Impulse-based Simulation of Rigid Bodies

Brian Mirtich *
John Canny †
University of California at Berkeley

Abstract

We introduce a promising new approach to rigid body dynamic simulation called impulse-based *simulation. The method is well suited to modeling physical systems with large numbers of collisions, or with contact modes that change frequently. All types of contact (colliding, rolling, sliding, and resting) are modeled through a series of collision impulses between the objects in contact, hence the method is simpler and faster than constraint-based simulation. We have implemented an impulse-based simulator that can currently achieve interactive simulation times, and real time simulation seems within reach. In addition, the simulator has produced physically accurate results in several qualitative and quantitative experiments. After giving an overview of impulse-based dynamic simulation, we discuss collision detection and collision response in this context, and present results from several experiments.*

1 Introduction

The foremost requirement of a dynamic simulator is physical accuracy. The simulation is to take the place of a physical model, and hence its utility is directly related to how well it mimics this model. A second important requirement is computational efficiency. Many applications (e.g. electronic prototyping [9]) benefit most from interactive simulation; others (e.g. virtual reality) demand real time speeds.

This paper discusses a new approach to dynamic simulation called impulse-based simulation, founded on the twin goals of physical accuracy and computational efficiency. The initial results from our impulse-based simulator look very promising, both from speed and accuracy standpoints. In this paper we give an overview of the impulse-based approach, then discuss collision detection and resolution and results from several experiments.

* *mirtich@cs.berkeley.edu*, Department of Computer Science, 387 Soda Hall, University of California, Berkeley, CA 94720. Supported in part by NSF grant #FD93-19412.

† *jfc@cs.berkeley.edu*, Department of Computer Science, 529 Soda Hall, University of California, Berkeley, CA 94720. Supported in part by NSF grant #FD93-19412.

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 [14]. Hahn also pioneered dynamic simulation, modeling sliding and rolling contacts using impact equations [8]. His work is the precursor of our method, although we extend the applicability of impulse dynamics to resting contacts, and model multiple objects in contact with impulse trains as well. These early approaches all suffered from inefficient collision detection and unrealistic assumptions concerning impact dynamics (e.g. infinite friction at the contact point).

Cremer and Stewart describe *Newton* [7, 17], 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, although the contact modeling is fairly simplistic. Baraff has studied multiple rigid bodies in contact [1, 2], and shown that computing contact forces in the presence of friction is NP-hard [3]. A summary of his work in this area appears in [4].

There are few full treatments of frictional collisions. Routh [16] is still considered the authority on this subject, and more recently, Keller gives an excellent treatment of frictional collisions [10]. Our analysis is extremely similar to that of Bhatt and Koechling, who independently derived the same key equation for integration of relative contact velocities during impact. They give a classification of frictional collisions, based on the flow patterns of tangential contact velocity [6].

Wang and Mason have studied two-dimensional impact dynamics for robotic applications, based on Routh's approach [18]. Finally, a number of researchers have investigated several problems and paradigms for dynamic simulation and physical-based modeling [5, 19, 20].

2 The impulse-based method

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 [4, 5, 7, 19]. 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 no work on the ball, but only enforces a non-penetration constraint. In the Lagrangian constraint-based approach, this force is not modeled explicitly, but is accounted for by a constraint on the configuration of the ball (here, its z-coordinate is held constant). Alternatively, one may model the forces explicitly, solving for their magnitudes using Lagrange multipliers. However this still requires complete, exact knowledge of the instantaneous state of contact between the objects, since that determines where and when such forces can exist.

A 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 always clear what type of constraint should be applied; there exist at least two models for rolling contact which in some cases predict different behaviors [11]. 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 to constraint-based methods, impulse-based dynamics involves no explicit constraints on the configurations of the moving objects; when the objects are not colliding, they are in ballistic trajectories. Furthermore, all modes of continuous contact are handled via trains of impulses applied to the objects, whether they be resting, sliding, or rolling on one another. Under impulse-based simulation, a block resting on a table is actually experiencing many rapid, tiny collisions with the table, each of which is resolved using only local information at the collision point.

Now consider the case of a ball bouncing along the terrain shown in figure 1. Under constraint-based simulation, the

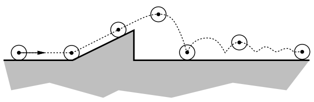

Figure 1: *A nightmare for constraint-based simulation.*

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 this sense, 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 phases.

We do not wish to discredit constraint-based methods of dynamic simulation; indeed, there are many situations for which they are the perfect tool. We believe the impulse-based method is better suited to simulating many common physical systems, especially those which are collision intensive, or that have many changes in contact mode. We examine the possibility of using both methods of simulation together, combining the strengths of each, in section 6.

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 to be practical? 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, and the approach is extremely fast. Simulations can certainly be run interactively with our current implementation, and we believe real time simulation is a reachable goal.

3 Collision detection

Impulse-based dynamic simulation is inherently collision intensive, since collisions are used to affect all types of interaction between objects. Hahn found collision detection to be the bottleneck in dynamic simulation [8], and efficient data structures and algorithms are needed to make impulse-based simulation feasible.

