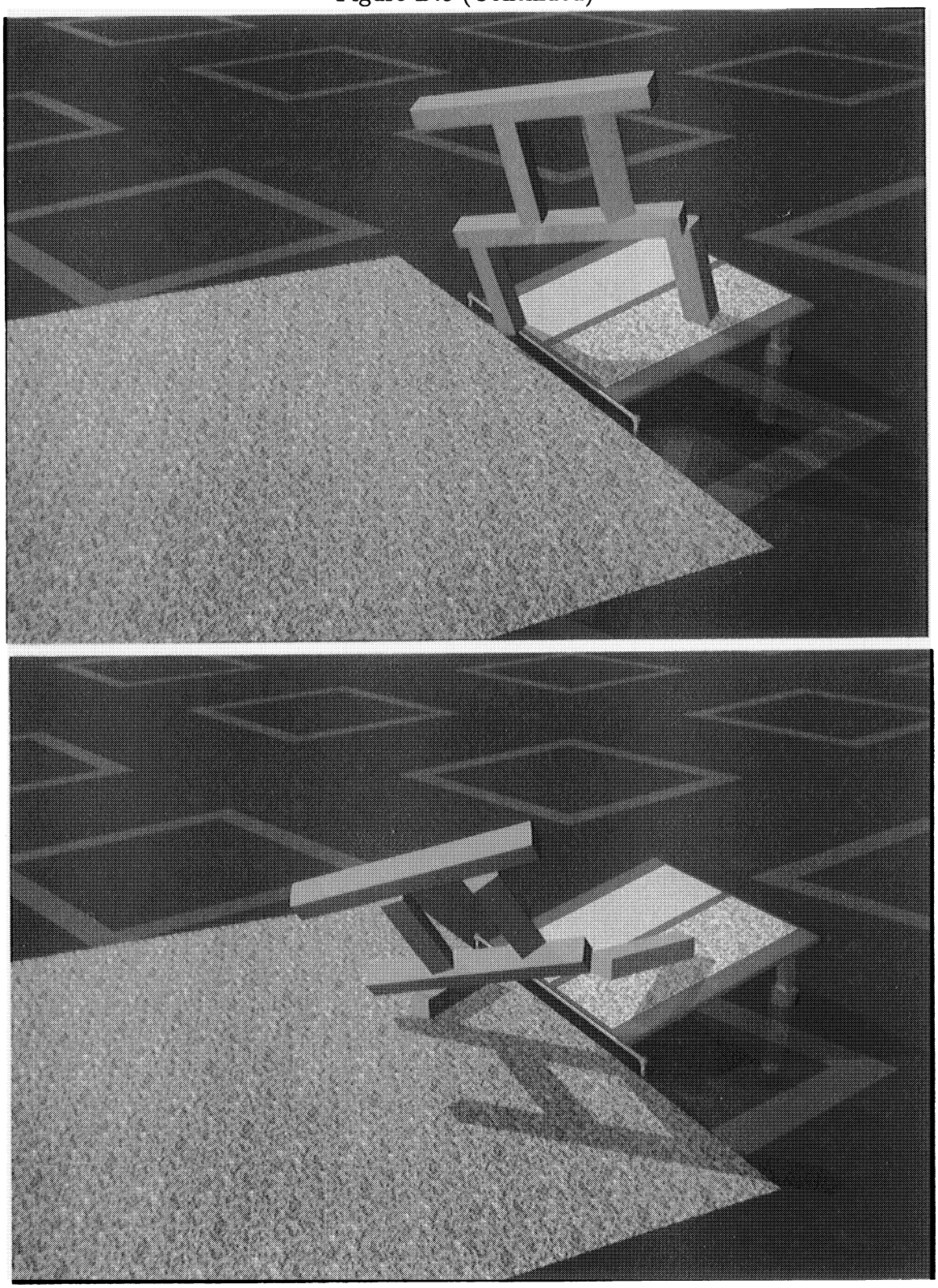


Figure D.3: Collapsing structure.

Figure D.3 (Continued)



Bibliography

[1] V.I. Arnold. *Mathematical Methods of Classical Mechanics.* Springer-Verlag, 1978.

[2] D. Baraff and A. Witkin. Dynamic simulation of non-penetrating flexible bodies. Submitted to *Computer Graphics* (*Proc. SIGGRAPH*), 1992.

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

[4] C. Cai and B. Roth. On the spatial motion of a rigid body with point contact. In *International Conference on Robotics and Automation*, pages 686–695. IEEE, 1987.

[5] 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.

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

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

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

[9] C.W. Cryer. The solution of a quadratic programming problem using systematic overrelaxation. *SIAM Journal on Control*, 9(3):385–392, 1971.

[10] 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.

[11] J.E. Dennis, Jr. and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice Hall, Inc., 1983.

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

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

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

[15] G.E. Forsythe, M.A. Malcolm, and C.B. Moler. *Computer Methods for Mathematical Computations.* Prentice Hall, Inc., 1977.

[16] M.R. Garey and D.S. Johnson. *Computers and Intractability.* Freeman, 1979.

[17] 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.

[18] 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.

[19] 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.

[20] B.J. Gilmore and R.J. Cipra. Simulation of planar dynamic mechanical systems with changing topologies: Part 1—characterization and prediction of the kinematic constraint changes; Part 2—implementation strategy and simulation results for example dynamic systems. In *Advances in Design Automation*, pages 369–388. ASME, 1987.

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

[22] 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.

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

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

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

[26] 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.

[27] C.E. Lemke. Bimatrix equilibrium points and mathematical programming. *Management Science*, 11:681–689, 1965.

[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] T. Lozano-Pérez. Automatic planning of manipulator transfer movements. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-11(10):681–698, 1981.

[32] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120, 1983.

[33] O.L. Mangasarian. Solution of symmetric linear complementarity problems by iterative methods. *Journal of Optimization Theory and Applications*, 22(4):465–485, 1977.

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

[35] 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.

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

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

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

[39] 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.

[40] K.G. Murty. *Linear Complementarity, Linear and Nonlinear Programming.* Heldermann Verlag, 1988.

[41] Ju.I. Neĭmark and N.A. Fufaev. *Dynamics of Nonholonomic Systems.* American Mathematical Society, 1972.

[42] R.S. Palmer. *Computational Complexity of Motion and Stability of Polygons.* PhD thesis, Cornell University, January 1987.

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

[44] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes.* Cambridge University Press, 1986.

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

[46] R. Sedgewick. *Algorithms.* Addison-Wesley, 1983.

[47] L.F. Shampine and M.K. Gordon. *Computer Solution of Ordinary Differential Equations: The Initial Value Problem.* Freeman, 1975.

[48] K. Shoemake. Animating rotation with quaternion curves. In *Computer Graphics* (*Proc. SIGGRAPH*), volume 19, pages 245–254. ACM, July 1985.

[49] A.E. Taylor and R.M. Mann. *Advanced Calculus.* John Wiley & Sons, Inc., 1983.

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

[51] 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.

[52] S.A. Vavasis. Quadratic programming is in *NP.* Technical Report TR 90-1099, Department of Computer Science, Cornell University, 1990.

[53] 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.

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

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

[56] A. Witkin and W. Welch. Fast animation and control of nonrigid structures. In *Computer Graphics (Proc. SIGGRAPH)*, volume 24, pages 243–252. ACM, August 1990.