A Modeling System Based On Dynamic Constraints

Ronen Barzel
Alan H. Barr
California Institute of Technology
Pasadena, CA 91125

Abstract

We present "dynamic constraints," a physically-based technique for constraint-based control of computer graphics models. Using dynamic constraints, we build objects by specifying geometric constraints; the models assemble themselves as the elements move to satisfy the constraints. The individual elements are rigid bodies which act in accordance with the rules of physics, and can thus exhibit physically realistic behavior. To implement the constraints, a set of "constraint forces" is found, which causes the bodies to act in accordance with the constraints; finding these "constraint forces" is an ***inverse dynamics*** problem.

KEYWORDS: Modeling, Dynamics, Constraints, Simulation
CR categories: I.3.5—Computational Geometry and Object Modeling; I.3.7—Three-Dimensional Graphics and Realism

1 Introduction

Some of the most natural and graceful motion in computer animation has been achieved recently by simulating the physical behavior of objects. But physical simulation has not yet become the standard technique for modeling and animation, because of several limitations:

- *Simulations are hard to implement:* Typically, a special-purpose program is written to simulate the behavior of a given computer graphics model; the overhead for making new models is large.
- *Simulations are hard to control:* If "innate" behavior is programmed into models, it becomes hard to make the models do exactly what we want; the behavior is often determined indirectly by non-intuitive or non-orthogonal parameters.
- *Simulations are slow:* Physical simulation can be computationally intensive.

The goal of this work is to develop a modeling system in which it is easy to build and animate physically-based computer graphics models. To this end, our modeling approach is based on four features:

- *Generality:* A model is built from a collection of primitive physically-based elements.
- *Geometric Constraints:* A model is constructed by applying constraints to the objects, starting from an initial configuration of the primitive elements. A model is also positioned and animated through constraints.
- *Newtonian Mechanics:* Each primitive element is a rigid body whose motion is due to the effects of inertia and forces and torques acting on the body. Many of the forces and torques are externally applied; other forces and torques, however, are derived from the geometric constraints.
- *Equivalence of Modeling and Animation:* The temporal behavior of physically based objects is bound up in the model itself.

To implement the constraints, we solve an *inverse dynamics* problem: given constraints on the behavior of the model, the problem is to determine the forces which result in an example of the constrained behavior we desire. Thus, we convert each constraint into a "constraint force"; as the model animates, the constraint forces are continuously computed, to continuously maintain the constraints.

©1988 ACM-0-89791-275-6/88/008/0179 $00.75

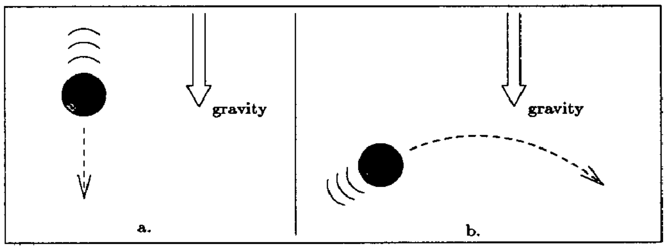

Figure 1: All primitive bodies obey Newton's laws. e.g. (a) A ball released in gravity falls; (b) A ball thrown in gravity moves in an arc.

Sec. 2 of this paper presents the modeling system, and provides implementation notes. Sec. 3 discusses the inverse dynamics problem. Sec. 4 presents the technique for setting up and solving a "constraint-force" equation. The simulation of Newtonian mechanics, derivations for various examples of constraints, and miscellaneous mathematical details are found in the Appendices.

Related Work

[Witkin, Fleischer, and Barr 87] uses "energy" constraints to assemble 3D models, for changing the shape of parametrically-defined primitive objects. This work is not concerned with dynamic mechanical simulation of models. [Platt and Barr 88] uses augmented lagrangian constraints in the physical simulation of flexible objects. [Isaacs and Cohen 87] does physical simulation of rigid bodies, for the special case of linked systems without closed kinematic loops. They share our emphasis on ease of modeling, and also use an inverse-dynamics formulation to control the models' behavior. [Wilhelms and Barsky 85] utilizes physically based modeling, but has a reduced emphasis on control.

2 The Modeling System

Modeling with the "Dynamic Constraints" system consists of instantiating primitive bodies, connecting and controlling them with constraints, and influencing their behavior by explicitly applying external forces. The modeling system thus has three libraries:

- *Primitive bodies:* A collection of rigid bodies, such as spheres, rods, torii, and more complex shapes, that are the component elements of models. The modeler specifies the body density, as well as specific parameters such as the length and radius of a rod. Each body type defines the quantities needed for physical simulation, such as the rotational inertia tensor for that body type.
- *External applied forces:* Forces that the modeler can introduce into the model, including gravity, springs, and damping forces. Each force has parameters specific to the force, such as damping coefficients or spring constants.
- *Constraints:* Various types of geometric constraints, such as "point-to-nail" or "orientation" are described below.

To build a model, we make instances of objects, calculate the forces, and run the simulation. We also specify "timelines" of events to take place, such as creating or removing instances, turning constraints or explicit forces on or off, or otherwise adjusting parameters.

2.1 Newtonian Mechanics

Fig. 1 illustrates one of the simplest examples of a body obeying Newton's laws. This behavior is easy to simulate; Appendix A describes the general newtonian simulation procedure.

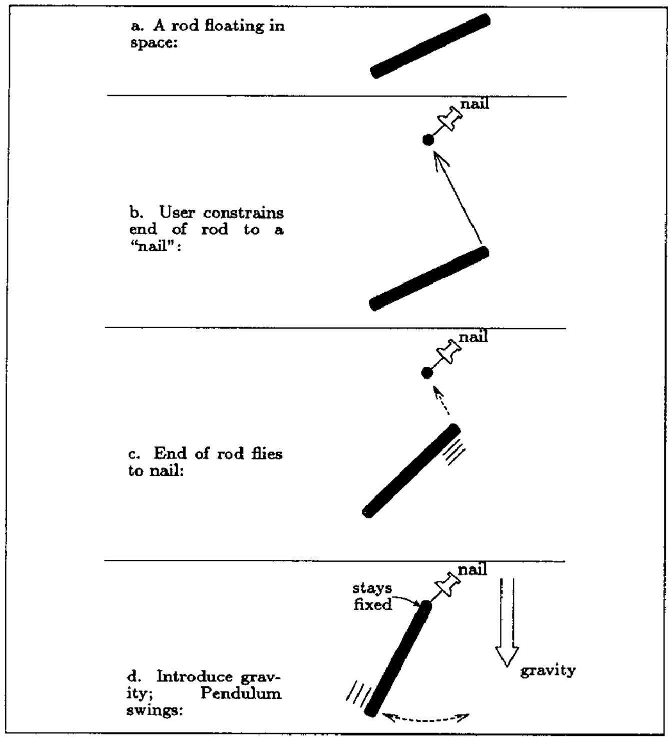
a. A rod floating in space:
nail
b. User constrains end of rod to a "nail":
nail
c. End of rod flies to nail:
nail
stays fixed
gravity
d. Introduce gravity; Pendulum swings:

Figure 2: "Point-to-nail" constraint. A user creates a pendulum by fixing an endpoint of a rod at some location in space. The constraint causes the rod to "fly" into place, assembling the pendulum. See videotape [Caltech '87 Demo Reel].

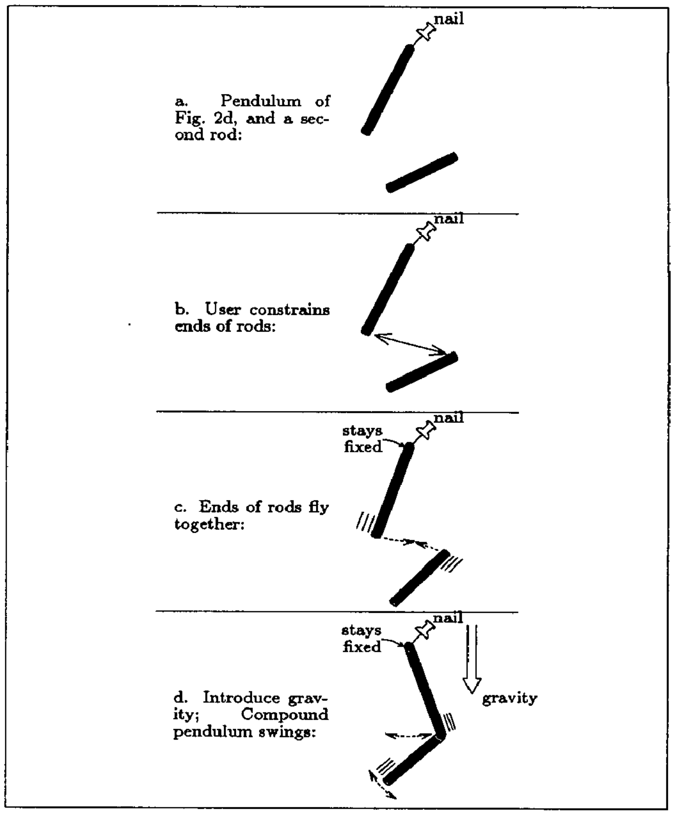
nail
a. Pendulum of Fig. 2d, and a second rod:
nail
b. User constrains ends of rods:
stays fixed
nail
c. Ends of rods fly together:
stays fixed
nail
gravity
d. Introduce gravity; Compound pendulum swings:

Figure 3: A "point-to-point" constraint. A users adds a second rod to the pendulum of Fig. 2, to create a compound pendulum. See videotape [Caltech '87 Demo Reel].

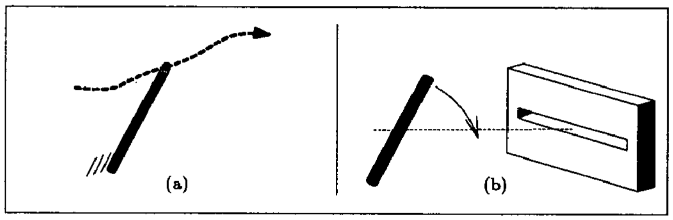

Figure 4: Other constraint examples. (a) "Point-to-path" constraint. This constraint pulls objects along user-specified paths. (b) "Orientation" constraint. We rotate the rod to make its axis parallel to the slot axis.

All primitive bodies in our system exhibit physically realistic behavior, in the sense that they respond correctly to forces and torques.

2.2 Constraints

We show some examples of constraints supported by our modeling system.

