# Impulse-based Dynamic Simulation

Brian Mirtich, *University of California, Berkeley, CA, USA*
John Canny, *University of California, Berkeley, CA, USA*

*We introduce a promising new approach to dynamic simulation called* impulse-based *simulation. The distinguishing feature of this method is the unification of all types of contact (colliding, rolling, sliding, and resting) under a single framework; non-colliding contacts are simulated as a series of tiny* microcollisions. *The approach is simpler and more robust than previous constraint-based methods. Simulation results agree with physical experiments, and the method is fast enough to make real time simulation possible. In the course of describing impulse-based simulation, we present an efficient collision detection scheduling scheme and a fully general treatment of frictional collisions. We conclude with some of the results generated by our simulator.*

## 1 Introduction

The goal of dynamic simulation is to make the computer into a tool which mimics the physical world; the applications of such a tool are countless. Electronic prototyping allows the engineer to interactively test and modify designs while they are still on the drawing board, before an actual prototype is ever constructed [8]. Experiments which are too costly or impractical to perform can be simulated, such as failure mode tests of bridges or dams. Even experiments which are performed today, such as automobile crash tests, can be done at much lower cost and under more varied conditions. Finally, dynamic simulation is an integral part of the expanding area of virtual reality. In everything from architectural walk-through programs to flight simulators, virtual environments need to behave as closely as possible to the actual physical world we inhabit.

The foremost requirement of dynamic simulation is physical accuracy. The goal of a simulation system is not simply to produce an animation sequence which "looks right" to a human; the sequence must *be* right. The simulation is to take the place of a physical model, and hence its utility is directly related to how well it mimics this physical system. Assumptions such as frictionless collisions may be allowable for generating realistic looking graphics, but they have no place in a system designed to model reality.

The second major requirement is computational efficiency. Clearly in virtual reality applications, the simulation must run in real time. Furthermore, in engineering applications it is most beneficial when the user can make changes to a design, and see the results at fully interactive speeds. If the designer must wait hours or even days to analyze the behavior of a system, the electronic prototype loses its great advantage over an actual physical prototype.

This paper discusses a new approach to dynamic simulation called impulse-based simulation. We have focused on the twin goals of physical accuracy and computational efficiency. Our simulator can accurately model complex dynamic systems in real time. The organization of this paper is as follows. Section 2 gives an overview of the impulse-based method for dynamic simulation, highlighting its differences from and advantages over more traditional constraint-based methods. Section 3 describes collision check scheduling, and how this standard bottleneck in dynamic simulation can be streamlined. Section 4 discusses our method of resolving collisions between bodies. We treat collisions in a fully general manner, accounting for friction as well as non-perfectly elastic behavior. Correctly computing collision impulses is critical for achieving physically accurate simulations. Finally, section 5 describes some of the simulations we have performed with our simulator, illustrating the speed and accuracy of the approach, and mentions some future work.

### 1.1 Related work

Moore and Wilhelms give one of the earliest treatments of two fundamental problems in dynamic simulation: collision detection and collision response [13]. Hahn also pioneered dynamic simulation, modeling sliding and rolling contacts using impact equations [7]. His work is the precursor of our method, although we extend the applicability of impulse dynamics to resting contacts, and give a more unified treatment of multiple objects in contact. Furthermore, these early approaches all suffered from inefficient collision detection and unrealistic assumptions concerning impact dynamics (e.g. infinite friction at the contact point). Our method combines fast collision detection with a fully general treatment of frictional collisions. Cremer and Stewart describe *Newton* [6], probably the most advanced general-purpose dynamic simulator in use today. Newton's forte is the formulation and simulation of constraint-based dynamics for linked rigid bodies. It has been used to simulate a high degree of freedom walking robot [16], although the contact modeling is fairly simplistic. Baraff has published a great deal on simulation of bodies in contact [1, 2]. His work focuses on the resolution of forces when bodies are in resting (non-colliding) contact. His earlier work is for frictionless collisions, and he later showed that computing contact forces in the presence of friction is NP-hard [3]. A summary of his work in this area can be found in [4]. There are few full treatments of frictional collisions. Routh [15] is still considered the authority on this subject, and is the source cited by most mechanics texts which address it. More recently, Keller gives a slightly different treatment of frictional collisions [9]; our approach is quite similar to his. Wang and Mason have studied impact dynamics for robotic applications; their approach is based on Routh's, but deals only with the two-dimensional case [17]. Finally, a number of researchers have investigated several problems and paradigms for dynamic simulation and physical-based modeling. We refer the reader to [5, 18, 19].

## 2 Constraint-based versus impulse-based simulation

One of the most difficult aspects of dynamic simulation is dealing with the interactions between bodies in contact. Most of the work which has been done in this area falls into the category of constraint-based methods [5, 18, 6, 4]. An example will illustrate the approach. Consider a ball rolling along a table top. The normal force which the table exerts on the ball is a constraint force that does not do work on the ball, but only enforces a non-penetration constraint. In a constraint-based system, this force is not modeled explicitly but is instead accounted for by a constraint on the configuration of the ball (in this case, the ball's $z$-coordinate is held constant). The problem with this method is that as a dynamic system evolves, the constraints may change many times, e.g. the ball may roll off the table, may hit an object on the table, etc. Determining the correct equations of motion for the ball means keeping track of these changing constraints, which can become complicated. Moreover, it is not even clear what type of constraint should be applied; there exist at least two models for rolling contact which in some cases predict different behaviors [10]. Finally, impacts are not easily incorporated into the constraint model, as they generally give rise to impulses, not constraint forces present over some interval. These collision impulses must be handled separately, as in [1].

In contrast, our system is based on a method we call impulse-based dynamic simulation. Unlike constraint-based methods, no constraints are imposed on the configurations of the moving objects; when the objects are not colliding, they are in ballistic trajectories. The key advantage of the impulse-based method is the unification of all types of contact under a single model. The model used for collisions between objects can also be used for continuous contact situations in which one object is resting, sliding, or rolling on another object. Consider for example a block resting on a table. Under impulse-based simulation, the block is actually experiencing many rapid, tiny collisions with the table, each of which can be resolved as any other collision. During this small, vibratory motion, different corners of the block will collide and rebound with the table. We call these small, frequent collisions between objects in continuous contact *microcollisions*.