Currently in our simulator, all objects are geometrically modeled as convex polyhedra or combinations of them. The polyhedral restriction is not at all severe, because our collision detection system is very insensitive to the complexity of the geometric models, permitting fine tessellations. Indeed, some of the simulations described in section 5 use polyhedral models with over 20,000 facets, with negligible slowdown.

3.1 Prioritizing collisions

Obviously, checking for possible collisions between all pairs of objects after every integration step is too inefficient. Instead, collisions are prioritized in a heap (see figure 2). For

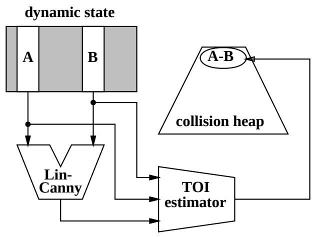

Figure 2: *Prioritizing collisions in a heap.*

each pair of objects in the simulation, there is an element in the heap, which also contains a lower bound on the time of impact (TOI) for the given pair of objects. The heap is sorted on the TOI field, thus the TOI field of the top heap element always gives a "safe" value for the next collision free integration step.

After an integration step, the distance between the objects on the top of the heap (call them A and B) must be recomputed. In our implementation, we use the Lin-Canny closest features algorithm [12]. This is an extremely efficient algorithm which maintains the closest features (vertices, edges, or faces) between a pair of convex polyhedra. It is fastest in applications like dynamic simulation, when the objects move continuously through space and geometric coherence can be exploited.

Collisions are declared when the distance between objects falls below some threshold ε_c. First suppose the distance between A and B lies above ε_c. In this case, the dynamic states of A and B along with the output of the Lin-Canny algorithm are used to compute a new conservative bound on the time of impact of A and B. The A–B heap pair is updated with this new value, possibly affecting its heap position, and the integrator is ready for another step.

If the distance between A and B is less than ε_c, a collision is declared. The collision resolution system computes and applies collision impulses to the two objects, changing their dynamic state. At this point the TOI is recomputed for these objects as before, however another step is necessary: the TOI between all object pairs of the form A–x and B–x must also be recomputed. The reason is that the TOI estimator uses a ballistic trajectory assumption to bound the time of impact for a pair of objects. Applying collision impulses to objects violates this assumption, and so every previous TOI involving such an object becomes invalid. Note that this is an $O(n)$ update step.

3.2 Further reducing collision checks and TOI updates

The strategy described above reduces collision checks significantly, especially between objects which are far apart or moving slowly. However, the number of collision checks is still $O(n^2)$ because they are performed periodically between every pair of objects. A more serious problem is the $O(n)$ TOI update step that must be performed every time a collision impulse is applied to an object. What the heap scheme misses is the fact that some objects never come near each other, and collision checks as well as TOI updates for such pairs of objects are unnecessary.

To alleviate this problem, we employ a spatial tiling technique based on Overmars' efficient point-location algorithms in fat subdivisions [15]. For each object i in the simulation, one can easily find an enclosing, axis-aligned rectangular volume B_i which is guaranteed to contain the object during the next integration step. This is possible because of the ballistic trajectory assumption.

The idea is to keep track of which objects are near each other, by keeping track of which bounding boxes overlap. To this end, the physical space is partitioned into a cubical tiling with resolution ρ. Under this tiling, Coordinates in physical space are mapped to integers under the tiling map τ:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \xrightarrow{\tau} \begin{bmatrix} \lfloor x/\rho \rfloor \\ \lfloor y/\rho \rfloor \\ \lfloor z/\rho \rfloor \end{bmatrix}. \tag{1}$$

Let S_i be the set of tiles which B_i intersects. We store i in a hash table multiple times, hashed on the coordinates of each tile in S_i. Clearly objects i and j can only possibly collide during the next integration step if i and j are both present in some hash bucket. Only in this case do we keep object pair i-j in the collision heap. Furthermore, if object i experiences a collision impulse, TOIs need only be recomputed for object pairs i-k, where object k shares a hash bucket with object i.

This scheme tremendously reduces the number of collision checks and TOI computations that must be performed, since most objects are generally in the vicinity of only a small subset of the set of all objects. Collision detection is still $O(n^2)$ in the worst case, but almost always better. Consider for example the case of simulating a vibratory bowl feeder sorting hundreds of small parts. Since the number of parts near another part can be bounded by a constant, the number of collision checks are $O(n)$.

One added wrinkle is that one must actually employ a hierarchy of spatial tilings and hash tables of varying resolutions, in order to prevent having to hash a sofa according to tiles the size of ice cubes. The hierarchy is needed to keep the rate of bucket updates small. See Overmars for more information on this multiple resolution hashing scheme [15].

3.3 Time of impact estimator

The time of impact (TOI) estimator takes the current dynamic state (pose and velocity) of two objects as well as the closest points between them, and returns a lower bound on the time of impact for those two objects. We assume the objects are convex; concavities are handled by convex decomposition.