- "Point-to-Nail" constraint (see Fig. 2): This fixes a point on a body to a user-specified location in space. The body may swivel and swing about the constrained point, but the constrained point may not move.
- "Point-to-Point" constraint (see Fig. 3): This forms a joint between two bodies. The bodies may move about freely, as long as the two constrained points stay in contact.
- "Point-to-Path" constraint (see Fig. 4a): We can require a point on an object to follow an arbitrary user-specified path; this allows us to animate models by using standard kinematic keyframe techniques.
- "Orientation" constraint (see Fig. 4b): A constraint to align objects by rotating them.
- Other constraints (not illustrated): Other constraints include "point-on-line," which restricts a point to lie on a given line, and "sphere-to-sphere," which requires two spheres to touch, but lets them slide along each other.

We can easily add new types of geometric constraints to our constraint library, by defining the constraint "deviation" function and deriving various required quantities, as described in Appendix C. The only restriction is that the "deviation" function be twice-differentiable (as is discussed in the appendix).

2.3 Constraint Forces

When we have built a model using dynamic constraints, the model is held together by constraint forces, as illustrated in Fig. 5. Thus the constraint forces are analogs of the internal forces which hold the parts of compound objects together.

Constraint forces also assemble the models, pulling the components into the proper configurations. Thus constraint forces represent forces which could be used to assemble real-world objects. For example, figures in the appendices show frames from animations demonstrating the self-assembly of space structures [Barr, Von Herzen, Barzel, and Snyder 87].

2.4 Implementation

We describe here the high-level program structure; details of simulating Newtonian mechanics are given in Appendix A, the procedure to calculate the constraint forces is given in Sec. 4.4.

Our modeling system is implemented in Common Lisp on Symbolics Lisp Machines, using Symbolics' object-oriented "Flavors" mechanism. The fundamental object classes we have defined are:

- *rigid-body:* a primitive body in the model. This class defines the functions and state variables needed for the dynamics calculation (see Appendix A), including a list of forces and torques acting on the body. There are subclasses for each type of body in our library; each subclass provides type-specific information, such as the rotational inertia tensor.
- *control-point:* a point on a body, or in space. A point on a body contains a reference to the body, as well as the position of the point in body-coordinates. A point in space defines its position, which can be constant, or a function of time. Forces and constraints are typically created by specifying the control-points at which they act.
- *force:* a force being applied to a body. Each force contains a reference to a control point at which it is applied. There are subclasses for each type of force in our library; each subclass provides a function that computes the force.
- *constraint:* any type of dynamic constraint. There are subclasses for each type of constraint in our library. Each subclass provides

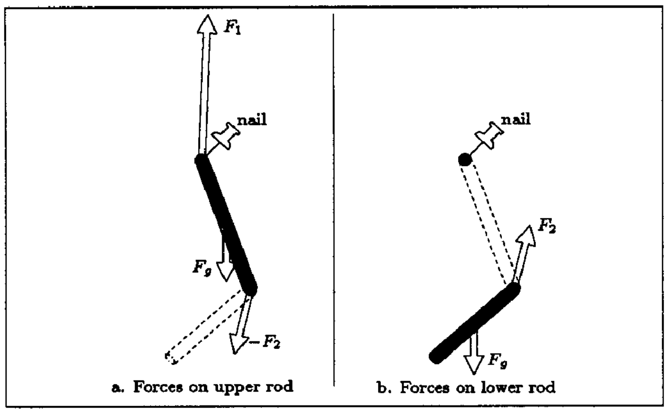

Figure 5: Constraint forces holding together compound pendulum of Fig. 3d. The constraint forces model the internal forces of a real-world pendulum. (a) shows forces on the upper rod, (b) shows forces on the lower rod: F_g is gravity pulling down on rods. F_1 is the "point-to-nail" constraint force on the upper rod, holding it at the nail. F_2 is the "point-to-point" constraint force on the lower rod, holding it to the upper rod; $-F_2$ is the reaction force on the upper rod.

the quantities needed to determine the constraint force (described in Appendix C). Each constraint also keeps references to the bodies being constrained, and associates the appropriate forces with the bodies.

All objects handle a "draw" message, which displays the object in its current state. For debugging a model, we send the "draw" message to all objects, including forces, control points, and constraints; for producing an animation, we send the "draw" message only to bodies.

Some examples of subclasses are:

- *rod:* a subclass of *rigid-body*. This class provides the values specific to rods, e.g. the rotational inertia tensor (see Appendix A). The class also associates two control points with each rod, named "end1" and "end2", at the ends of the rod, and provides functions to access them.
- *nail:* a control point fixed at a location in space.
- *point-to-nail:* a subclass of *constraint*. Provides the functions which calculate the terms needed for a "point-to-nail" constraint (see Sec. 4, Example 1).

The addition of new types of bodies, forces, or constraints to the system merely requires the creation of an appropriate new subclass.

Currently, the user-interface is via the lisp environment; for example, the pendulum of Fig. 3 could be built via the series of commands:

```
; create bodies and control points
(make rod "upper-rod")
(make rod "lower-rod")
(make nail "nail" 0 0 100)
; specify constraints
(connect (end1 "upper-rod") "nail")
(connect (end2 "upper-rod") (end1 "lower-rod"))
; add external forces
(gravity-on) ; apply gravity to each body
```

To animate a model once the instances are made, we simply iterate these steps:

- Simulate until end of frame (Appendix A).
- Send "draw" message to objects.

The implementation makes heavy use of a home-grown package of numerical routines, which include linear-system solvers, differential equation solvers, and the like; some useful references are [Press et. al. 88,Golub and Van Loan 83,Ralston and Rabinowitz 78, Boyce and Deprima 77]. We also have embedded into lisp an extension to "Einstein Summation Notation" for mathematical expressions [Barr 83,Misner, Thorne and Wheeler 73]; this makes it quite simple to create lisp functions by merely typing in the mathematical formulae using the same notation with which we derive them.

3 Inverse Dynamics

If we are given the forces which act on a collection of objects, we can easily solve the *forward dynamics* problem—that of determining the objects' behavior—as described in Appendix A. However, to meet constraints, we must solve the reverse problem: Given a partial description of the desired behavior, we must determine forces which will yield an appropriate behavior. This *inverse dynamics* problem, summarized in Fig. 6, consists of two parts: (a) finding forces to meet a constraint, and (b) finding forces to maintain a constraint.

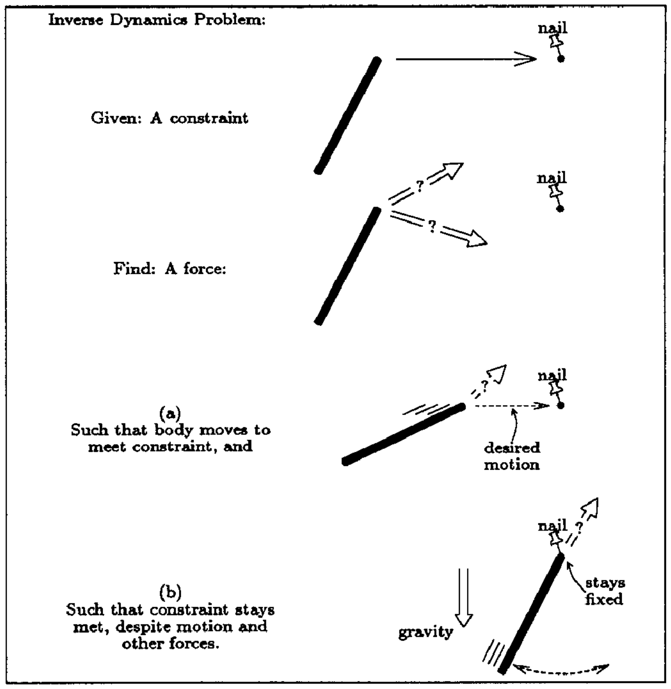
Inverse Dynamics Problem:
Given: A constraint
nail
Find: A force:
?
?
nail
(a)
Such that body moves to meet constraint, and
?
nail
desired motion
(b)
Such that constraint stays met, despite motion and other forces.
nail
?
stays fixed
gravity

Figure 6: The inverse dynamics problem for dynamic constraints.

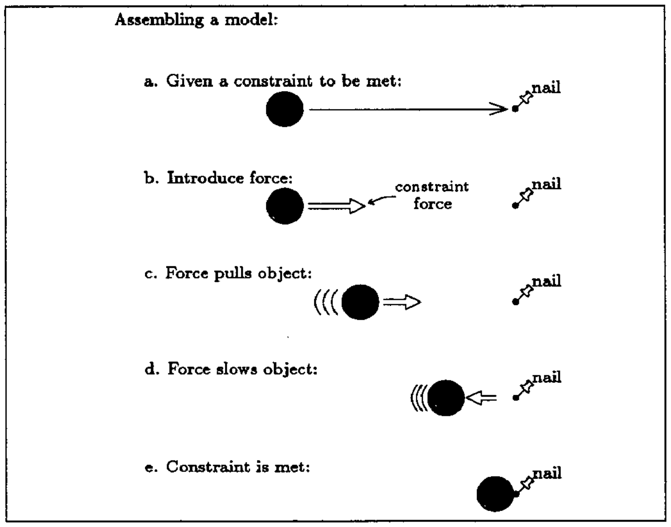

Figure 7: Meeting a Constraint. The constraint force pulls the ball towards the nail, then brings the ball to rest at the nail.

Meeting A Constraint

Fig. 7 shows a constraint force being used satisfy to part (a) of the inverse dynamics problem, that of moving the objects to meet an initially unmet constraint. Notice that this part of the problem is actually very loosely specified: How quickly should the constraint be met? Along what path should the object move? For our solution, as we shall see in Sec. 4.2, the "deviation" of the constrained point decays exponentially, with a user-specified time constant.

Maintaining A Dynamic Constraint

Fig. 8 shows a dynamic constraint force adapting to satisfy part (b) of the inverse dynamics problem, that of keeping a constraint met despite motion and other forces. There is typically a unique solution, in which the dynamic constraint forces provide the internal forces that hold together an object.