Now consider the case of a ball bouncing along a terrain as shown in figure 1. Under constraint-based simulation, the constraints change as the ball begins traveling up the ramp, leaves the ramp, and settles into a roll along the ground. All these occurrences must be detected and processed. Impulse-based simulation avoids having to worry about such transitions. In fact,

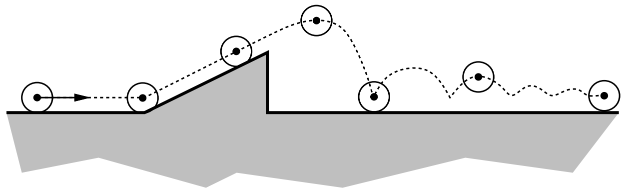

**Figure 1:** *A nightmare for constraint-based simulation.*

it is a more physically sound treatment since it does not establish an artificial boundary between, for example, bouncing and rolling, but instead handles the entire continuum of contact between these states.

Two obvious questions concerning impulse-based simulation are: (1) Does it work, i.e. does it result in physically accurate simulations?, and (2) Is it fast enough for simulation purposes? We defer more thorough answers to these questions to section 5, but for now state the following: impulse-based dynamic simulation *does* produce physically accurate results, even for cases which have been problematic for previous simulators. The microcollision contact model produces the correct macroscopic behavior; simulations agree with physical experiments. What is even more surprising is that the approach is extremely fast. Because of the simplicity of the model, the large number of collisions which must be resolved is not prohibitive. In fact, real time simulation is possible.

In summary, there are several advantages to impulse-based dynamic simulation:

1. All types of contact (colliding, rolling, sliding, sticking) are unified under a single approach; it is not necessary to classify the contact types and deal with them separately or in different manners.
2. It is not necessary to maintain a set of constraints, nor to determine when this set changes.
3. Simulating various types of contact with microcollisions gives rise to the correct macroscopic physical behavior, as verified experimentally.
4. The method is conceptually simpler and more robust then constraint-based dynamic simulation.
5. The method is fast. Despite the large numbers of collisions, real time simulation is possible.

## 3 Collision detection

Impulse-based dynamic simulation is obviously quite collision intensive—consider for example the high number of microcollisions which occur as a ball rolls across a table top. Furthermore, the naive approach to collision detection is inherently quadratic in the number of moving objects. Hahn and others have found collision detection to be the bottleneck in dynamic simulation [7], and certainly much effort should be put into streamlining it.

The heart of our collision detection system is the Lin-Canny closest features algorithm [12]. This algorithm returns the closest features (vertices, edges, or faces) between a pair of convex polyhedra. It is especially suited to applications like dynamic simulation, where a sequence of queries are made as objects move continuously in space. In these cases, geometric coherence can be exploited to achieve a constant expected query time. Non-convex objects can be handled by decomposition into convex pieces, and there is even an extension of the algorithm to curved surfaces [11]. Once the closest features are known, computing the distance between the two objects is a simple operation. The collision detection system reports a possible collision when this distance falls below some epsilon, $\varepsilon_c$. In our simulations using standard single-precision floating point arithmetic, $\varepsilon_c$ is about three to four orders of magnitude smaller than the dimensions of the objects. For bowling simulations using a 60' alley and 18" inch pins, $\varepsilon_c$ is one millimeter.

The basic simulation loop comprises three steps: dynamic state evolution, collision detection, and collision resolution. A naive approach for collision detection would test for possible collisions between all pairs of objects after each dynamic state evolution step. For a simulation involving $n$ moving objects, this gives rise to an $O(n^2)$ collision detection step, despite the efficient constant time distance query. A second problem is how to choose the length of the integration time step, i.e. for how long should the dynamic state be evolved before the collision detection step is executed? Often, this integration step size is chosen to be "small enough so that no collisions are missed" (such as in [6]), but this is ad hoc and forces a small step size even when one is not necessary. We employ a strategy which reduces the number of collision checks from the naive approach, facilitates an adaptive step size for dynamic state evolution, and insures no collisions are missed.

### 3.1 Prioritizing collisions

The basic idea is to find a lower bound on the time of collision when two objects have not yet collided. Such a conservative bound is computed as a function of the distance between the closest points on the objects and the current dynamic state. It is also necessary to bound the linear and angular accelerations of the objects, since these also affect the minimum time to collision. By making the assumption that the objects will continue to travel in ballistic trajectories until impact, one can bound the linear acceleration of the center of mass to be $g$, the acceleration of gravity. Bounding the angular acceleration is a little trickier, but such a bound can be found as a function of the current angular velocity and the mass matrix of the object, again assuming a ballistic trajectory. We mention that the collision detection algorithm, which returns a soonest possible time of collision if the objects are not colliding, is in contrast to most other algorithms which simply compute a predicate indicating if the objects interpenetrate or not. For these latter algorithms, the exact time of collision is usually found by forward evolution and backtracking, using a binary search or other iterative method.

The information returned by the collision detection algorithm is stored in a heap; each heap item consists of a pair of objects and the soonest possible time at which these objects could collide. The heap is prioritized by collision time, so that at any moment, the object pair on the top of the heap is the nearest pair to collision. At each dynamic evolution step, the integration is performed up to the time of collision of the top item in the heap. At this point, collision detection is performed for this single pair of objects. If the objects have collided, the collision resolution step is performed, otherwise the time of impact is recomputed and the heap updated appropriately. When two objects do collide and experience collision impulses, the ballistic trajectory assumption is violated and all times of collision involving either of these objects must be updated in the heap. This strategy greatly reduces collision checks. If two objects are far apart or moving very slowly, collision checks between them will be very infrequent. As the objects approach one another, collision checks will increase as necessary.

### 3.2 Further reducing collision checks

Although the above strategy considerably lowers the asymptotic constant considerably, collision detection is still an $O(n^2)$ proposition. The problem is that collision checks between every pair of objects must still be performed at regular intervals, even if the pair of objects never come near one another. To alleviate this problem, some sort of spatial decomposition scheme must be built on top of the collision time priority queue. We now describe a method to eliminate these unneeded checks.