Let $\mathbf{c}_i$ and $\mathbf{c}_j$ be the current closest points between two objects i and j on a collision course. Let $\hat{\mathbf{d}}$ be a unit vector in the direction of $\mathbf{c}_i - \mathbf{c}_j$, and d be the distance between $\mathbf{c}_i$ and $\mathbf{c}_j$. A convexity argument shows that no matter where the ultimate contact points are located, these contact points must cover the distance d in the direction of $\hat{\mathbf{d}}$ before collision can occur. From this one obtains a conservative bound on the time of collision:

$$t_c \geq \frac{d}{(\mathbf{v}_j - \mathbf{v}_i) \cdot \hat{\mathbf{d}} + r_i\omega_i + r_j\omega_j}, \tag{2}$$

where $\mathbf{v}$ denotes center of mass velocity, r denotes maximum "radius," ω denotes maximum angular velocity magnitude, and the subscripts refer to the body. This bound assumes both objects are ballistic, so that gravitational effects cancel out. If, for instance, object i is a fixed table top, then the gravitational acceleration of j must be accounted for.

The conservation of momentum can be used to bound the angular velocity magnitude of a body in a ballistic trajectory:

$$\omega_{max} \leq \frac{\|(J_x\omega_x, J_y\omega_y, J_z\omega_z)^T\|}{\min(J_x, J_y, J_z)}, \tag{3}$$

where $\mathbf{J}$ is the vector of diagonal elements of the diagonalized mass matrix, and ω is the current angular velocity.

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 center of mass and angular velocities for each body. After updating these velocities, dynamic state evolution can continue, assuming ballistic trajectories for all moving objects. The point of application is computed by the collision detection system, and hence the central problem in collision resolution is to determine the collision impulse $\mathbf{p}$. Accurate computation of this impulse is critical to the physical accuracy of the simulator. We now discuss how $\mathbf{p}$ may be computed; a more detailed discussion can be found in [13].

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 [16], although we derive equations which are more amenable to numerical integration. Keller also gives an excellent treatment [10], and Bhatt and Koechling's analysis is quite similar to ours [6]. There are three assumptions central to our analysis:

1. Infinitesimal collision time
2. Poisson's hypothesis
3. Coulomb friction model

The infinitesimal collision time assumption is commonly made in dynamic simulation [10]. It implies that the positions of the objects can be treated as constant over the course of a collision. Furthermore, the effect of one object on the other can be described by an impulse, which unlike a normal force can instantaneously change velocities. This assumption does *not* imply that the collision can be treated as a discrete event. The velocities of the bodies are not constant during the collision, and since collision (frictional) forces depend on these velocities, it is necessary to examine the dynamics during the collision. In short, a collision is a single point on the time line of the simulation, but to determine the collision impulses which are generated, one must use a magnifying glass to "blow up" this point, examining what happens inside the collision.

Poisson's hypothesis is an approximation to the complex deformations and energy losses which occur when two real bodies collide. Trying to explicitly model these stresses and deformations is too slow for interactive simulation; Poisson's hypothesis is a simple empirical rule that captures the basic behavior during a collision. A collision is divided into a compression and a restitution phase, based on the direction of the relative contact velocity along the surface normal. The boundary between these phases is the point of maximum compression, at which point the relative normal contact velocity vanishes. Let p_{total} be the magnitude of the normal component of the impulse imparted by one object onto the other over the entire collision, and p_{mc} be the magnitude of the normal component of the impulse just over the compression phase, i.e. up to the point of maximum compression. Poisson's hypothesis states

$$p_{total} = (1+e)p_{mc} \tag{4}$$

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

Our final assumption is the Coloumb friction law. At a particular point during a collision between bodies A and B, let $\mathbf{u}$ be the contact velocity of A relative to B, 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) components of force exerted by B on A, respectively. Then

$$\mathbf{u}_t \neq \mathbf{0} \quad \Rightarrow \quad \mathbf{f}_t = -\mu\|\mathbf{f}_n\|\hat{\mathbf{u}}_t \tag{5}$$

$$\mathbf{u}_t = \mathbf{0} \quad \Rightarrow \quad \|\mathbf{f}_t\| \leq \mu\|\mathbf{f}_n\| \tag{6}$$

where μ 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. $\mathbf{u}_t$ vanishes), all that is known is that the total force lies in the friction cone.

4.2 Initial collision analysis

A possible collision is reported whenever the distance between two bodies falls below the collision epsilon, ε_c. This is only a *possible* collision, because the objects may be receding. If the normal component of the relative velocity of the closest points has appropriate sign, no collision impulse should be applied. Note we are assuming the existence a normal direction; polyhedral objects have discontinuous surface normals, however reasonable surface normals can always be found.

Establish a collision frame with the z-axis aligned with the collision normal, directed towards body 1. Let $\mathbf{u} = \mathbf{u}_1 - \mathbf{u}_2$ be the relative contact velocity between bodies 1 and 2. When $u_z < 0$, a collision impulse must be applied to prevent interpenetration; it is necessary to analyze the dynamics of the bodies during the collision to determine this impulse. We use γ to denote the collision parameter; that is, γ is a variable which starts at zero, and continuously increases through the course of the collision until it reaches some final value, γ_f. All velocities are functions of γ, and $\mathbf{p}(\gamma)$ denotes the impulse delivered to body 1 up to point γ in the collision. The goal is to determine $\mathbf{p}(\gamma_f)$, the final total impulse delivered.