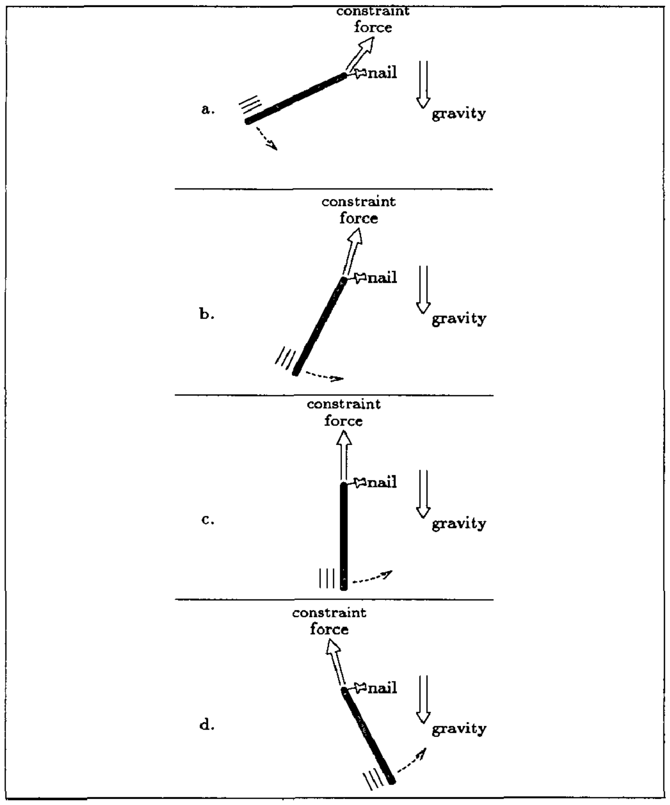

Figure 8: Maintaining a Constraint. (a-d) The constraint force adapts to hold constraint even as object moves and other forces act on it. The constraint force pulls up, to counteract gravity, and sideways, to keep the pendulum's inertia from flinging it sideways off the nail.

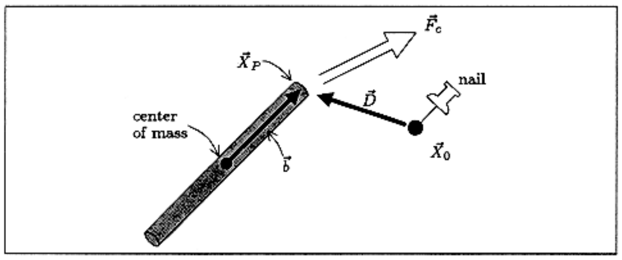

Figure 9: "Point-to-nail" constraint. The "deviation" measure is $\vec{D}(\mathcal{Y}) = \vec{X}_P(\mathcal{Y}) - \vec{X}_0$, the constraint force is $\vec{F}_c$, and the constraint torque is $\vec{b} \times \vec{F}_c$.

4 Calculating Constraint Forces

Note: We suggest skimming Appendix A, to become familiar with our notation and formulation of rigid-body mechanics, before reading this section.

In this section, we present the technique for computing the constraint forces. The presentation is in several sections:

1. Definition of several mathematical quantities associated with each constraint.
2. Construction of a "constraint-force" equation for a dynamic constraint; if the constraint force is chosen such that this equation holds, the constrained objects will behave in accordance with the dynamic constraint.
3. Grouping the constraint-force equations for multiple constraints into a single multidimensional constraint-force equation for all the dynamic constraints.
4. Setting up and solving the dynamic constraint-force equation.

4.1 Definitions

Fig. 11 gives the definition of the quantities which must be supplied for each constraint. The derivations of these quantities for various types of constraints are given in Appendix C.

$\vec{D}$	A measure of "deviation" for the constraint: $\vec{D}(\mathcal{Y},t) = 0 \Longleftrightarrow$ constraint is met $\vec{D}$ is a d-dimensional vector.
d	The number of dimensions of $\vec{D}$.
$\vec{D}^{(1)}$	The rate of change of $\vec{D}$: $\vec{D}^{(1)}(\mathcal{Y}(t),t) \equiv \frac{d}{dt}\vec{D}(\mathcal{Y}(t))$ $\vec{D}^{(1)}$ is a d-dimensional vector.
$\vec{D}^{(2)}$	The acceleration of $\vec{D}$: $D^{(2)}(\mathcal{Y}(t),\mathcal{F}(t),\mathcal{T}(t),t) \equiv \frac{d^2}{dt^2}\vec{D}(Y(t),t)$ $D^{(2)}$ will depend linearly on $\mathcal{F}$ and $\mathcal{T}$; thus: $D^{(2)} = \sum_{\text{bodies } i} (\mathbf{\Gamma}^i(\mathcal{Y})\vec{F}^i + \mathbf{\Lambda}^i(\mathcal{Y})\vec{T}^i) + \vec{\beta}(\mathcal{Y})$ where we define: $\mathbf{\Gamma}^i$ — A $d \times 3$ matrix corresponding to the net force on body i; we have one such for each body in the constraint. $\mathbf{\Lambda}^i$ — A $d \times 3$ matrix corresponding to the net torque on body i; we have one such for each body in the constraint. β — The part of $\vec{D}^{(2)}$ independent of $\mathcal{F}$ and $\mathcal{T}$; a d-dimensional vector.
f	The number of degrees of freedom in the constraint force.
$\mathbf{G}^i$	A $3 \times f$ matrix. The constraint force on body i is given by $\mathbf{G}^i\vec{F}_c$ We have one such matrix for each body in the constraint.
$\mathbf{H}^i$	A $3 \times f$ matrix. The constraint torque on body i is given by $\mathbf{H}^i\vec{F}_c$ We have one such matrix for each body in the constraint.

Figure 11: Quantities associated with a dynamic constraint. $\vec{F}_c$ is the unknown "constraint force" for the constraint. $\mathcal{Y}$, $\mathcal{F}$, and $\mathcal{T}$ are the state, net force, and net torque in the model, as defined in Eqn. 9 (Appendix A). See discussion in Sec. 4.1. Derivations of these quantities for various constraints are given in Appendix C

We also define

$$\begin{array}{lll} \vec{F}_c & \text{—} & \text{The unknown "constraint force,"} \\ & & \text{an } f\text{-dimensional vector.} \\ \vec{F}^i_E & \text{—} & \text{The net external applied force on the } i\text{-th body} \\ \vec{T}^i_E & \text{—} & \text{The net external applied torque on the } i\text{-th body} \end{array} \tag{1}$$

Note that strictly speaking, $\vec{F}_c$ is not necessarily a force, but rather is a quantity that determines the constraint force ($= \mathbf{G}^i\vec{F}_c$) and constraint torque ($= \mathbf{H}^i\vec{F}_c$) on the constrained bodies; colloquially, however, we refer to $\vec{F}_c$ as the "constraint force." The vectors $\vec{F}^i_E$ and $\vec{T}^i_E$ are due to the external forces, such as gravity, that act on the i-th body body.

The net force (torque) on a body is the sum of the net external force (torque) and the constraint forces (torques):

$$\begin{aligned} \vec{F}^i &= \Big(\sum_{\text{constraints } j} \mathbf{G}^i_j\vec{F}_{c_j}\Big) + \vec{F}^i_E \\ \vec{T}^i &= \Big(\sum_{\text{constraints } j} \mathbf{H}^i_j\vec{F}_{c_j}\Big) + \vec{T}^i_E \end{aligned} \tag{2}$$

In the above equations, we label terms for the ith body with superscript i's, and the jth constraint with subscript j's.

See Example 1 for a description of terms needed for a "point to nail" constraint.

4.2 Constraint-force Equation

For each constraint, we describe the desired behavior of the constraint "deviation" by linearly combining $\vec{D}$, $\vec{D}^{(1)}$, and $\vec{D}^{(2)}$:

$$\vec{D}^{(2)}(\mathcal{Y},\mathcal{F},\mathcal{T},t) + \frac{2}{\tau}\vec{D}^{(1)}(\mathcal{Y},t) + \frac{1}{\tau^2}\vec{D}(\mathcal{Y},t) = 0, \ t \geq t_0 \tag{3}$$

This equation is equivalent to the differential equation in Fig. 12; its solution brings $\vec{D}$ down to 0, then holds it at 0.[1] We will substitute for $D^{(2)}$, $D^{(1)}$ and D in Eqn. 3 to produce a linear system of equations in which we solve for $\vec{F}_c$. If we continually adjust the force so that Eqn. 3 is met, we will be solving the inverse dynamics problem.

Thus, we expand $\vec{D}^{(2)}$ in Eqn. 3, using the definitions in Fig. 11, yielding:

$$\sum_{\text{bodies } i} (\mathbf{\Gamma}^i\vec{F}^i + \mathbf{\Lambda}^i\vec{T}^i) + \vec{\beta} + \frac{2}{\tau}\vec{D}^{(1)} + \frac{1}{\tau^2}\vec{D} = 0 \tag{4}$$

[1]Analytically, if $\vec{D}_0 \neq 0$ the solution to the equation in Fig. 12 asymptotically approaches 0, but doesn't ever reach $\vec{D} = 0$. Numerically, however, we soon reach $\vec{D} = 0$ within error tolerances.

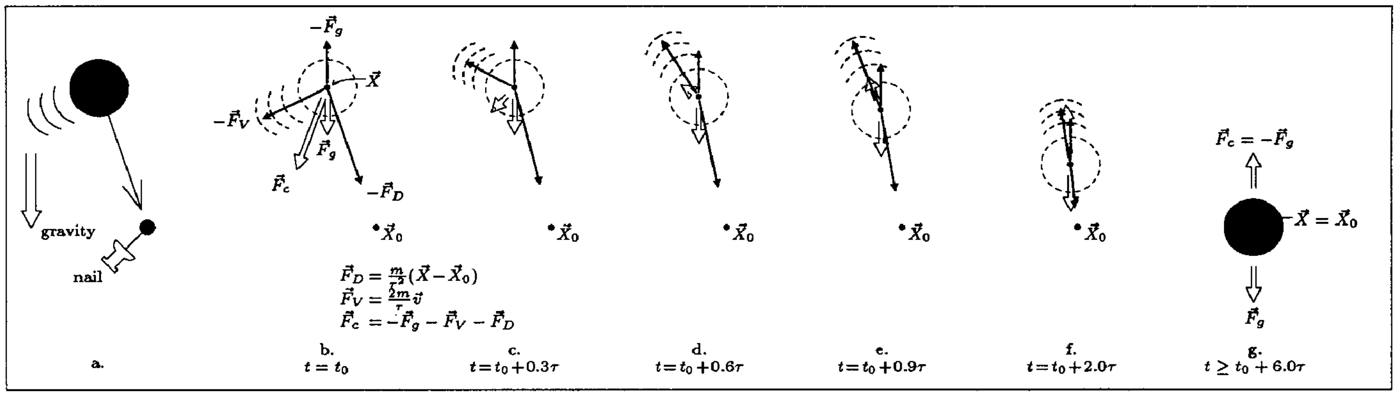

Figure 10: Constraint-force calculation for a "point-to-nail" constraint (details in Sec. 4.2): Constraint force has components opposing gravity ($-\vec{F}_g$), opposing motion ($-\vec{F}_v$), and pulling towards nail ($-\vec{F}_D$). (a) User specifies constraint at center of mass. (b) Constraint force initially pulls towards nail. (c-e) Once ball is moving towards nail, constraint force turns around. (f) Constraint force slows ball. (g) Steady-state: $\vec{F}_D = \vec{F}_V = 0$, $\vec{F}_c = -\vec{F}_g$. Net force on ball is 0; by Newton's first law, the ball remains at rest.