First, a basic time period $t_f$ is chosen over which the swept volumes of moving objects will be bounded. It is convenient to choose $t_f$ to be the frame period for the simulation, e.g. $t_f = \frac{1}{30}$ s. Let $t$ denote the current time. For an object $O$,a rectangular bounding volume $B_O$ with faces parallel to the global coordinate planes is computed. $B_O$ bounds the volume occupied by $O$ during the time interval $[t, t+t_f]$. Let $r_O$ be the "radius" of object $O$, that is the greatest distance of any point on $O$ from $O$'s center of mass. $B_O$ can be found by noting the position of $O$'s center of mass at the current time $t$, at the time $t + t_f$, and possibly at the apex of its parabolic trajectory, should this occur during the interval $[t, t + t_f]$. The box which bounds these two or three points is then grown by $r_O$ to give the final bounding volume, $B_O$ (see figure 2).

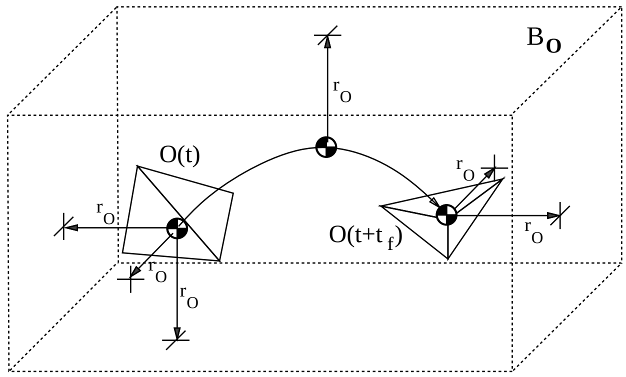


**Figure 2:** *The bounding box for an object O's swept volume.*

At the beginning of each frame, the bounding boxes for all objects are computed in linear time. The intersections between these $n$ boxes are then found; for cubical boxes, Overmars has given an algorithm for doing this in $O(n(1 + \log R))$ time, where $R$ is the ratio of largest to smallest box size [14]. If two bounding boxes do not intersect, then the corresponding objects will not collide during the next frame, and no distance or time of impact calculations need be performed for that pair of objects. If the boxes do intersect, these calculations are performed, and the upcoming collision is inserted into the heap. The simulation then proceeds as before, using an adaptive dynamic evolution step based on the pending time of collision at the top of the heap. Upon collision, the swept volume bounding boxes of the colliding objects are recomputed. These new bounding boxes are then intersected with the other bounding boxes, and object pairs are inserted or removed from the heap as appropriate. Note that distance and time of collision calculations are never performed for a pair of objects that never come near each other.

## 4 Computing collision impulses

When two bodies collide, an impulse $\mathbf{p}$ must be applied to one of the bodies to prevent interpenetration; an equal but opposite impulse $-\mathbf{p}$ is applied to the other. Once $\mathbf{p}$ and its point of application are known, it is a simple matter to compute the new linear center of mass velocity and the new angular velocity for each body. After updating these velocities, dynamic state evolution can continue, assuming ballistic trajectories for all moving objects. Thus, the central problem in collision resolution is to determine the collision impulse $\mathbf{p}$. Accurate computation of the impulses arising between colliding bodies or bodies in rolling or sliding contact is critical to the physical accuracy of the simulator.

### 4.1 Assumptions for collisions

For impulse-based simulation, it is not feasible to make gross simplifying assumptions such as frictionless contacts or perfectly elastic collisions. Our approach for analyzing general frictional impacts is similar to that of Routh [15], although we derive equations which are more amenable to numerical integration; Keller also gives an excellent treatment [9]. Before describing our method of computing $\mathbf{p}$, we state the assumptions.

**Assumption 1 (Infinitesimal collision time)** *The duration of a collision is negligible compared to the time over which simulated objects move appreciably. As a result, (1) the configurations of two colliding objects may be taken as constant during the entire collision, and (2) the effect of one object on the other can be described by an impulse, giving rise to instantaneous changes in the linear and angular velocities of the object.*

This is a common assumption made in dynamic simulation [9]. The second part simply means that, unlike ordinary forces which can only affect accelerations instantaneously, the collision impulses can instantaneously affect velocities. Such behavior is necessary if we are to prevent objects from interpenetrating once they come into contact. What assumption 1 does *not* imply is that the collision can be treated as a discrete event. Even though the positions of the bodies are constant during the collision, the velocities are not. Since collision forces are functions of these velocities, it is necessary to examine the dynamics during the collision. One way to think of this is that the collision is a single point on the time line of the simulation, but to determine the collision impulses which are generated, we must use a magnifying glass to "blow up" this point, and examine what happens inside the collision.

In reality no body is completely rigid. When two bodies collide, there is a period of deformation in which elastic energy is stored in the bodies, and a restitution phase during which the bodies return to their original shapes (if the collision is non-destructive), rebounding off each other as the stored energy is released (see figure 3). One could use finite element analysis to study the stresses and strains occurring during a collision, but such a method is certainly not reasonable for real time simulation—furthermore, it is overkill. A simple empirical rule captures the essential behavior which occurs during this compression-restitution sequence.

**Assumption 2 (Poisson's hypothesis)** *For a collision between two objects, let $p_{total}$ be the magnitude of the normal component of the impulse imparted by one object onto the other over the entire collision. Let $p_{mc}$ be the magnitude of the normal component of the impulse imparted by one object onto the other up to the point of maximum compression. Then*

$$p_{total} = (1 + e)p_{mc}$$

*where $e$ is a constant between zero and one, dependent on the objects' materials, and called the coefficient of restitution.*