Initially, one might choose γ 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, begging the question of how to choose the spring constants. Nonetheless, a variety of "penalty methods" do attempt to choose such spring constants.

A way of avoiding this problem 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. Let $\Delta\mathbf{u}(\gamma)$ denote the total change in relative contact velocity at point γ in the collision, and $\mathbf{p}(\gamma)$ be the impulse delivered to body 1 up to this point. Straightforward physics leads to the equation

$$\Delta\mathbf{u}(\gamma) = M\mathbf{p}(\gamma) \tag{7}$$

(see [13] for a detailed analysis). Here, M is a 3×3 matrix dependent only upon the masses and mass matrices of the colliding bodies, and the locations of the contact points relative to their centers of mass. By our infinitesimal collision time assumption, M is constant over the entire collision. It is useful to differentiate equation 7 with respect to the collision parameter γ, obtaining

$$\mathbf{u}'(\gamma) = M\mathbf{p}'(\gamma). \tag{8}$$

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 γ is chosen to be p_z, then while the bodies are sliding relative to one another,*

$$\mathbf{p}' = \begin{bmatrix} -\mu\cos\theta \\ -\mu\sin\theta \\ 1 \end{bmatrix}. \tag{9}$$

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, $f_x = -(\mu\cos\theta)f_z = -(\mu\cos\theta)\frac{dp_z}{dt}$. 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 8, with θ expressed in terms of u_x and u_y, we obtain:

$$\begin{bmatrix} u'_x \\ u'_y \\ u'_z \end{bmatrix} = M \begin{bmatrix} -\mu\frac{u_x}{\sqrt{u_x^2+u_y^2}} \\ -\mu\frac{u_y}{\sqrt{u_x^2+u_y^2}} \\ 1 \end{bmatrix}. \tag{10}$$

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 γ (i.e. p_z), we can track $\mathbf{u}$ during the course of the collision. Projections of the trajectories into the u_x-u_y plane are shown in figure 3 for a particular matrix M; the crosses mark the initial sliding velocities.

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 10. During this integration, u_z will increase[1]. When it reaches zero, the point of maximum compression has been attained.

[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.

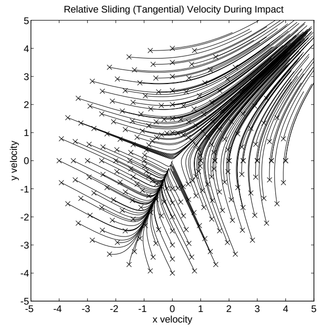

Figure 3: *Solution trajectories of equation 8 projected into the u_x-u_y plane.*

At this point, p_z is the total normal impulse which has been applied during compression. Multiplying this value by $(1 + e)$ gives the terminating value for the collision parameter, γ_f. The integration then continues to this point, to obtain $\Delta\mathbf{u}(\gamma_f)$. Inverting equation 7 then gives the total collision impulse $\mathbf{p}(\gamma_f)$.

4.4 Sticking mode

When the relative tangential velocity vanishes, the direction of the frictional force is not known a priori, and lemma 1 no longer applies. We assume like Routh 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 8, and solve for $\mathbf{p}'$. There is a unique solution for which $p'_z = 1$, say $\mathbf{p}' = (\alpha, \beta, 1)^T$. If

$$\alpha^2 + \beta^2 \leq \mu^2, \tag{11}$$

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

If $\alpha^2 + \beta^2 > \mu^2$, the friction is not sufficient to maintain sticking, and sliding will immediately resume. Equation 10 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 3, there is a unique sliding direction leaving the origin; sliding must resume along this direction. It can be proven that the trajectories of equation 10 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 10 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 halts. If the criterion given by equation 11 is met, sticking is maintained for the duration of the collision and both $\mathbf{u}$ and $\mathbf{p}$ vary along a straight line. Otherwise, we solve a quartic equation to determine the inward and outward sliding directions for the collision, and take the next integration step along the (conjectured unique) outward sliding direction. Once the sliding has resumed, the normal integration can continue;

Figure 4 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.

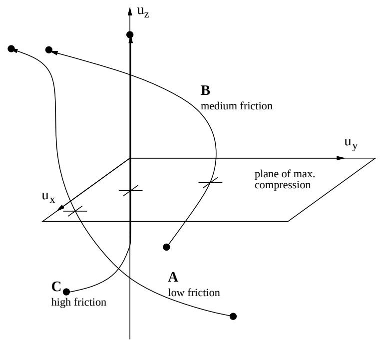

Figure 4: *Trajectories through relative contact velocity space for three different collisions.*

4.5 Static contact and microcollisions

The collision resolution method described thus far is suitable for resolving colliding contacts, but is not enough to model continuous contact as objects come to rest upon one another. In this case, the collisions must not produce an energy loss in the colliding objects, since they are modeling static forces which do no work.

Two important questions are: how can this static situation be detected using only local information at the contact point, and how should the collision model be modified to give the correct macroscopic behavior? Certainly the initial relative normal velocity at the contact point must be small; static contact only occurs as objects begin to settle onto one another. We define "small" precisely with the threshold v_e, the velocity an initially resting object acquires as it falls through the collision envelope:

$$v_e = \sqrt{2g\varepsilon_c}, \tag{12}$$

where g is the acceleration of gravity. If the relative normal velocity is below this threshold, a check is made to see if the impulse required to reverse the initial relative velocity lies within the friction cone, and if so, it is applied to resolve the collision. Such a collision is called a *microcollision.* One can show that microcollision impulses do no work on the object, just like the physical static contact forces that they model.