Example 1: "Point-to-Nail" Constraint. We choose $\vec{D}$ to be the vector from the constrained point, at $\vec{X}_P$, to the "nail", at $\vec{X}_0$ (see Fig. 9). We thus have:

$$\begin{aligned} d &= 3 \\ \vec{D}(\mathcal{Y}) &= \vec{X}_P - \vec{X}_0 \end{aligned}$$

The differentiation in Appendix B.2 immediately gives us the quantities:

$$\begin{aligned} \vec{D}^{(1)}(\mathcal{Y}) &= \vec{v}_P(\mathcal{Y}) \\ \vec{D}^{(2)}(\mathcal{Y}, \vec{F}, \vec{T}) &= \vec{a}_P(\mathcal{Y}, \vec{F}, \vec{T}) \\ \mathbf{\Gamma} &= \tfrac{1}{m} \\ \mathbf{\Lambda} &= \mathbf{b}^{*\mathsf{T}}\mathbf{I}^{-1} \\ \vec{\beta} &= \mathbf{b}^{*\mathsf{T}}\mathbf{I}^{-1}(\vec{L} \times \vec{\omega}) + \vec{\omega} \times (\vec{\omega} \times \vec{b}) \end{aligned}$$

We apply an arbitrary force to the constrained point. The "constraint force" $\vec{F}_c$ is used directly as a force on the body, and it contributes $\vec{b} \times \vec{F}_c = \mathbf{b}^* \vec{F}_c$ to the torque. Thus we define:

$$\begin{aligned} f &= 3 \\ \mathbf{G} &= 1 \quad \textit{SeeAppendix Cforotherexamples.} \\ \mathbf{H} &= \mathbf{b}^* \end{aligned}$$

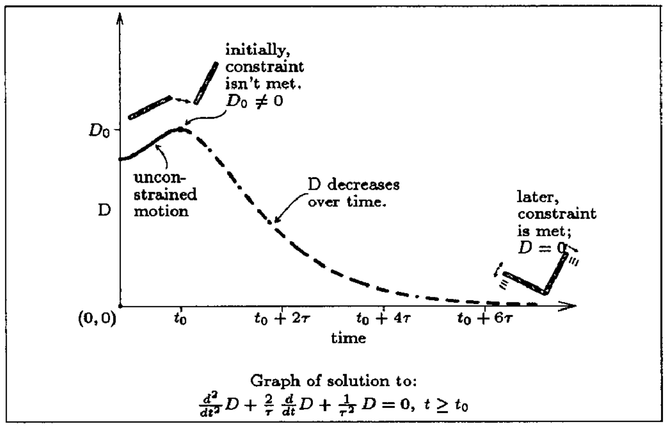

Figure 12: D evolving over time. We have picked a second-order differential equation to describe D as a function of time. The solution yields the behavior required by the inverse dynamics problem—the constraint "deviation" decays down to 0, assembling the model, then remains at 0, maintaining the constraint. The rate of assembly is controlled by the time constant τ. See [Boyce and Deprima 77] for a discussion of second-order differential equations.

Substituting Eqn. 2 into Eqn. 4, we get the constraint-force equation for a single constraint:

Example 2: "Point-to-Nail" Constraint. In Fig. 10, we illustrate a simple case: A single body, with a "point-to-nail" constraint acting at its center-of-mass, and with gravity. Since both the constraint force and gravity act on the ball's center-of-mass, there are no rotational terms. Thus the quantities in Example 1 reduce to:

$$\begin{aligned} \vec{b} = \mathbf{b}^* = \mathbf{\Lambda} = \vec{\beta} = \mathbf{H} = \vec{T}_E &= 0 \\ \vec{D} &= \vec{X} - \vec{X}_0 \\ \vec{D}^{(1)} &= \vec{v} \\ \vec{D}^{(2)} &= \tfrac{1}{m}\vec{F} \\ \mathbf{\Gamma} &= \tfrac{\mathbf{1}}{m} \\ \mathbf{G} &= \mathbf{1} \\ \vec{F}_E &= \vec{F}_g \end{aligned}$$

Substituting into Eqn. 5 gives:

$$\frac{1}{m}\vec{F}_c + \frac{1}{m}\vec{F}_g + \frac{2}{\tau}\vec{v} + \frac{1}{\tau^2}(\vec{X} - \vec{X}_0) = 0$$

We easily solve for the constraint force:

$$\vec{F}_c = -\vec{F}_g - \frac{2}{\tau}m\vec{v} - \frac{1}{\tau^2}m(\vec{X} - \vec{X}_0)$$

Thus we see the constraint force has three components: One opposing the force of gravity, one opposing the ball's velocity, and one pulling the ball towards the nail. Fig. 10 illustrates the constraint force adapting to pull the ball to the nail, and bring it to rest. Once the ball is at rest at the nail, we have $\vec{X} - \vec{X}_0 = 0$ and $\vec{v} = 0$; so the constraint force becomes $\vec{F}_c = -\vec{F}_g$, yielding a net force on the ball of $\vec{F} = \vec{F}_c + \vec{F}_g = 0$

$$\begin{aligned} &\sum_{\text{constraints } j} \Big(\sum_{\text{bodies } i} (\mathbf{\Gamma}^i \mathbf{G}^i_j + \mathbf{\Lambda}^i \mathbf{H}^i_j) \Big) \vec{F}_{c_j} \\ &\quad + \sum_{\text{bodies } i} (\mathbf{\Gamma}^i \vec{F}^i_E + \mathbf{\Lambda}^i \vec{T}^i_E) \qquad = 0 \qquad (5) \\ &\quad + \vec{\beta} + \frac{2}{\tau}\vec{D}^{(1)} + \frac{1}{\tau^2}\vec{D} \end{aligned}$$

Notice that, to meet this one constraint, we must take into account the effect of all the constraint forces.

See Example 2 for a sample derivation of the constraint forces.

4.3 Multiple Constraints

Each constraint results in a version of Eqn. 5; with several constraints, we have a set of simultaneous equations which must be solved.

We duplicate Eqn. 5, for each constraint in the model:

$$\begin{aligned} &\sum_{\text{constraints } j} \Big(\sum_{\text{bodies } i} (\mathbf{\Gamma}^i_k \mathbf{G}^i_j + \mathbf{\Lambda}^i_k \mathbf{H}^i_j) \Big) \vec{F}_{c_j} \\ &\quad + \sum_{\text{bodies } i} (\mathbf{\Gamma}^i_k \vec{F}^i_E + \mathbf{\Lambda}^i_k \vec{T}^i_E) \qquad = 0, \Big\{ \begin{matrix} \text{for all} \\ \text{constraints } k \end{matrix} \qquad (6) \\ &\quad + \vec{\beta}_k + \frac{2}{\tau_k}\vec{D}^{(1)}_k + \frac{1}{\tau^2_k}\vec{D}_k \end{aligned}$$

where we label terms for the kth constraint with subscript k's. Writing this system of equations more compactly, as a multidimensional vector equation, we have the constraint force equation for the model:

$$\boxed{\mathcal{M}\mathcal{F}_c + \mathcal{B} = 0}$$

where

$$\begin{aligned} \mathcal{M}_{kj} &= \sum_{\text{bodies } i} (\mathbf{\Gamma}^i_k \mathbf{G}^i_j + \mathbf{\Lambda}^i_k \mathbf{H}^i_j) \\ \mathcal{F}_{c_j} &= \vec{F}_{c_j} \\ \mathcal{B}_k &= \sum_{\text{bodies } i} (\mathbf{\Gamma}^i_k \vec{F}^i_E + \mathbf{\Lambda}^i_k \vec{T}^i_E) \\ &\quad + \vec{\beta}_k + \tfrac{2}{\tau_k} \vec{D}^{(1)}_k + \tfrac{1}{\tau_k^2} \vec{D}_k \end{aligned} \tag{7}$$

Fig. 13 illustrates collecting individual constraint equations into the multidimensional vector equation. Notice that each element of $\mathcal{M}$ is a matrix, and each element of $\mathcal{F}_c$ and of $\mathcal{B}$ is a vector.

4.4 Solving the Constraint-Force Equation

Fig. 15 outlines the procedure to set up Eqn. 7, as well as solve it and compute the net force and torque on each body.

In step 3 of Fig. 15, we call a standard linear-system solver to solve Eqn. 7. There are many well-known methods for solving linear systems (see [Press et. al. 88,Ralston and Rabinowitz 78]). We note some characteristics of $\mathcal{M}$ that should be taken into account when choosing a solution method:

- $\mathcal{M}$ is typically sparse. The $[k, j]$th entry in $\mathcal{M}$ is non-0 only if some body is influenced by both constraint k and constraint j. Typically, most of the elements are zero.
- $\mathcal{M}$ is not necessarily square. A constraint may have $d \neq f$; for example, the "orientation" constraint (Appendix C.3) has $d = 1$ and $f = 3$, yielding a matrix which is "wider" than it is "tall."
- $\mathcal{M}$ may be singular, implying that Eqn. 7 is overconstrained or underconstrained.[2]

We most often use singular-value decomposition (SVD) to solve Eqn. 7, because it robustly handles singularity and near-singularity, as well as non-square systems. However, SVD does not take advantage of sparseness, and is a relatively slow technique.

Underconstrained Equations

Constraint-force equation Eqn. 7 will sometimes be underconstrained, thus having many solutions. This can occur, for example, when there are several constraints acting on a single body; it may be possible to vary some of the individual constraint forces without affecting the net torque or force on the object. An example is shown in Fig. 14a, in which the pair of forces labeled "V" yield the same net force ($= 2\vec{F}_V$) and torque ($= 0$) as the pair labeled "W".

There is no difficulty caused by having many solutions to Eqn. 7; we could use any solution, since they will all yield satisfactory behavior. We might wish to use the solution which is smallest in magnitude, to avoid numerical difficulties; SVD yields this solution.