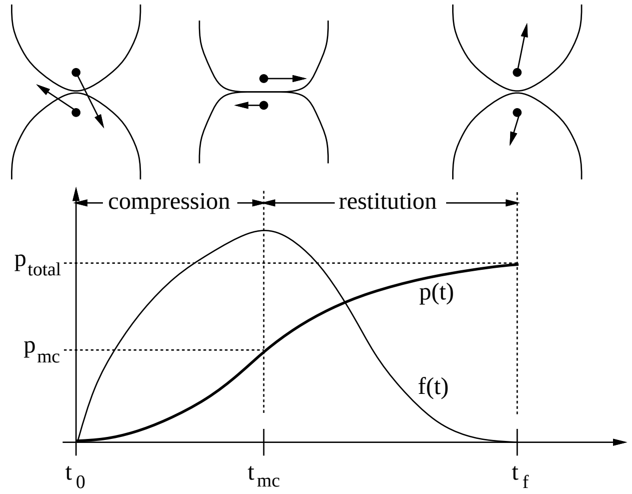


**Figure 3:** *A collision consists of a compression and a restitution phase. The boundary between these phases is the point of maximum compression, at which the relative contact velocity in the normal direction vanishes. $f(t)$ and $p(t) = \int f(t)dt$ are the force and total impulse delivered at time $t$ in the collision.*

In other words, the normal impulses delivered during the compression and restitution phases are in the ratio $1 : e$. If $e = 1$, the collision is totally elastic; no energy is lost. If $e = 0$, the collision is totally plastic; in general not all the energy is lost, but the objects do not separate after collision. Poisson's hypothesis is useful for resolving collisions because it relates final impulse values to maximum compression impulse values. The point of maximum compression is easier to characterize than the point at which the collision ends. It is simply the point at which the normal component relative contact velocity vanishes.

The tangential component of the impulse has not yet been mentioned. Analyzing frictionless collisions is easy since this component vanishes, but in the presence of friction this component cannot be ignored.

**Assumption 3 (Coulomb friction)** *At a particular point during a collision between bodies 1 and 2, let $\mathbf{u}$ be the contact velocity of body 1 relative to body 2, let $\mathbf{u}_t$ be the tangential component of $\mathbf{u}$, and let $\hat{\mathbf{u}}_t$ be a unit vector in the direction of $\mathbf{u}_t$. Let $\mathbf{f}_n$ and $\mathbf{f}_t$ be the normal and tangential (frictional) component of force exerted by body 2 on body 1, respectively. Then*

$$\mathbf{u}_t \neq \mathbf{0} \quad \Rightarrow \quad \mathbf{f}_t = -\mu||\mathbf{f}_n||\hat{\mathbf{u}}_t$$

$$\mathbf{u}_t = \mathbf{0} \quad \Rightarrow \quad ||\mathbf{f}_t|| \leq \mu||\mathbf{f}_n||$$

*where $\mu$ is the coefficient of friction.*

While the bodies are sliding relative to one another, the frictional force is exactly opposed to the direction of sliding. If the objects are sticking (i.e. the tangential component of relative velocity is zero), all that is known is that the total force lies in the friction cone.

## 4.2 Initial collision analysis

A collision takes place between body 1 and body 2, as shown in figure 4. We introduce the following notation (the subscript $i$ indicates the body number):

| | |
|---|---|
| $m_i$ | mass |
| $J_i$ | mass matrix |
| $\mathbf{v}_i$ | linear velocity of center of mass |
| $\mathbf{w}_i$ | angular velocity |
| $\mathbf{u}_i$ | absolute velocity of contact point |
| $\mathbf{r}_i$ | contact point position relative to c.o.m. |
| $\mathbf{p}$ | impulse imparted by body 2 on body 1 |

The vectors are expressed in the collision frame, which is some frame with $z$-axis perpendicular to the surfaces at the point of contact, and pointing from body 2 to body 1. When the colliding objects are polyhedra, the surfaces are not continuous, but reasonable surface normals can always be found. If one of the closest features is a face, the surface normal is the normal to this face; if the two closest features are edges, the normal is the vector mutually perpendicular to both edges; etc.

A possible collision is reported whenever the distance between two bodies falls below the collision epsilon, $\varepsilon_c$. The closest points on the objects are computed, and the collision frame is determined. A priori, the collision detection system only reports a *possible* collision, because the objects may be receding. If the closest points are moving away from each other, no collision impulse should be applied. (c.f. Baraff's constraint, $f\ddot{\chi}(\mathbf{f}) = 0$ [1].) The contact velocities are computed from

$$\mathbf{u}_i = \mathbf{v}_i + \mathbf{w}_i \times \mathbf{r}_i. \qquad (1)$$

The relative contact velocity $\mathbf{u}$ is simply $\mathbf{u}_1 - \mathbf{u}_2$. If the $z$-component of $\mathbf{u}$ is non-negative, the objects are not colliding, and no action is taken.

When $\mathbf{u}$ has negative $z$-component, a collision impulse must be applied to prevent interpenetration; it is

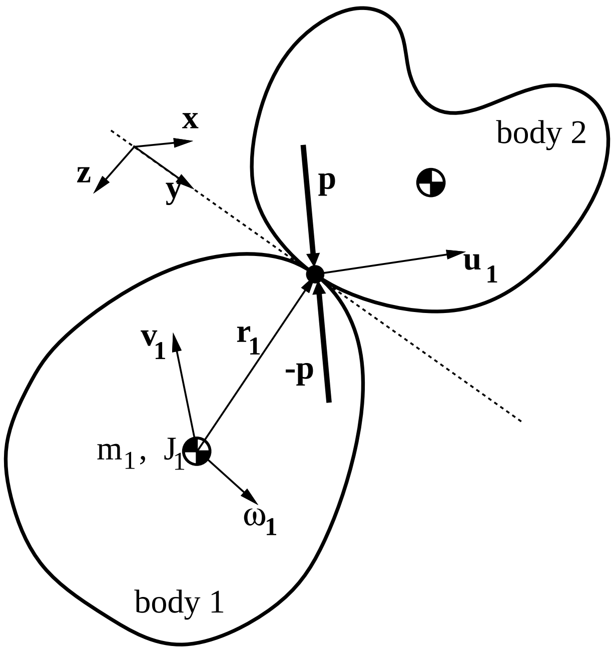


**Figure 4:** *Possible collision between two bodies.*

necessary to analyze the dynamics of the bodies during the collision to determine this impulse. We use $\gamma$ to denote the collision parameter, that is $\gamma$ is a variable which starts at zero, and continuously increases through the course of the collision until it reaches some final value, $\gamma_f$. All velocities are functions of $\gamma$, and $\mathbf{p}(\gamma)$ denotes the impulse delivered to body 1 up to point $\gamma$ in the collision. The goal is to determine $\mathbf{p}(\gamma_f)$, the total impulse delivered.