Microcollisions also solve another problem. Consider a block sitting on a shallow ramp with high friction, and modeling the contact as an impulse train. Even though the friction is sufficient to bring the sliding velocity to zero at every collision, the block will tend to creep down ramp because of the time it spends in a ballistic phase. However, the elastic nature of microcollisions will negate the effect of gravity during the intervening ballistic phases, by giving the block a small "kick" back up the ramp, once the tangential contact velocities become small enough. Figure 6 shows that microcollisions can bring the block to a complete stop.

The question arises as to whether microcollisions are not just some ad-hoc modification necessary to make impulse-based dynamics work. After all, one of the attractive features of the impulse-based method is that one need not enforce strict continuous contact constraints between obstacles.

Are microcollisions just such a constraint in disguise? The answer is no. As objects settle on one another, they experience a number of small collisions, none of which are initially microcollisions. Gradually, microcollisions account for a larger and larger fraction of total collisions, until eventually all collisions are microcollisions. In other words, there is a smooth transition between colliding and continuous contact. Moreover, the decision to apply a microcollision is based solely on local information at the contact point, not on some global information about the state of the system.

5 Results and Analysis

We have tested our simulator on a wide variety of problems. We now describe some qualitative and quantitative results.

5.1 Pool break

This simulation involved breaking a rack of fifteen pool balls with a high velocity cue ball. Constraint-based simulators have trouble with this example because of the large number of mutual contacts between the racked balls. Baraff has shown that the problem of finding a set of contact forces that instantaneously obey the Coulomb friction law at every contact point is NP-hard [3]. Furthermore, the contact constraints are quite transient, making it difficult to integrate along equations of motion derived from them.

The impulse-based method avoids these problems by treating the contacts as a series of closely spaced collisions. The racked balls (of standard size) were initially placed 0.1 millimeters apart. This distance is below ε_c, and thus when the cue ball strikes the rack, many collisions occur before the balls even begin to roll. Figure 5 show the high number of collisions that occurred during this simulation, especially at the point of the initial break. However, the simplicity of the collision model still permits fast simulation (see table 1). After the break, the collision rate stabilizes at roughly 3 kHz; these collisions are primarily between the balls and the table.

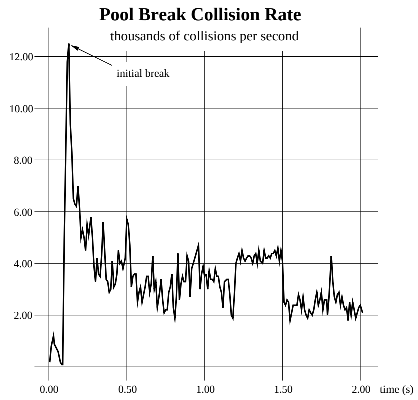

Figure 5: *Collision rate during a pool break.*

5.2 Block on ramp experiments

A good set of benchmarks for the physical accuracy of the collision model are "block on ramp" tests, involving a block sliding down a ramp with friction. We used a 20° ramp; the critical coefficient of friction at which the frictional force exactly resists the tangential component of gravity was $\mu_c = \tan 20° = 0.37$.

For the first test, the coefficient of friction was set to $\mu = 0.5 > \mu_c$, and the block was given an initial velocity down the ramp of 125 cm/sec. The theoretical and simulated velocities of the block down the ramp are shown in figure 6. The jaggedness of the simulated velocity curve is due

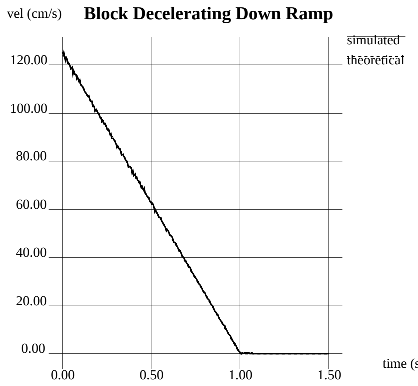

Figure 6: *Block velocity,* $\mu = 0.5 > \mu_c$.

to the discrete impulse train modeling the contact, however the average simulated velocity and the simulated position (figure 7) closely agree with theory.

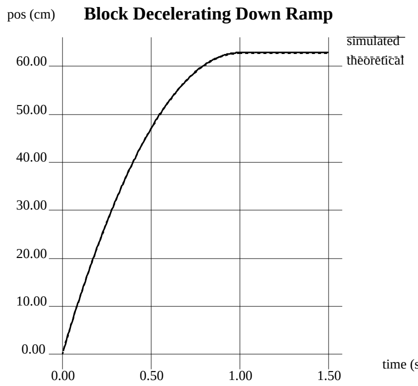

Figure 7: *Block position,* $\mu = 0.5 > \mu_c$.