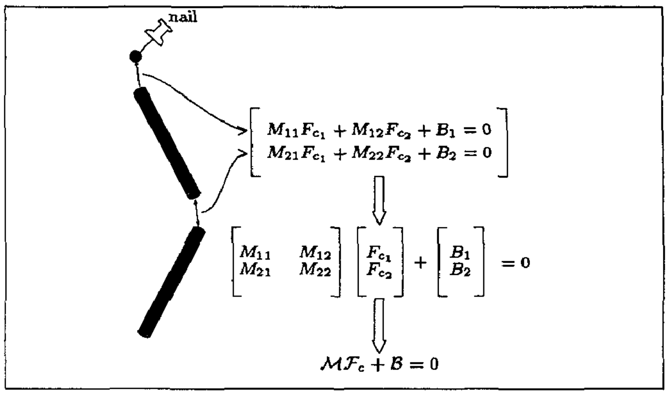
nail
$\mathcal{M}_{11}\mathcal{F}_{c_1} + \mathcal{M}_{12}\mathcal{F}_{c_2} + \mathcal{B}_1 = 0$
$\mathcal{M}_{21}\mathcal{F}_{c_1} + \mathcal{M}_{22}\mathcal{F}_{c_2} + \mathcal{B}_2 = 0$
$\begin{bmatrix} \mathcal{M}_{11} & \mathcal{M}_{12} \\ \mathcal{M}_{21} & \mathcal{M}_{22} \end{bmatrix} \begin{bmatrix} \mathcal{F}_{c_1} \\ \mathcal{F}_{c_2} \end{bmatrix} + \begin{bmatrix} \mathcal{B}_1 \\ \mathcal{B}_2 \end{bmatrix} = 0$
$\mathcal{M}\mathcal{F}_c + \mathcal{B} = 0$

Figure 13: Multiple constraints: Each constraint contributes one line to the equation. The collection of constraints together yields a set of simultaneous linear equations, expressible as a linear matrix equation.

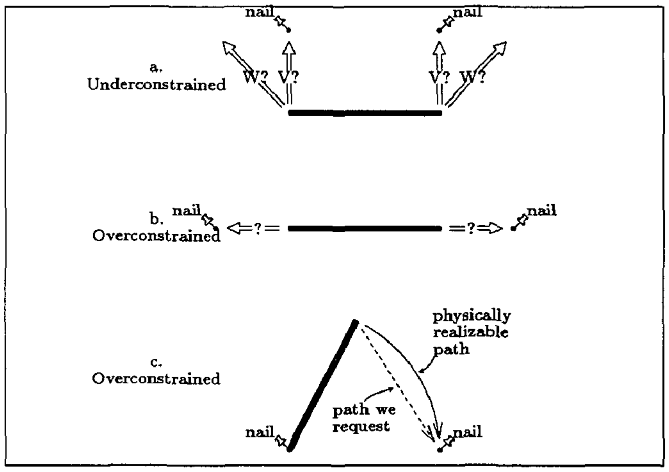

Figure 14: Under- and Overconstrained systems. (a) Underconstrained: Forces "V" and "W" yield the same net force. (b) Overconstrained: There is no way to meet both constraints. (c) Overconstrained: Both constraints could be met, but not via the path we have chosen.

Overconstrained Equations

eats In Fig. 14b, the user has specified constraints which can not be met; there is no "correct" constraint force to be applied. In Fig. 14c, the specified constraints can be met, but not by moving the constrained point in a straight line; however, Eqn. 3 requires that the point move in straight line if the constrained point is initially at rest.[3]

For overconstrained systems, using the least-squares solution for the constraint forces typically yields "reasonable" behavior – the object typically assumes some intermediate configuration, for the case of Fig. 14b, or moves along the feasible path, for the case of Fig. 14c. SVD computes the least-squares solution for overconstrained systems.

5 Summary

We have developed a modeling system featuring constraint-based control of rigid bodies. The bodies' behavior is determined by simulation of Newtonian mechanics. We compute dynamic "constraint forces" to apply to the bodies such that they behave in accordance with user-specified geometric constraints; the computation of these forces is an *inverse dynamics* problem.

The modeling system supports various types of geometric constraints, such as "point-to-nail" and "point-to-point." The modeler builds objects by using constraints to connect primitive components; the constraint forces cause the components to assemble themselves into the model, and ensure that the model stays assembled as it animates.

We have developed a technique to compute the constraint forces by setting up and solving a "constraint-force equation." The constraint-force equation is a multidimensional linear equation of the form $\mathcal{M}\mathcal{F}_c + \mathcal{B} = 0$, where $\mathcal{F}_c$ is the collection of unknown constraint forces.

Each constraint is described by a "deviation" measure $\vec{D}$, such that $\vec{D} = 0$ when the constraint is met. $\vec{D}$ must be a twice differentiable function of the positions and orientations of the constrained bodies. Appendix C derives $\vec{D}$ for several types of constraints.

6 Future Work

Further work we are interested in pursuing includes:

- *Expanding the constraint library.* Deriving new constraint "deviation" functions, as described in Appendix C.
- *Interactive graphical modeler.*
- *Object Intersection.* Development of non-interpenetration constraints
- *Flexible bodies.* Incorporation of flexible-body simulation with dynamic constraint control [Platt and Barr 88, Terzopoulos, Platt, Fleischer, and Barr 87].
- *Special-case models* Direct implementation of the equations of motion for common objects, such as the linked systems of [Isaacs and Cohen 87,Armstrong and Green 85]. Decreasing the number of constraints in the model speeds up the constraint-force calculation.
- *Constraints on velocity or acceleration.*

We are also looking forward to using the dynamic constraints modeling system as a tool in other research areas, such as molecular biology [Lengyel 87] and robotics.

[2] Unfortunately, we have some overloading of the word "constrain": "overconstrained" and "underconstrained" refer to the linear system of equations Eqn. 7, rather than to the constraints themselves.

[3] A solution to the problem of unrealizable paths is to use scalar constraint measures ($d = 1$). For example, the "point-to-nail" constraint could be redefined so that $\vec{D}$ is the distance from the point to the nail, rather than the vector separating the point and the nail.

```
PROCEDURE TO COMPUTE FORCE AND TORQUE ON EACH BODY:

; 1. Compute net explicit forces and torques
for each body i
    $\vec{F}_E^i$ = $\vec{T}_E^i$ = 0
    for each explicit force j on body i
        compute force $\vec{F}_{E_j}$
        $\vec{F}_E^i$ += $\vec{F}_{E_j}$
        $\vec{T}_E^i$ += $\vec{b}_j \times \vec{F}_{E_j}$
    end
    for each explicit torque j on body i
        compute torque $\vec{T}_{E_j}$
        $\vec{T}_E^i$ += $\vec{T}_{E_j}$
    end
end
; 2. Set up constraint-force equation
initialize $\mathcal{M}$ to 0
for each constraint k
    compute $\vec{\beta}_k$, $D_k^{(1)}$, and $D_k$
    $\mathcal{B}$[k] = $\vec{\beta}_k + \frac{2}{\tau_k}\vec{D}_k^{(1)} + \frac{1}{\tau_k^2}\vec{D}_k$
    for each body i in constraint k
        compute $\mathbf{\Gamma}_k^i$ and $\mathbf{\Lambda}_k^i$
        $\mathcal{B}$[k] += $\mathbf{\Gamma}_k^i\vec{F}_E^i + \mathbf{\Lambda}_k^i\vec{T}_E^i$
        for each constraint j acting on i
            $\mathcal{M}$[k,j] += $\mathbf{\Gamma}_k^i\mathbf{G}_j^i + \mathbf{\Lambda}_k^i\mathbf{H}_j^i$
        end
    end
end
; 3. Solve constraint-force equation (Eqn. 7)
$\mathcal{F}_c$ = solve($\mathcal{M}$,$\mathcal{B}$) ;linear-system solver
; 4. Compute net forces and torques.
for each body i
    $\vec{F}^i$ = $\vec{F}_E^i$
    $\vec{T}^i$ = $\vec{T}_E^i$
    for each constraint j acting on i
        $\vec{F}^i$ += $\mathbf{G}_j^i\mathcal{F}_c$[j]
        $\vec{T}^i$ += $\mathbf{H}_j^i\mathcal{F}_c$[j]
    end
end
```

Figure 15: The procedure to compute the constraint forces and the net force and torque on each body. See discussion in Sec. 4.4. Note that the [k,j]th element of $\mathcal{M}$ is a $d_k \times f_j$ matrix, and the [k]th element of $\mathcal{B}$ is a d_k-dimensional vector. Rather than implementing $\mathcal{M}$ as a "nested" array, it can be flattened into a $(\sum d) \times (\sum f)$ array; similarly, $\mathcal{B}$ is formed by concatenating the individual vectors into one $(\sum d)$-dimensional vector.

State variables of a body:

m	- Mass of the body
$\mathbf{I}$	- Rotational inertia tensor of the body
$\vec{x}$	- Position of the body
$\mathbf{R}$	- Orientation of the body
$\vec{p}$	- Momentum of the body
$\vec{L}$	- Angular Momentum of the body,

auxiliary variables:

$\vec{v} = \frac{1}{m}\vec{p}$	– Velocity of the body
$\vec{\omega} = \mathbf{I}^{-1}\vec{L}$	– Angular velocity of the body,

THE EQUATIONS OF MOTION:

$$\frac{d}{dt}\vec{x} = \vec{v}$$
$$\frac{d}{dt}\vec{p} = \vec{F}$$
$$\frac{d}{dt}\mathbf{R} = \omega^*\mathbf{R}$$
$$\frac{d}{dt}\vec{L} = \vec{T},$$

where:

$\vec{F}$ = net force on the body, *and*

$\vec{T}$ = net torque of the body.

Note: ω^* is the dual of $\vec{\omega}$ (see Appendix B.1).

Figure 16: Summary of the equations of motion of a rigid body.

References

[Armstrong and Green 85] Armstrong, William W., and Mark W. Green, *The dynamics of articulated rigid bodies for purposes of animation*, in **Visual Computer**, Springer-Verlag, 1985, pp. 231-240.

[Barr 83] Barr, Alan H., *Geometric Modeling and Fluid Dynamic Analysis of Swimming Spermatozoa*, Ph.D. Dissertation, Rensselaer Polytechnic Institute, 1983

[Barr 88] Barr, Alan H., *Topics in Physically Based Modeling, to appear, Addison Wesley*

[Barr, Von Herzen, Barzel, and Snyder 87] Barr, Alan H., Brian Von Herzen, Ronen Barzel, and John Snyder, *Computational Techniques for the Self Assembly of Large Space Structures* Proceedings of the 8th Princeton/SSI Conference on Space Manufacturing, Princeton New Jersey, May 6-9 1987, to be published by the American Institute of Aeronautics and Astronautics.

[Boyce and Deprima 77] Boyce, William E., and DiPrima, Richard C., **Elementary Differential Equations and Boundary Value Problems**, John Wiley & Sons, New York, 1977.

[Caltech '87 Demo Reel] Caltech studies in modeling and motion (videotape), in SIGGRAPH video Review #28, *Visualization in Scientific Computing Computer Graphics*, volume 21 number 6. ACM SIGGRAPH, 1987

[Fox 67] Fox, E.A., **Mechanics**, Harper and Row, New York, 1967

[Gear 71] Gear, C. William, **Numerical Initial Value Problems in Ordinary Differential Equations**, Prentice-Hall, Englewood Cliffs, NJ, 1971

[Goldstein 83] Goldstein, Herbert, **Classical Mechanics**, 2nd edition, Addison-Wesley, Reading, Massachusetts, 1983.