Initially, one might choose $\gamma$ to be "time since start of impact," but in fact this is not a very good choice. If the dynamics are studied with respect to time, the collision impulses are computed by integrating force. Unfortunately, the forces generated during a collision are not easily known; one can assume a Hooke's law behavior at the contact point, but this only leads to the question of how to choose the spring constants. Nonetheless, a variety of "penalty methods" do attempt to choose such spring constants. In addition to being chosen in a rather ad hoc way, these constants are very large, leading to stiff equations which are numerically intractable [18].

A way of avoiding all of these problems is to choose a different parameter for the collision, namely $\gamma = p_z$, the normal component of the impulse delivered to body 1. The scalar $p_z$ is zero at the moment the collision begins, and increases during the entire course of the collision, so it is a valid parameter. In our analysis, we will continue to use $\gamma$ to denote the collision parameter, for clarity. Consider the change in the contact point velocity of body 1 at a particular point during the collision

$$\Delta\mathbf{u}_1(\gamma) = \mathbf{u}_1(\gamma) - \mathbf{u}_1(0) = \Delta\mathbf{v}_1 + \Delta\mathbf{w}_1 \times \mathbf{r}_1. \quad (2)$$

Note that $\mathbf{r}_1$ is a constant throughout the collision, by assumption 1. Now $\Delta\mathbf{v}_1(\gamma)$ and $\Delta\mathbf{w}_1(\gamma)$ can easily be expressed in terms of the collision impulse:

$$\Delta\mathbf{v}_1(\gamma) = \frac{1}{m_1}\mathbf{p}(\gamma) \quad (3)$$

$$\Delta\mathbf{w}_1(\gamma) = J_1^{-1}[\mathbf{r}_1 \times \mathbf{p}(\gamma)]. \quad (4)$$

Substituting these into equation 2 and rearranging yields

$$\Delta\mathbf{u}_1(\gamma) = [\frac{1}{m_1}I - r_1^\times J_1^{-1} r_1^\times]\mathbf{p}(\gamma), \quad (5)$$

where $I$ is the $3 \times 3$ identity matrix, and $r_1^\times$ is the canonical $3 \times 3$ skew-symmetric matrix corresponding to $\mathbf{r}_1$. Performing a similar analysis for $\Delta\mathbf{u}_2$ ($-\mathbf{p}$ must be used instead of $\mathbf{p}$), and using $\mathbf{u} = \mathbf{u}_1 - \mathbf{u}_2$ to compute *relative* contact velocity, we obtain

$$\Delta\mathbf{u}(\gamma) = M\mathbf{p}(\gamma), \quad (6)$$

where $M$ is a $3 \times 3$ matrix dependent only upon the masses and mass matrices of the colliding bodies, and the location of the contact point relative to their centers of mass. By assumption 1, *$M$ is constant over the entire collision.* We can differentiate equation 6 with respect to the collision parameter $\gamma$ to obtain

$$\mathbf{u}'(\gamma) = M\mathbf{p}'(\gamma). \quad (7)$$

### 4.3 Sliding mode

While the tangential component of $\mathbf{u}$ is non-zero, the bodies are sliding relative to each other, and $\mathbf{p}'$ is completely constrained. Let $\theta(\gamma)$ be the relative direction of sliding during the collision, that is $\theta = \arg(u_x + iu_y)$.

**Lemma 1** *If the collision parameter $\gamma$ is chosen to be $p_z$, then while the bodies are sliding relative to one another,*

$$\mathbf{p}' = \begin{bmatrix} -\mu\cos\theta \\ -\mu\sin\theta \\ 1 \end{bmatrix}. \quad (8)$$

*Proof:* $p'_x = \frac{dp_x}{dp_z} = \frac{dp_x}{dt}\frac{dt}{dp_z} = f_x\frac{dt}{dp_z}$, where $\mathbf{f}$ is the instantaneous force exerted by body 2 on body 1. Under sliding conditions, assumption 3 implies $f_x = -(\mu\cos\theta)f_z = -(\mu\sin\theta)\frac{dp_z}{dt}$. Combining results gives $p'_x = -\mu\cos\theta$. The derivation for $p'_y$ is similar. Finally, $p'_z = \frac{dp_z}{dp_z} \equiv 1$. □

It is now clear why $p_z$ is a good choice for the collision parameter. By applying the results of lemma 1 to equation 7, with $\theta$ expressed in terms of $u_x$ and $u_y$, we obtain:

$$\begin{bmatrix} u'_x \\ u'_y \\ u'_z \end{bmatrix} = M \begin{bmatrix} -\mu\frac{u_x}{\sqrt{u_x^2+u_y^2}} \\ -\mu\frac{u_y}{\sqrt{u_x^2+u_y^2}} \\ 1 \end{bmatrix}. \tag{9}$$

This nonlinear differential equation for $\mathbf{u}$ is valid as long as the bodies are sliding relative to each other. By integrating the equation with respect to the collision parameter $\gamma$ (i.e. $p_z$), we can track $\mathbf{u}$ during the course of the collision. Because of the linear relationship between $\mathbf{p}$ and $\Delta\mathbf{u}$ (equation 6), we can also track $\mathbf{p}$ throughout the collision. Projections of the trajectories into the $u_x$-$u_y$ plane are shown in figure 5 for a particular matrix $M$; the crosses mark the initial sliding velocities. The trajectory plot is somewhat counterintuitive since for some initial conditions the sliding velocity *increases* although friction tends to resist sliding (for this plot $\mu = 0.55$). This is because the sliding velocity is also affected by the non-frictional (normal) component of the collision impulse, as shown in figure 6.

The basic impulse calculation algorithm proceeds as follows. After computing the initial $\mathbf{u}$ and verifying that $u_z$ is negative, we numerically integrate $\mathbf{u}$ using equation 9. During this integration, $u_z$ will increase[1]. When it reaches zero, the point of maximum compression has been attained. At this point, $p_z$ is the total normal impulse which has been applied. Multiplying this value by $(1+e)$ gives the terminating value for the collision parameter, $\gamma_f$. The integration then continues to this point, and $\mathbf{p}(\gamma_f)$ is the desired total collision impulse.