In a second test, the coefficient of friction was lowered to $\mu = 0.25 < \mu_c$, with the block beginning at rest. The theoretical and simulated velocities and positions are shown in figures 8 and 9, respectively. There is close agreement between simulation and theory; the slopes of the two velocity curves are nearly identical, indicating that the impulse-based model predicts the correct frictional force on the block.

5.3 Measuring the strike pocket

We used our simulator to study the effect of a hooking ball on the width of the "strike pocket" in standard tenpin bowling. The best place for the ball to hit the pins is between the head pin and a second row pin; good bowlers throw a hooking ball, which hits the pins moving toward the center of the arrangement.

How does a hooking ball affect the chances of bowling a strike? The chaotic nature of the system makes a mathematical analysis nearly impossible, and it is also difficult to perform real experiments with sufficient control over conditions. In short, the problem is ideal for stochastic simulation. It is also a perfect application for impulse-based dynamics—the

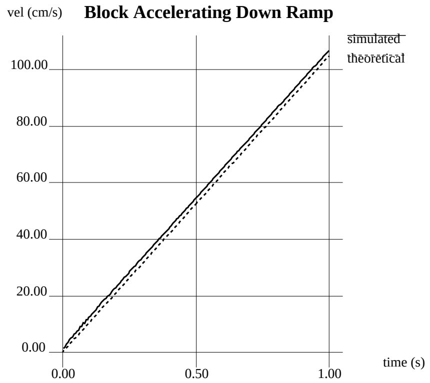

Figure 8: *Block velocity, $\mu = 0.25 < \mu_c$.*

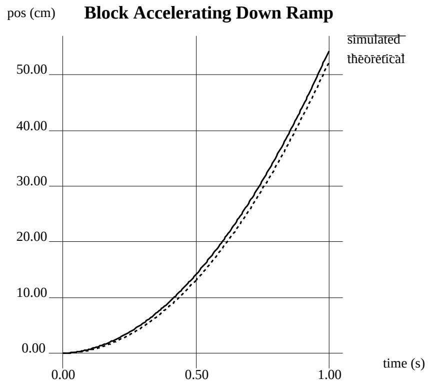

Figure 9: *Block position, $\mu = 0.25 < \mu_c$.*

evolution is collision intensive, with many transient contacts between objects, and there is a gradual change in contact mode between the ball and the alley (bouncing to sliding to rolling). For our simulations, we used accurate physical dimensions for the alley, ball and pin sizes and masses, pin spacing, etc.; a slight approximation was made in the shape of the pins.

In the first batch of simulations, a straight ball was thrown down the alley by launching the ball with zero angular velocity, and a center of mass velocity in the $+y$ direction (see figure 10). We performed 320 trials, keeping the initial ball velocities constant, but varying the initial x-coordinate of the ball's center of mass over a 40 centimeter window, recording the number of felled pins for each trial. In a second batch of 320 trials, the initial ball velocity conditions were altered to produce a right-hander's hooking ball: angular velocity of -12 rad/s in the $+y$ direction and a linear velocity at an angle of $-2°$ from the y-axis.

Figure 11 shows the number of felled pins versus the ball position as it crossed the pin line (ordinates are averaged over 5 mm wide abscissa windows). The hooking ball is slightly better than the straight ball at most positions along the pin line, and is significantly better over a range between the head pin and rightmost second row pin (+6 to +12 cm on the pin line). This agrees with the accepted wisdom that a right-handed bowler's best strategy is to throw a hooking ball between these two pins. The plots also illustrate the dip in felled pins due to splits, when the ball hits the head pin dead on.

We could improve our model by more carefully specifying

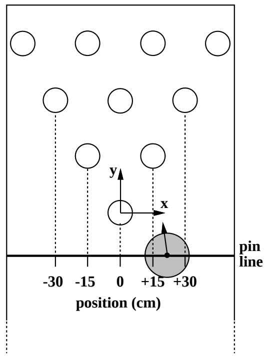

Figure 10: *Set up for the measurement of the strike pocket.*

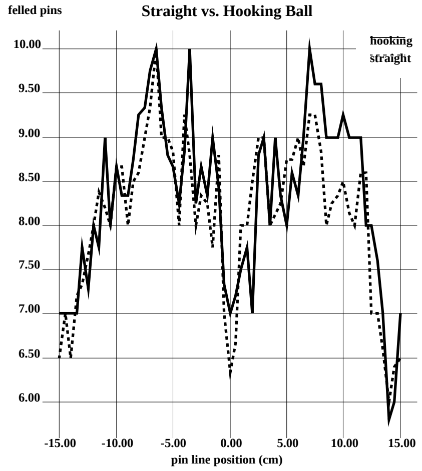

Figure 11: *Results from the strike pocket study.*

the shape of the pins and location of the ball's center of mass, which is not in general at the geometric center. However, this experiment demonstrates the feasibility and utility of using impulse-based dynamics for modeling a complex system and generating physically accurate results.

5.4 Other simulations

We briefly mention several other simulator problems we have tried, and summarize the execution time results for our simulator (see figure 12 for simulation snapshots).