[Golub and Van Loan 83] Golub, G., and Van Loan, C., **Matrix Computations**, Johns Hopkins University Press, Baltimore, 1983.

[Isaacs and Cohen 87] Isaacs, Paul M. and Michael F. Cohen, *Controlling Dynamic Simulation with Kinematic Constraints, Behavior Functions, and Inverse Dynamics,* Proc. SIGGRAPH 1987, pp. 215-224

[Lengyel 87] Lengyel, Jed, *Dynamic Assembly and Behavioral Simulation of the Flagellar Axoneme,* in **Caltech SURF Reports**, 1987

[Lien and Kajiya 84] Lien, Sheue-ling, and James T. Kajiya, *A symbolic method for calculating the integral properties of arbitrary nonconvex polyhedra,* IEEE Computer Graphics and Applications, Vol. 4 No. 10, Oct. 1984, pp. 35–41.

[Misner, Thorne and Wheeler 73] Misner, Charles W., Kip S. Thorne, and John Archibald Wheeler, **Gravitation**, W.H. Freeman and Co., San Francisco, 1973.

[Press et. al. 88] Press, William H., Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, **Numerical Recipes in C/The Art of Scientific Computing**, Cambridge University Press, Cambridge, 1988.

[Platt and Barr 88] Platt, John, and Alan Barr, *Constraints on Flexible Objects,* Submitted to SIGGRAPH 1988.

[Ralston and Rabinowitz 78] Ralston, Anthony, and Philip Rabinowitz, *A First Course in Numerical Analysis*, McGraw-Hill, New York, 1978.

[Shoemake 85] Shoemake, Ken, *Animating Rotation with Quaternion Curves*, Computer Graphics, Vol. 19 No. 3, July 1985. pp. 245-254.

[Terzopoulos, Platt, Fleischer, and Barr 87] Terzopoulos, Demetri, John Platt, Alan Barr, and Kurt Fleischer *Elastically Deformable Models*, Proc. SIGGRAPH, 1987, pp. 205-214.

[Witkin, Fleischer, and Barr 87] Witkin, Andrew, Kurt Fleischer, and Alan Barr, *Energy Constraints on Parametrized Models,* Proc. SIGGRAPH 1987, pp. 225-232

[Wilhelms and Barsky 85] Wilhelms, Jane, and Brian Barsky *Using Dynamic Analysis To Animate Articulated Bodies Such As Humans and Robots*, Graphics Interface, 1985.

Appendices:

A Simulating Newtonian Mechanics

Fig. 16 summarizes the equations of motion of a rigid body. A full discussion of rigid-body dynamics can be found in [Fox 67,Goldstein 83].

A.1 Notes On The Equations Of Motion

- *The Orientation Matrix* **R***:* **R** transforms tensors from body-coordinates to world coordinates (see Fig. 17). As we numerically integrate **R**, numerical noise tends to cause **R** to drift away from a pure rotation, yielding noticeable skewing. This can be alleviated by using a feedback technique, as in [Barr 83]. Alternatively, we can represent the orientation as a quaternion **Q** (see [Shoemake 85] for an introduction to quaternions). The equation of motion for **Q** is (see [Misner, Thorne and Wheeler 73]):

$$\frac{d}{dt}\mathbf{Q} = \frac{1}{2}\vec{\omega}\mathbf{Q}$$

  We then define **R** to be an auxiliary variable, which is computed from **Q** as discussed in [Shoemake 85].

- *Rotational Inertia Tensor* **I***:* **I** determines the rotational behavior of a body.[4] For a rigid body, $\mathbf{I}_{body}$ is constant. Note also that in Fig. 16

[4] A discussion of the characterstics of **I** is beyond the scope of this paper; see [Fox 67,Goldstein 83]. [Lien and Kajiya 84] gives an algorithm to compute **I** for arbitrary nonconvex polyhedra.

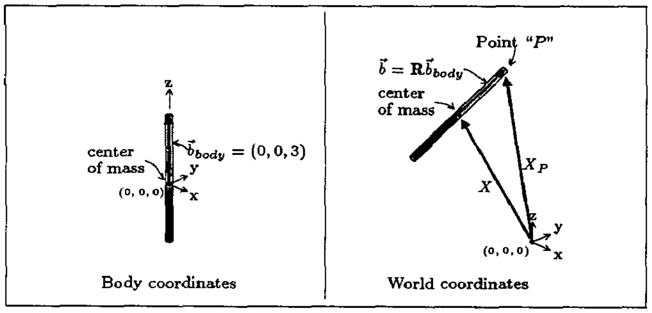

Figure 17: A rigid body. Orientation matrix R transforms vectors from body coordinates to world coordinates.

we use $\mathbf{I}^{-1}$ rather than $\mathbf{I}$. $\mathbf{I}^{-1}_{body}$ can be precomputed for each body; we convert to world coordinates using R:

$$\mathbf{I}^{-1} = \mathbf{R}\mathbf{I}^{-1}_{body}\mathbf{R}^{\mathsf{T}}$$

- *The Net Force $\vec{F}$ and Torque $\vec{T}$:* Euler's principle of superposition allows us to combine the forces applied to a body into an equivalent net force applied at the center of mass. Each force applied at a radius $\vec{b}$ from the center of mass contributes $\vec{b} \times \vec{F}$ to the net torque on the body; we can also have "pure torques" acting on the body. The net force and torque are thus:

$$\begin{aligned} \vec{F} &= \sum_{\text{forces } i} \vec{F}_i \\ \vec{T} &= \sum_{\text{forces } i} (\vec{b}_i \times \vec{F}_i) + \sum_{\text{torques } j} \vec{T}_j \end{aligned}$$

A.2 Canonical O.D.E. Notation

For brevity, we express the collection of equations in Fig. 16 as an ordinary differential equation (O.D.E.), in canonical form. For a single body, which we label A, we have:

$$\frac{d}{dt}Y^{\mathsf{A}} = f(Y^{\mathsf{A}}, \vec{F}^{\mathsf{A}}, \vec{T}^{\mathsf{A}}) \tag{8}$$

where we define

$$\begin{aligned} &Y^{\mathsf{A}} = \{\vec{x}^{\mathsf{A}}, \mathbf{R}^{\mathsf{A}}\vec{p}^{\mathsf{A}}, \vec{L}^{\mathsf{A}}\} && \text{—State of body A} \\ &\vec{F}^{\mathsf{A}} && \text{—Net force on A} \\ &\vec{T}^{\mathsf{A}} && \text{—Net torque on A} \end{aligned}$$

For a model consisting of a collection of bodies, we have:

$$\frac{d}{dt}\mathcal{Y} = f(\mathcal{Y}, \mathcal{F}, \mathcal{T}) \tag{9}$$

where we define

$$\begin{aligned} \mathcal{Y} &= \{Y^{\mathsf{A}}, Y^{\mathsf{B}}, \ldots\} && \text{—State of the model} \\ \mathcal{F} &= \{\vec{F}^{\mathsf{A}}, \vec{F}^{\mathsf{B}}, \ldots\} && \text{—Forces in the model} \\ \mathcal{T} &= \{\vec{T}^{\mathsf{A}}, \vec{T}^{\mathsf{B}}, \ldots\} && \text{—Torques in the model} \end{aligned}$$

Numerical methods for solving first-order O.D.E.'s are well known (see [Press et. al. 88,Boyce and Deprima 77,Ralston and Rabinowitz 78]). When the equations are not stiff (stiff differential equations occur when there are multiple widely varying time constants for the solutions), an adaptive Adams predictor corrector is suitable; otherwise we recommend a method such as Gear's Method ([Gear 71]).

B Mathematical Details

B.1 Dual of a vector

The *dual* of a vector $\vec{b}$ is the antisymmetric matrix $\mathbf{b}^*$:

$$\text{With } \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}, \text{ define } \mathbf{b}^* = \begin{bmatrix} 0 & b_3 & -b_2 \\ -b_3 & 0 & b_1 \\ b_2 & -b_1 & 0 \end{bmatrix}$$

For any vectors $\vec{b}$ and $\vec{a}$, we have the following identities:

$$\begin{aligned} \mathbf{b}^*\vec{a} &\equiv \vec{b} \times \vec{a} \\ \mathbf{b}^{*\mathsf{T}}\vec{a} &\equiv \vec{a} \times \vec{b} \\ \mathbf{b}^{*\mathsf{T}}\vec{b} &\equiv 0 \end{aligned}$$

B.2 Behavior of a Point

Consider a point "P" which is fixed relative to a rigid body (Fig. 17). We define $\vec{b}_{body}$ to be the vector from the center-of-mass of the body to P, expressed in the body's home coordinates; since the point is fixed, $\vec{b}_{body}$ is constant. We would like to derive expressions for the position, velocity, and acceleration of P.

We will need to know the derivative of $\mathbf{I}^{-1}$. Remember that since the body is rigid, $\mathbf{I}^{-1}_{body}$ is constant:

$$\begin{aligned} \mathbf{I}^{-1} &= \mathbf{R}\mathbf{I}^{-1}_{body}\mathbf{R}^{\mathsf{T}} \\ \tfrac{d}{dt}\mathbf{I}^{-1} &= (\tfrac{d}{dt}\mathbf{R})\mathbf{I}^{-1}_{body}\mathbf{R}^{\mathsf{T}} + \mathbf{R}\mathbf{I}^{-1}_{body}(\tfrac{d}{dt}\mathbf{R}^{\mathsf{T}}) \\ &= \omega^*\mathbf{R}\mathbf{I}^{-1}_{body}\mathbf{R}^{\mathsf{T}} + \mathbf{R}\mathbf{I}^{-1}_{body}\mathbf{R}^{\mathsf{T}}\omega^{*\mathsf{T}} \\ &= \omega^*\mathbf{I}^{-1} + \mathbf{I}^{-1}\omega^{*\mathsf{T}} \end{aligned} \tag{10}$$

We have substituted $\omega^*\mathbf{R}$ for $\frac{d}{dt}\mathbf{R}$, according to the equations of motion (Appendix A).

We will also need the derivative of $\vec{\omega}$:

$$\begin{aligned} \vec{\omega} &= \mathbf{I}^{-1}\vec{L} \\ \tfrac{d}{dt}\vec{\omega} &= (\tfrac{d}{dt}\mathbf{I}^{-1})\vec{L} + \mathbf{I}^{-1}(\tfrac{d}{dt}\vec{L}) \\ &= (\tfrac{d}{dt}\mathbf{I}^{-1})\vec{L} + \mathbf{I}^{-1}\vec{T} \\ &= \omega^*\mathbf{I}^{-1}\vec{L} + \mathbf{I}^{-1}\omega^{*\mathsf{T}}\vec{L} + \mathbf{I}^{-1}\vec{T} \\ &= \omega^*\vec{\omega} + \mathbf{I}^{-1}(\omega^{*\mathsf{T}}\vec{L} + \vec{T}) \\ &= \mathbf{I}^{-1}(\vec{L} \times \vec{\omega} + \vec{T}) \end{aligned} \tag{11}$$

We have substituted $\vec{T}$ for $\frac{d}{dt}\vec{L}$ according to the equations of motion.