[1]Baraff and others have noted that it is possible to construct cases for which $u_z$ decreases as $p_z$ increases [3]. However, this situation seems to be extremely rare; it has not occurred in any of our simulations.

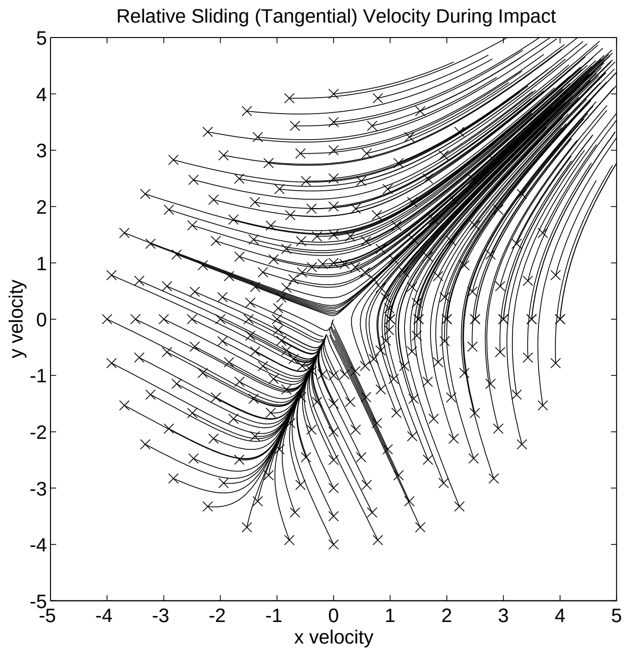


**Figure 5:** *Solution trajectories of equation 9 projected into the $u_x$-$u_y$ plane.*

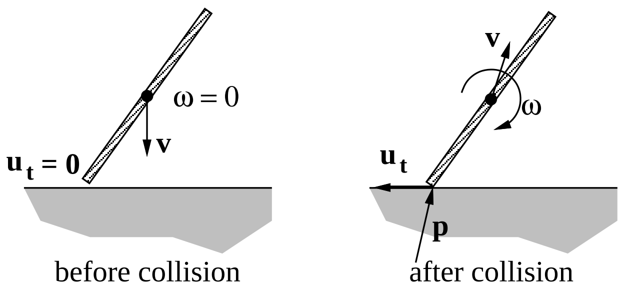


**Figure 6:** *A situation where the tangential relative contact velocity of the rod ($\mathbf{u}_t$) starts at zero and increases during the course of the collision, even though the frictional force resists this change in velocity.*

### 4.4 Sticking mode

Thus far we have not considered what happens if $u_x$ and $u_y$ both vanish, so that sliding motion ceases, and the objects are sticking. In this case, the direction of the frictional force is not known a priori, and lemma 1 no longer applies. The principle of virtual work implies that if the frictional force is strong enough to maintain the sticking condition, it will do so. To see if this is the case, we set $u'_x = u'_y = 0$ in equation 7, and solve for $\mathbf{p}'$. There is a unique solution for which $p'_z = 1$, say $\mathbf{p}' = (\alpha, \beta, 1)^T$. Now if

$$\alpha^2 + \beta^2 \leq \mu^2, \tag{10}$$

the friction is sufficient to maintain sticking, and so $u_x = u_y = 0$ and $\mathbf{p}' = (\alpha, \beta, 1)^T$ for the duration of the collision.

If $\alpha^2 + \beta^2 > \mu^2$, the friction is not sufficient to maintain sticking, and sliding will immediately resume. Equation 9 is not valid when $u_x = u_y = 0$, and so is of no help in predicting the initial direction of sliding. In the case depicted in figure 5, there is a unique sliding direction leaving the origin; sliding must resume along this direction. It can be proven that the trajectories of equation 9 projected into the $u_x$-$u_y$ plane never spiral around the origin, and we conjecture that in cases when the friction is not sufficient to maintain sliding there is always exactly one sliding direction away from the origin. Once $u_x$ or $u_y$ is nonzero, equation 7 again applies.

Our previous algorithm for computing collision impulses must be slightly modified to account for possible sticking. If at any point during the integration of $\mathbf{u}$, $u_x$ and $u_y$ both vanish, the integration is halted. If the criterion given by equation 10 is met, sticking is maintained for the duration of the collision and both $\mathbf{u}$ and $\mathbf{p}$ vary along a straight line. Otherwise, sliding resumes and the integration continues as before. Figure 7 illustrates some of the possible trajectories of $\mathbf{u}$ for different collisions. Path $A$ represents a collision under low friction, in which the tangential component of relative contact velocity never vanishes, and the two objects slide on one another during the entire collision. Path $C$ corresponds to a collision in which the frictional forces bring the sliding contact to a halt; as the object rebound off each other there is no relative sliding velocity. Finally, path $B$ corresponds to a case in which sticking occurs momentarily, but the friction is insufficient to maintain this condition and sliding resumes.

### 4.5 Static friction under continuous contact

Consider a block sitting on a ramp, held at rest by friction. This continuous contact is modeled by a series of microcollisions. Under the model described thus far, the resulting behavior is exactly what would happen if

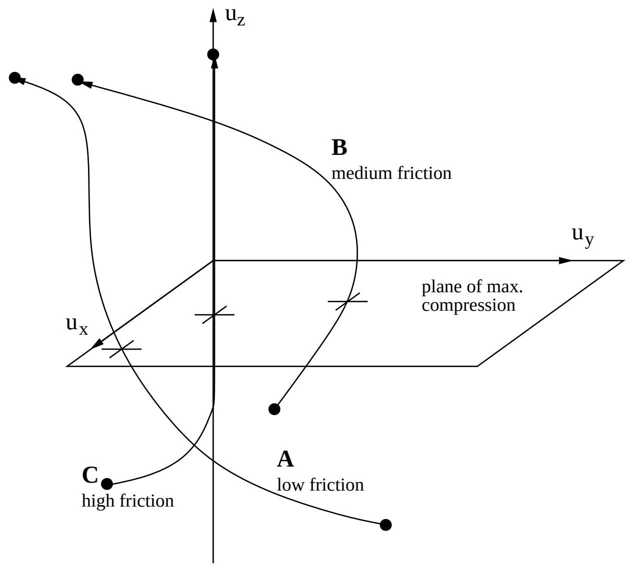


**Figure 7:** *Trajectories through relative contact velocity space for three different collisions.*

the ramp were experiencing very low amplitude, high frequency vibrations, namely, the block slowly creeps down the plane. The small collision impulses repeatedly bring the sliding to a stop, but during the ballistic phases gravity pulls the block slightly down the ramp. To eliminate this problem, a slight modification is needed to correctly model static friction under continuous contact.