Ball on spinning platter. This simulation involves a ball rolling on a disc that is spinning at high velocity. The example is interesting because of the nonholonomic constraint between the ball and disc, and in fact there are two classical models for this rolling contact which predict different behaviors! Experimental results show that the ball rolls in circles of gradually increasing radii, eventually rolling off the platter [11]. Our impulse-based simulator produces this result, demonstrating correct macroscopic behavior from the impulse-based contact model.

Block dropped on block. One block is dropped onto another, the former coming to rest on the latter.

Dominos. A line of seven dominos is set in motion by bumping the lead domino.

Chain of balls. Five balls the are placed next to each other in a straight line, and a rolling ball strikes the chain on one end. The momentum is transferred to the other end of the chain, launching the end ball.

Coins. Eight coins are tossed onto the same general area of a flat plate, and come to rest with some partially on top of others. This simulation is a good test case for all the contact modes: colliding, sliding, rolling, and resting.

Balls in dish. Seven balls are dropped into a shallow dish approximated by planar wedges. The balls come to rest in the physically accurate minimum energy configuration: one ball at the center of the dish, surrounded by the six other balls.

Table 1 gives the simulation times for all of the experiments. *Virtual time* is the length of time which passed in the simulation, *real time* is the actual time needed to compute the simulation[2], and *slowdown* is the ratio of the latter to the former (a 1.0 slowdown corresponds to real time simulation).

simulation	virtual time (s)	real time (s)	slow-down
pool break	3.0	95	32
dec. down ramp	1.5	41	27
acc. down ramp	1.0	23	23
bowling a strike	5.0	167	23.9
ball on platter	40	129	3.2
block drop	0.78	6.7	8.6
dominos	1.2	17	14
chain of balls	2.5	7.3	2.9
coins	3.6	50	14
balls in dish	7.8	95	12

Table 1: *Simulation times for experiments.*

6 Conclusions

We have described the impulse-based approach to dynamic simulation, and reported results from several simulation problems. Interactive simulation speeds have already been attained, and we believe real time simulation is ultimately possible. Also encouraging is the wide variety of physical systems that we have successfully simulated; no special tweaking was performed for any of the simulations we have described. One important efficiency point is that the impulse-based approach is highly parallelizable. Because there are no global constraints on the state of the system, the dynamic integration of an n body system is neatly decomposed into n small pieces. Such a decomposition is not possible when there are explicit constraints between the states of different bodies.

The issue of physical accuracy is also an important one to consider. Modeling a rock sitting on a table through a series of impulses seems at first questionable. However, we are not making the claim that the rock is actually experiencing microcollisions, only that by modeling the contact in this way, the correct macroscopic behavior is affected. Our simulator has produced physically plausible results for many problems. Furthermore, quantitative results withstand scrutiny when compared to theoretical models. More study is needed here, but the initial results are encouraging.

As stated previously, we do not intend impulse-based dynamics to be a complete replacement for constraint-based dynamics. A perfect application for the latter is the modeling of a hinge joint. In principle, one could model the joint in an impulse-based way, enforcing the hinge constraint through collisions between the hinge pin and sheath. However, the impulse-based approach is clearly the wrong tool for this natural constraint-based problem. We are currently adding a multibody capability to our simulator, in order to model linked rigid body structures. We are using a hybrid approach: constraint-based methods are used to enforce joint constraints, while impulse-based dynamics are used to model contact between bodies not connected via joints. We are optimistic that using the right tool for the right problem can greatly extend the frontier of dynamic simulation.

[2] Real times were computed by averaging over several trials. All simulations were performed on an *SGI Indigo I*.

References

[1] Baraff, David. Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies. Computer Graphics, 23(3):223–232, July 1989.

[2] Baraff, David. Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation. Computer Graphics, 24(4):19–28, August 1990.

[3] Baraff, David. Coping with Friction for Non-penetrating Rigid Body Simulation. Computer Graphics, 25(4):31–40, August 1991.

[4] Baraff, David. Issues in Computing Contact Forces for Non-penetrating Rigid Bodies. Algorithmica, 10:292–352, 1993.

[5] Barzel, Ronen and Barr, Alan H. A Modeling System Based on Dynamic Constraints. Computer Graphics, 22(4):179–188, August 1988.

[6] Bhatt, Vivek and Koechling, Jeff. Classifying Dynamic Behavior During Three Dimensional Frictional Rigid Body Impact. In International Conference on Robotics and Automation. IEEE, May 1994.

[7] Cremer, James F. and Stewart, A. James. The Architecture of Newton, a General-purpose Dynamics Simulator. In International Conference on Robotics and Automation, pages 1806–1811. IEEE, May 1989.

[8] Hahn, James K. Realistic Animation of Rigid Bodies. Computer Graphics, 22(4):299–308, August 1988.

[9] Hopcroft, John E. Electronic Prototyping. Computer, pages 55–57, March 1989.

[10] Keller, J. B. Impact with Friction. Journal of Applied Mechanics, 53, March 1986.

[11] Lewis, A. and M'Closkey, R. and Murray, Richard. Modelling Constraints and the Dynamics of a Rolling Ball on a Spinning Table. Technical report, California Institute of Technology, 1993. Preprint.

[12] Lin, Ming C. and Canny, John F. A Fast Algorithm for Incremental Distance Calculation. In International Conference on Robotics and Automation, pages 1008–1014. IEEE, May 1991.