We can now differentiate $\vec{b}$:

$$\begin{aligned} \vec{b} &= \mathbf{R}\vec{b}_{body}, \text{ and} \\ \tfrac{d}{dt}\vec{b} &= \tfrac{d}{dt}(\mathbf{R}\vec{b}_{body}) \\ &= (\tfrac{d}{dt}\mathbf{R})\vec{b}_{body} \\ &= \omega^*\mathbf{R}\vec{b}_{body} \\ &= \omega^*\vec{b} \\ &= \vec{\omega} \times \vec{b} \\ \tfrac{d^2}{dt^2}\vec{b} &= (\tfrac{d}{dt}\vec{\omega}) \times \vec{b} + \vec{\omega} \times (\tfrac{d}{dt}\vec{b}) \\ &= (\tfrac{d}{dt}\vec{\omega}) \times \vec{b} + \vec{\omega} \times (\vec{\omega} \times \vec{b}) \\ &= (\mathbf{I}^{-1}(\vec{L} \times \vec{\omega} + \vec{T})) \times \vec{b} + \vec{\omega} \times (\vec{\omega} \times \vec{b}) \\ &= (\mathbf{b}^{*\mathsf{T}}\mathbf{I}^{-1})\vec{T} + (\mathbf{b}^{*\mathsf{T}}\mathbf{I}^{-1}(\vec{L} \times \vec{\omega}) + \vec{\omega} \times (\vec{\omega} \times \vec{b})) \\ &= \mathbf{H}\vec{T} + \vec{\beta} \\ \text{where we define} & \\ \mathbf{H} &= \mathbf{b}^{*\mathsf{T}}\mathbf{I}^{-1} \\ \vec{\beta} &= \mathbf{b}^{*\mathsf{T}}\mathbf{I}^{-1}(\vec{L} \times \vec{\omega}) + \vec{\omega} \times (\vec{\omega} \times \vec{b}) \end{aligned} \tag{12}$$

We have again substituted $\omega^*\mathbf{R}$ for $\frac{d}{dt}\mathbf{R}$.

Finally, we can express the position, velocity, and acceleration of point P in terms of the state of the body and the net force and torque on the body:

$$\begin{aligned} \vec{X}_P &= \vec{X} + \vec{b} \\ \vec{v}_P &= \tfrac{d}{dt}\vec{X}_P \\ &= \tfrac{d}{dt}\vec{X} + \tfrac{d}{dt}\vec{b} \\ &= \vec{v} + \vec{\omega} \times \vec{b} \\ \vec{a}_P &= \tfrac{d^2}{dt^2}\vec{X}_P \\ &= \tfrac{d^2}{dt^2}\vec{X} + \tfrac{d^2}{dt^2}\vec{b} \\ &= \tfrac{1}{m}\vec{F} + \mathbf{H}\vec{T} + \vec{\beta} \\ &= \mathbf{G}\vec{F} + \mathbf{H}\vec{T} + \vec{\beta} \\ \text{where we define} & \\ \mathbf{G} &= \tfrac{1}{m} \end{aligned} \tag{13}$$

C Constraint Derivations

For each type of constraint, we must derive expressions for the various quantities defined in Fig. 11. The steps we follow are:

1. Choose a simple "deviation" measure $\vec{D}$. $\vec{D}$ is a function of the positions ($\vec{X}$) and orientations (R) of the constrained bodies, and may optionally depend on t.
2. Differentiate $\vec{D}$, to derive $\vec{D}^{(1)}(\mathcal{Y}, t)$; Substitute $\vec{v}$ and $\omega^* R$ for the $\frac{d}{dt}\vec{X}$ and $\frac{d}{dt}R$ terms which will arise (see Fig. 11).
3. Differentiate again, to derive $\vec{D}^{(2)}(\mathcal{Y}, \vec{F}, \vec{T}, t)$. Replace $\frac{d}{dt}\vec{p}$ and $\frac{d}{dt}\vec{L}$ terms with $\vec{F}$ and $\vec{T}$, thus giving rise to the linear dependence of $\vec{D}^{(2)}$ on the forces and torques. Define the $d \times 3$ matrices Γ, A, and the d-vector $\vec{\beta}$
4. Choose where to apply the constraint forces needed to meet the constraint. Most often, we apply a vector force to a fixed location of the constrained body; in this case, we have $f = 3$ degrees of freedom.
5. Use steps 2 and 3 to derive G and H for each body. These convert the f values in the "constraint force" $\vec{F}_c$ into the actual forces and torques on the bodies.

Often, some of the quantities Γ, A, G, and H, which are nominally matrices, turn out to be scalar. Scalars can be handled as a special case in the implementation, or scaled identity matrices can be used.

We give examples of the constraint derivations for the constraints illustrated in Sec. 2.2.

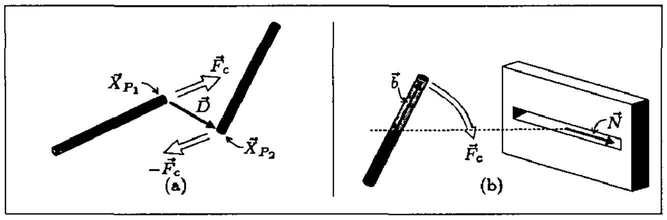

Figure 18: (a) "Point-to-point" constraint. We apply equal-and-opposites to the two constrained points, to cause the deviation function $\vec{D}(\mathcal{Y}) = \vec{X}_{P_2}(\mathcal{Y}) - \vec{X}_{P_1}(\mathcal{Y})$ to go to 0. (b) "Orientation" constraint. We rotate the body to cause the deviation function $\vec{D}(\mathcal{Y}) = \vec{b} \cdot \vec{N} - 1$ to go to 0. The constraint torque is given directly by $\vec{F}_c$; there is no force due to the constraint.

C.1 "Point-To-Path" Constraint

This constraint is met at a time t if the constrained point "P" is on the path, at $\vec{X}^{\mathsf{path}}(t)$; it is the same as the "point-to-nail" constraint (Example 1 in Sec. 4) but with a nail that moves. Thus several terms have a dependency on $\vec{X}^{\mathsf{path}}$:

$$\begin{aligned}
\vec{D}(\mathcal{Y},t) &= \vec{X}_P(\mathcal{Y}) - \vec{X}^{\mathsf{path}}(t)\\
\vec{D}^{(1)}(\mathcal{Y},t) &= \vec{v}_P(\mathcal{Y}) - \tfrac{d}{dt}\vec{X}^{\mathsf{path}}(t)\\
\vec{D}^{(2)}(\mathcal{Y},\vec{F},\vec{T},t) &= \vec{a}_P(\mathcal{Y},\vec{F},\vec{T}) - \tfrac{d^2}{dt^2}\vec{X}^{\mathsf{path}}(t)\\
\vec{\beta} &= \mathrm{b}^{*\mathsf{T}}\mathrm{I}^{-1}(\vec{L}\times\vec{\omega}) + \vec{\omega}\times(\vec{\omega}\times\vec{b})\\
&\quad - \tfrac{d^2}{dt^2}\vec{X}^{\mathsf{path}}(t)
\end{aligned}$$

C.2 "Point-To-Point" Constraint

This constraint is met if the two constrained points "P_1" and "P_2" are at the same location (Fig. 18a). We thus define $\vec{D}$ to be the vector separating the two points. The derivation proceeds analogously to that of the "point-to-nail" constraint:

$$\begin{aligned}
\vec{D}(\mathcal{Y}) &= \vec{X}_{P_2}(\mathcal{Y}) - \vec{X}_{P_1}(\mathcal{Y})\\
\vec{D}^{(1)}(\mathcal{Y}) &= \vec{v}_{P_2}(\mathcal{Y}) - \vec{v}_{P_1}(\mathcal{Y})\\
\vec{D}^{(2)}(\mathcal{Y},\vec{F},\vec{T}) &= \vec{a}_{P_2}(\mathcal{Y}) - \vec{a}P_1(\mathcal{Y})\\
\mathbf{\Gamma}^1 &= \tfrac{1}{m_1}\mathbf{1}\\
\mathbf{\Lambda}^1 &= \mathbf{b}_1^{*\mathsf{T}}\mathbf{I}_1^{-1}\\
\mathbf{\Gamma}^2 &= -\tfrac{1}{m_2}\mathbf{1}\\
\mathbf{\Lambda}^2 &= -\mathrm{b}_2^{*\mathsf{T}}\mathbf{I}_2^{-1}
\end{aligned}$$

To be in keeping with Newton's third law, the two bodies must exert equal and opposite forces on each other. We apply an arbitrary force, $\vec{F}_c$, to one of the constrained points, and the negation of that force, $-\vec{F}_c$, to the other. We thus have $f = 3$, and define

$$\begin{aligned}
\mathbf{G}^1 &= 1; &\quad \mathbf{H}^1 &= \mathrm{b}_1^*\\
\mathbf{G}^2 &= -1; &\quad \mathbf{H}^2 &= -\mathrm{b}_2^*
\end{aligned}$$

C.3 "Orientation" Constraint

This constraint is met if a specified unit vector $\vec{b}$ fixed in the body lines up with a unit vector $\vec{N}$ fixed in the world (see Fig. 18b). We could define D to be the angle between the vectors; it easier, however, if we define D to be 0 when the cosine of the angle (i.e. the dot-product of the vectors) is 1. Thus we have $d = 1$, and:

$$\begin{aligned}
D(\mathcal{Y}) &= \vec{b}(Y)\cdot\vec{N} - 1\\
D^{(1)}(\mathcal{Y}) &= (\tfrac{d}{dt}\vec{b}(\mathcal{Y}))\cdot\vec{N}\\
D^{(2)}(\mathcal{Y},\vec{T}) &= (\tfrac{d^2}{dt^2}\vec{b}(\mathcal{Y}))\cdot\vec{N}\\
&= ((\mathbf{b}^{*\mathsf{T}}\mathbf{I}^{-1})\vec{T})\cdot\vec{N}\\
&\quad +(\mathrm{b}^{*\mathsf{T}}\mathrm{I}^{-1}(\vec{L}\times\vec{\omega}) + \vec{\omega}\times(\vec{\omega}\times\vec{b}))\cdot\vec{N}\\
\mathbf{\Gamma} &= 0\\
\mathbf{\Lambda} &= \vec{N}^{\mathsf{T}}\mathrm{b}^{*\mathsf{T}}\mathbf{I}^{-1}\\
\vec{\beta} &= (\mathbf{b}^{*\mathsf{T}}\mathbf{I}^{-1}(\vec{L}\times\vec{\omega}) + \vec{\omega}\times(\vec{\omega}\times\vec{b}))\cdot\vec{N}
\end{aligned}$$

Notice that $\mathbf{\Lambda}$ is a 1×3 matrix, and $\vec{\beta}$ is a scalar.