A collision is characterized as a microcollision when the relative contact velocity in the normal direction is below some threshold. In this case, a check is first made to see if the impulse required to exactly reverse the contact velocity lies within the friction cone (using equation 6). If so, this impulse is applied, otherwise the full collision response computation is performed, as previously described.

There is physical basis for handling microcollisions in this manner. We are using impulses to model a contact force. In the case of one object statically resting on another, it is clear that this force does no work on the object. If we choose an impulse that accelerates the contact point velocity from $\mathbf{u}$ to $-\mathbf{u}$, and treat that impulse as a constant force acting for an infinitesimal time interval, than the impulse does no work. This is because the velocity of the contact point changes linearly from $\mathbf{u}$ to $-\mathbf{u}$ (equation 7, with $\gamma$ denoting time), therefore the time integral of force times velocity, i.e. work done, vanishes. With this modification, the collision response algorithm correctly models static friction under continuous contact—the block does not creep down the ramp.

# 5 Results

We describe a few results produced by our simulator. All of these simulations were computed at close to real time speeds, and would be real time on a slightly faster platform (we are using a Silicon Graphics Iris Indigo XZ). For example, the colliding coins simulation, which involves eight moving objects and five fixed objects simulated for five seconds, takes less than 15 seconds to generate. These fast simulation times also reflect the efficiency of the collision detection algorithm: the bowling pins are 162-facet polyhedra; the marbles each have over 5000 facets.

## 5.1 Simulation descriptions

The colliding coins simulation involves eight coins being tossed or rolled into the center of a platform; complex interactions between the coins are followed by a segment dominated by microcollisions as the coins settle down or roll off the platform (figure 8).

In the bowling simulation, a bowling ball is thrown down an alley, with the same initial angular velocity that a bowler gives the ball upon release. With the low coefficient of friction between the alley and the ball, the ball initially slides down the alley, but it gradually accumulates a component of angular velocity in the forward rolling direction. The slow shift from sliding mode to rolling mode is complete as the ball hits the pins. This process gives the ball the familiar hook seen when good bowlers bowl. This portion of the simulation validates our collision model, and simulation of continuous contact by microcollisions. In the latter part of the simulation, the ball knocks down the pins, in a complex assortment of colliding and continuous contact (figure 9—it's a strike!).

We have run various marbles simulations which study the behavior of nearly elastic collisions between rolling balls. One of particular interest is a simulation in which one rolling black marble hits the end of a line of four stationary marbles, causing the blue marble at the other end of the line to roll away, while the others remain basically at rest (figure 10). Constraint-based simulators often do not predict the correct response for these simultaneous or near-simultaneous collisions. However, under impulse-based dynamics this situation is treated no differently than any other series of collisions. Technically, the collisions are not simultaneous—they are transferred through the marble chain, just as they are in reality. It is not necessary to go through any contortions to get the proper response.

A final simulation worth mentioning is the ball on a spinning platter. What happens when a ball is placed on a spinning platter with a high coefficient of friction, such that the initial contact velocities match (i.e. the ball is rolling, not sliding)? The result is certainly non-intuitive, but has been verified by an actual experiment. The answer is that the ball rolls in circles, not centered at the axis of the platter, which gradually increase in radius until the ball rolls off of the platter [10]. When confronted with this problem, our simulator produced this very result, again verifying the collision model and the feasibility of generating correct macroscopic behavior through microcollisions (figure 11).

## 5.2 Conclusion

The impulse-based method is an excellent, new technique for dynamic simulation for two reasons: speed and accuracy. Simulations can be performed in real time, producing physically verifiable results. Most encouraging is the variety of systems and behaviors that can be modeled by our simulator; no ad hoc modifications or tweaking was necessary to produce results in agreement with the physical world. The impulse-based method is conceptually and algorithmically simpler than constraint-based methods, and perhaps the principle of Occam's razor applies here. After all, it seems most plausible that collisions forces in nature are based on local properties like contact velocity, and not on some global state configuration involving many bodies in contact.

There are situations in which constraint-based methods are more appropriate than the impulse-based approach—modeling an ideal hinge constraint by microcollisions between the hinge pin and its sheath would be slow and unnecessary (unless one were concerned with the actual vibration and slipping of the pin within the sheath). Future work should be done on integrating these two dynamic simulation methodologies into a hybrid system which can model linked bodies. We are also interested in enhancing our current simulator to make it more of an analysis tool. Statistics on contact forces and total impulses delivered would be useful, as well as a visualization capability for examining the forces at contact or collision points.