[13] Mirtich, Brian and Canny, John. Impulse-based Dynamic Simulation. In K. Goldberg, D. Halperin, J.C. Latombe, and R. Wilson, editors, The Algorithmic Foundations of Robotics. A. K. Peters, Boston, MA, 1995. Proceedings from the workshop held in February, 1994.

[14] Moore, Matthew and Wilhelms, Jane. Collision Detection and Response for Computer Animation. Computer Graphics, 22(4):289–298, August 1988.

[15] Overmars, Mark. Point Location in Fat Subdivisions. Information Processing Letters, 44:261–265, 1992.

[16] Routh, Edward J. Elementary Rigid Dynamics. 1905.

[17] Stewart, A. James and Cremer, James F. Algorithmic Control of Walking. In International Conference on Robotics and Automation, pages 1598–1603. IEEE, May 1989.

[18] Wang, Yu and Mason, Matthew T. Modeling Impact Dynamics for Robotic Operations. In International Conference on Robotics and Automation, pages 678–685. IEEE, May 1987.

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

[20] Witkin, Andrew and Welch, William. Fast Animation and Control of Nonrigid Structures. Computer Graphics, 24(4):243–252, August 1990.

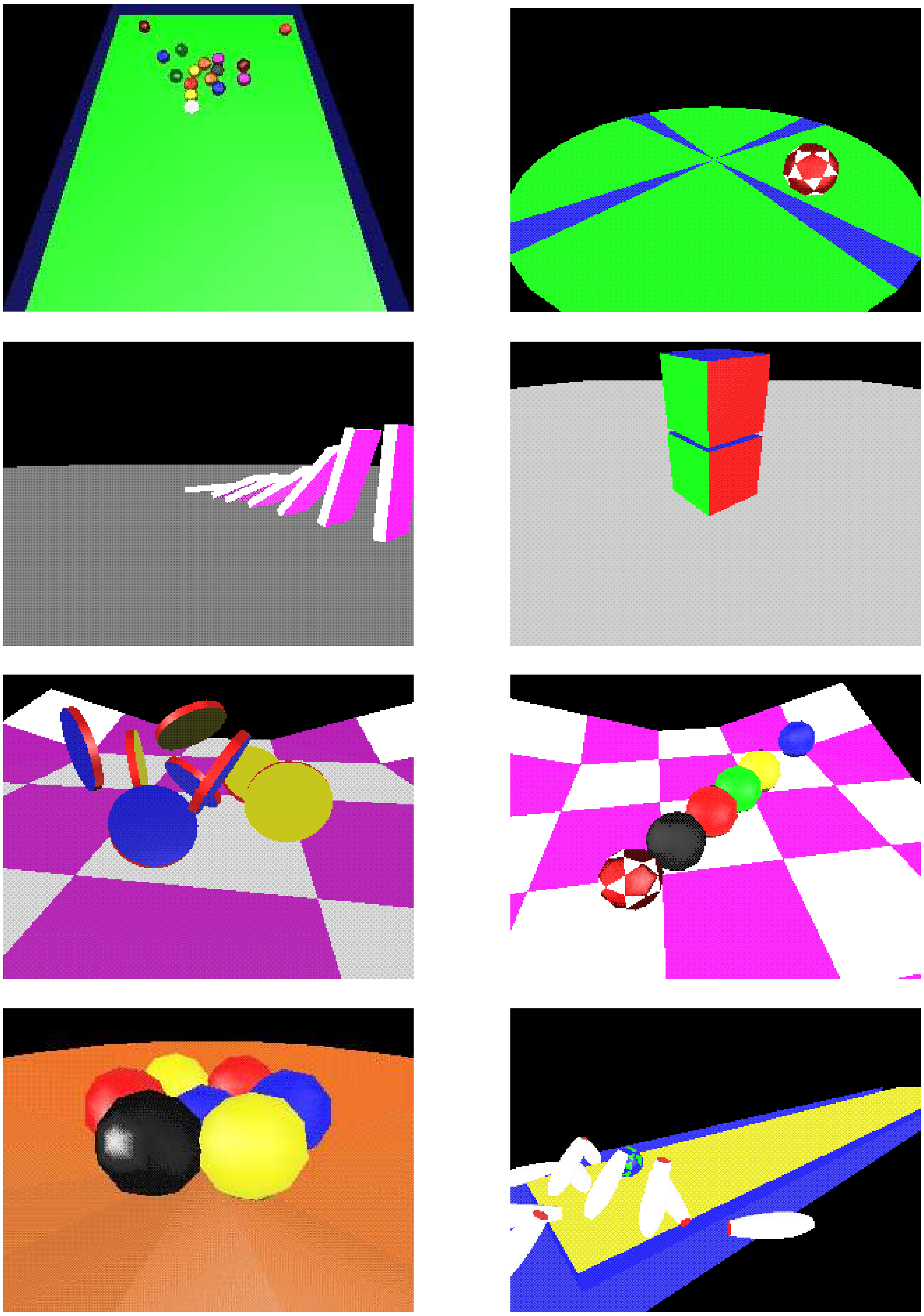

Figure 12: *Simulation snapshots.*