We apply an arbitrary pure torque, $\vec{F}_c$, to the body, and no force. We therefore have $f = 3$, and

$$\mathbf{G} = 0; \quad \mathbf{H} = \mathbf{1}$$

Notice that this constraint is "non-square" - we are applying 3 degrees of freedom to affect a scalar constraint "deviation."

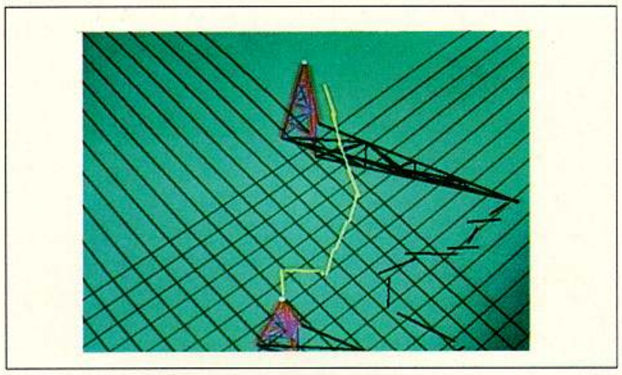

Figure 19: Linking Chain between Two Towers. The chain swings naturally after assembly.

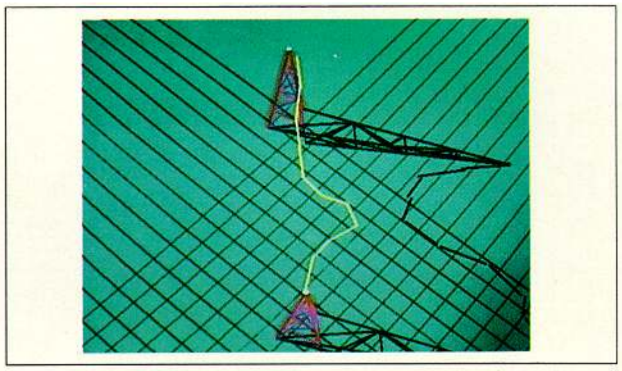

Figure 20: Linking Chain between Two Towers, continued

D Acknowledgements

Thanks to: Jed Lengyel, for being a "guinea pig" user of the modeling system, and for putting together the video system that was used in making animations; John Snyder, for the rendering of the animations; John Platt, for discussions and for numerical software; and Brian Von Herzen, for the motivation and modeling for the Harwood space-station self-assembly animations and for the polygonal playing card models.

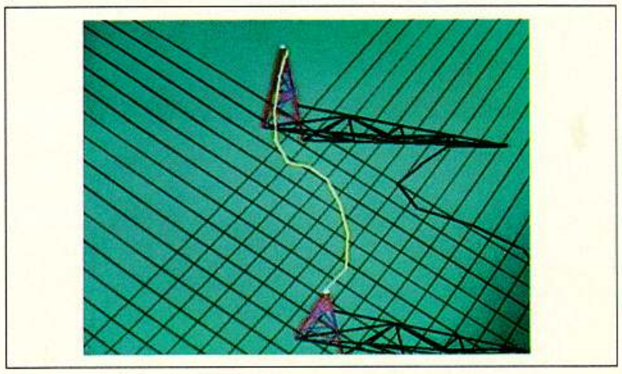

Figure 21: Linking Chain between Two Towers, continued

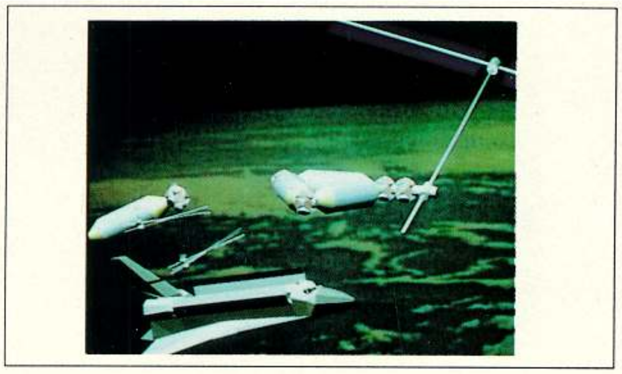

Figure 22: Space Station Assembly. The modules are assembled via "orientation", "point-to-point", and "point-to-nail" constraints. The constraint forces determine the strengths of the rocket thrusts.

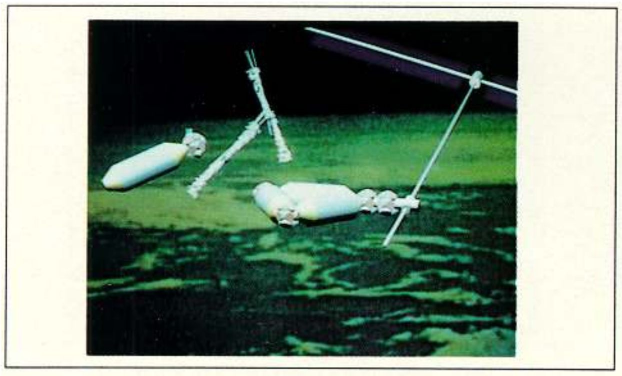

Figure 23: Space Station Assembly, continued.

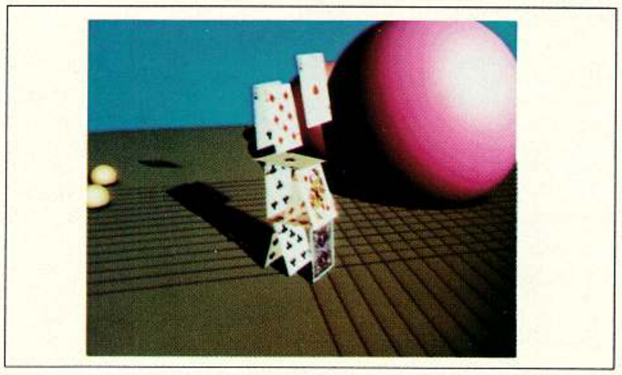

Figure 24: Cardhouse Assembly. We use "point-to-point," and "pt-to-plane" constraints to assemble a cardhouse.

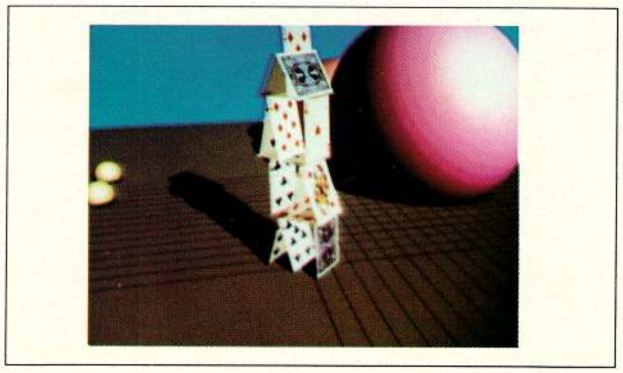

Figure 25: Cardhouse Assembly, completed.

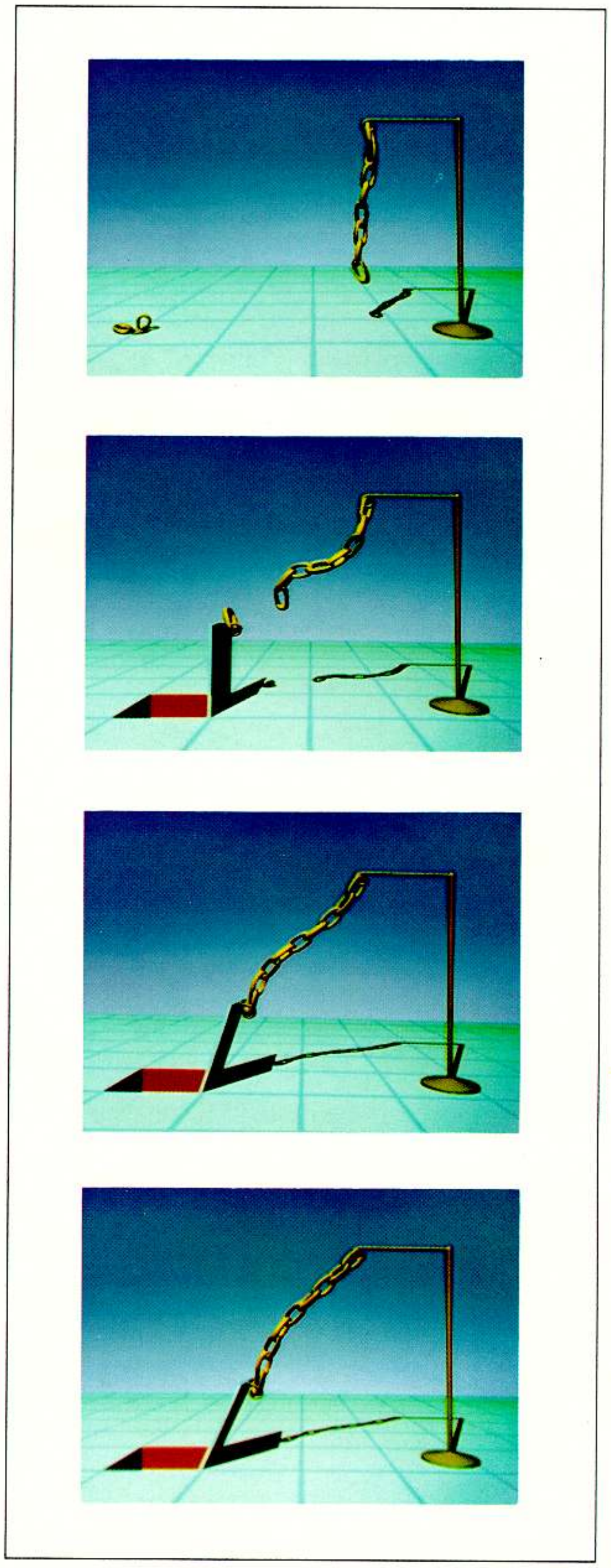

Figure 26: Pandora's Chain. The chain links are instructed to connect togetherand hook to a trap door. Torsion springs keep the links roughly perpendicular to each other. Gravity and viscous damping are applied to all bodies. The chains and trap door swing naturally once they are assembled.