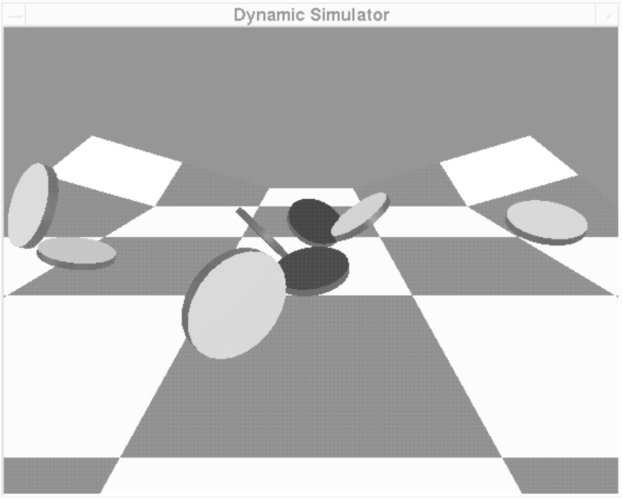


**Figure 8:** *Colliding coins.*

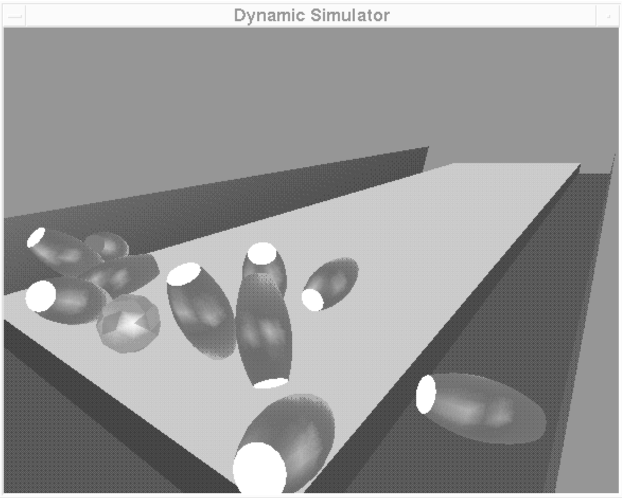


**Figure 9:** *Bowling.*

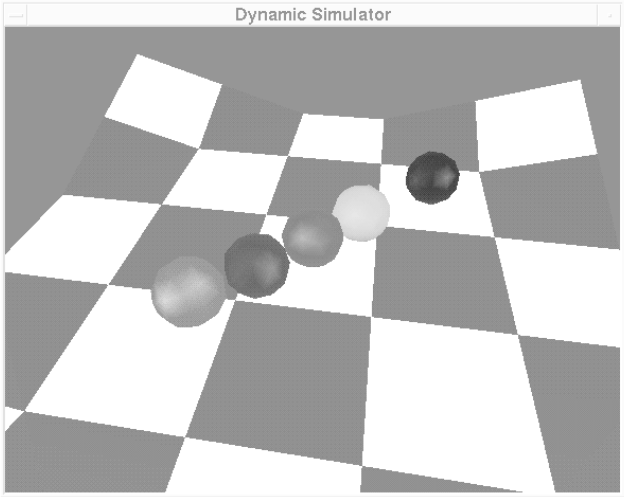


**Figure 10:** *Marbles. Snapshot taken just after the leftmost marble hit the marble chain.*

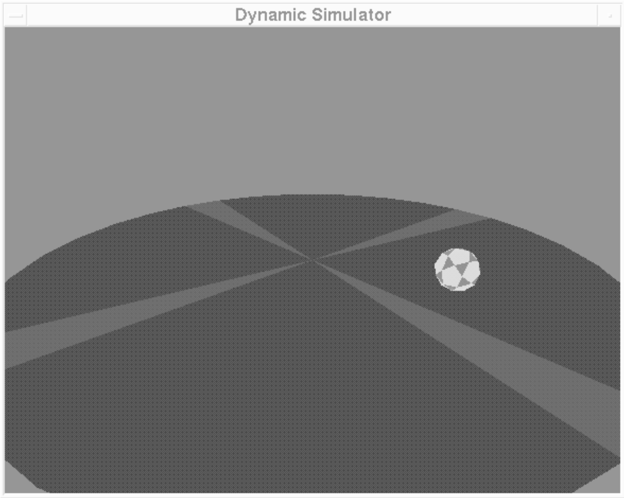


**Figure 11:** *Ball on a spinning platter.*

## Acknowledgements

This research was supported in part by an NSF Graduate Fellowship, a David and Lucile Packard Fellowship, and a National Science Foundation Presidential Young Investigator Award (# IRI-8958577). We thank Jeff Wendlandt, Ming Lin, and numerous other inhabitants of the Berkeley Robotics Lab for many fruitful discussions.

## References


[1] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics*, 23(3):223–232, July 1989.

[2] David Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *Computer Graphics*, 24(4):19–28, August 1990.

[3] David Baraff. Coping with friction for non-penetrating rigid body simulation. *Computer Graphics*, 25(4):31–40, August 1991.

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

[5] Ronen Barzel and Alan H. Barr. A modeling system based on dynamic constraints. *Computer Graphics*, 22(4):179–188, August 1988.

[6] James F. Cremer and A. James Stewart. The architecture of newton, a general-purpose dynamics simulator. In *International Conference on Robotics and Automation*, pages 1806–1811. IEEE, May 1989.

[7] James K. Hahn. Realistic animation of rigid bodies. *Computer Graphics*, 22(4):299–308, August 1988.

[8] John E Hopcroft. Electronic prototyping. *Computer*, pages 55–57, March 1989.

[9] J. B. Keller. Impact with friction. *Journal of Applied Mechanics*, 53, March 1986.

[10] A. Lewis, R. M'Closkey, and R. M. Murray. Modelling constraints and the dynamics of a rolling ball on a spinning table. Technical report, California Institute of Technology, 1993. Preprint.

[11] M.C. Lin and Dinesh Manocha. Interference detection between curved objects for computer animation. In *Models and Techniques in Computer Animation*, pages 431–57. Springer-Verlag, 1993.

[12] Ming C. Lin. *Efficient Collision Detection for Animation and Robotics*. PhD thesis, University of California, Berkeley, December 1993.

[13] Matthew Moore and Jane Wilhems. Collision detection and response for computer animation. *Computer Graphics*, 22(4):289–298, August 1988.

[14] M. Overmars. Point location in fat subdivisions. *Information Processing Letters*, 44:261–265, 1992.

[15] Edward J. Routh. *Elementary Rigid Dynamics*. 1905.

[16] A. James Stewart and James F. Cremer. Algorithmic control of walking. In *International Conference on Robotics and Automation*, pages 1598–1603. IEEE, May 1989.

[17] Yu Wang and Matthew T. Mason. Modeling impact dynamics for robotic operations. In *International Conference on Robotics and Automation*, pages 678–685. IEEE, May 1987.

[18] Andrew Witkin, Michael Gleicher, and William Welch. Interactive dynamics. *Computer Graphics*, 24(2):11–22, March 1990.

[19] Andrew Witkin and William Welch. Fast animation and control of nonrigid structures. *Computer Graphics*, 24(4):243–252, August 1990.