Backward Steps in Rigid Body Simulation

Christopher D. Twigg
Carnegie Mellon University

Doug L. James
Cornell University

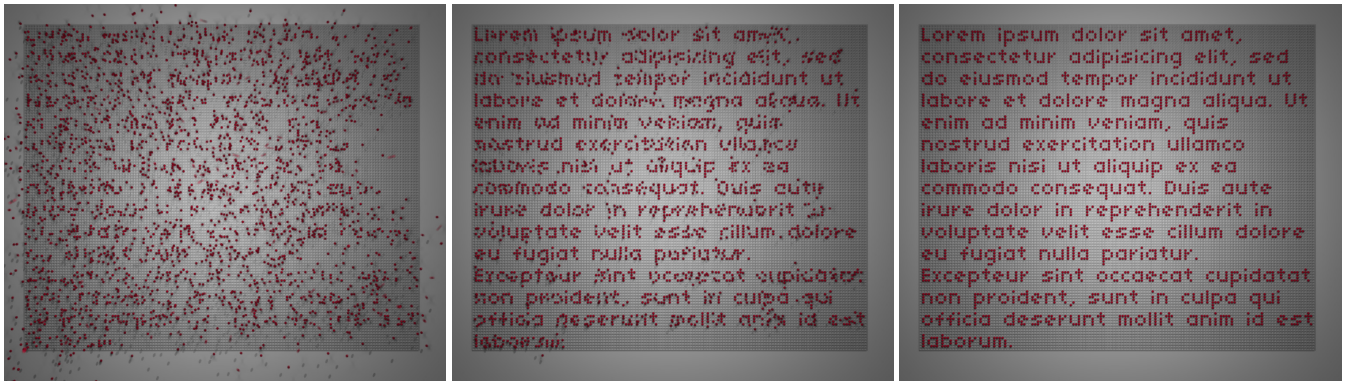


Figure 1: **A time-reversed rigid body simulation** *with 3037 rigid balls bouncing, colliding, and sliding into place that was computed in under one hour.*

Abstract

Physically based simulation of rigid body dynamics is commonly done by time-stepping systems *forward* in time. In this paper, we propose methods to allow time-stepping rigid body systems *backward* in time. Unfortunately, reverse-time integration of rigid bodies involving frictional contact is mathematically ill-posed, and can lack unique solutions. We instead propose time-reversed rigid body integrators that can sample *possible* solutions when unique ones do not exist. We also discuss challenges related to dissipation-related energy gain, sensitivity to initial conditions, stacking, constraints and articulation, rolling, sliding, skidding, bouncing, high angular velocities, rapid velocity growth from micro-collisions, and other problems encountered when going against the usual flow of time.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.6.8 [Simulation and Modeling]: Types of Simulation—Animation,

Keywords: rigid body dynamics, frictional contact, linear complementarity, ill-posedness, inverse problems, motion planning

1 Introduction

With few exceptions, physically based animation is formulated as an initial value problem: initial conditions are specified, and a time-stepping scheme advances the system forward until the desired motion is generated. While this is suitable for the vast majority of situations, there are times where we care more about the *final* configuration of the system, or an intermediate one, rather than the starting configuration. In these cases, it can make more sense to use *time-reversed simulation*, in which we specify the final configuration of the system (possibly at rest) and step it *backward in time* until we have generated enough motion. An illustrative example is shown in Figure 1.

In simple mathematical terms, one can consider "time reversal" of Newton's equations of motion, $\mathbf{f}=\mathbf{Ma}$, as just the replacement of time, t, by $-\tau$, where τ is a "reverse time" variable:

$$\mathbf{M}\frac{d^2\mathbf{x}}{dt^2}=\mathbf{f}\left(\mathbf{x},\frac{d\mathbf{x}}{dt},t\right) \quad \overset{t\to-\tau}{\longrightarrow} \quad \mathbf{M}\frac{d^2\mathbf{x}}{d\tau^2}=\mathbf{f}\left(\mathbf{x},-\frac{d\mathbf{x}}{d\tau},-\tau\right). \quad (1)$$

If the forces are autonomous and independent of velocity, e.g., $\mathbf{f}=\mathbf{f}(\mathbf{x})$, such as for a bunch of particles interacting via gravity, then the forces will look the same forward or backward in time, and the system can be said to have *time-reversal symmetry* [Reichenbach 1999]. However, for other systems, such as those involving velocity-dependent damping forces, the dynamics are not necessarily symmetric, e.g., if energy is lost moving forward in time, then it is gained moving backward in time. While the extension to rigid body animation might seem straightforward, unfortunately, as they say, the "devil is in the details," and running systems backward in time is not for the faint of heart.

Perhaps the most discouraging fact is that, unlike in forward simulation, we can not uniquely solve a backward simulation problem, in general. For dissipative systems, such as rigid bodies with frictional contact, the problem of generating motion ending in a given state is vastly under-determined; mathematically speaking, it is an ill-posed problem. For example, take a single block sitting at rest on a plane, and consider the nearly limitless set of motions ending in that position. It could have been sitting in place for a second or a year. It could have balanced on a single edge for an arbitrary length of time before falling victim to gravity and landing on its face. It could have slid in from any compass direction, while rotating (or not) about its vertical axis. It could have bounced some before sliding, or simply fallen a vertical mile only to land perfectly in place. With friction, it is even possible for the box to slide, then bounce, then slide again. If the geometry were rounded, it might have rolled into place, perhaps while sliding or bouncing.

In this paper, we consider time-reversed simulation of rigid bodies with frictional contact. Since we know we can not find a unique

solution to backward simulation, we take a different approach. We propose to sample *possible* backward motions that could have generated the final simulation state. Doing so will require a holistic approach: we must consider not just the mathematics and time-stepping scheme, but also sampling strategies and ways to communicate user intent about the desired motion, e.g., "when should it start moving, and in what way." To begin with, in the first half of the paper, we propose changes to a typical rigid body time-stepping scheme that allow it to step backward in the presence of frictional contacts; we consider schemes based on the Linear Complementarity Problem (LCP) formulation.

Unfortunately, merely being able to take backward timesteps of some *possible* motion does not guarantee that we will generate long sequences of *plausible* motion. The true challenges and limitations of backward simulation emerge only *after* we have designed and implemented a time-stepping scheme. It turns out that the naïve application of reverse simulation results in motion that, while perhaps technically and physically possible, can be qualitatively and quantitatively different from typical forward simulations. Therefore, in the second half of the paper, we catalogue a number of situations where any basic reverse simulator with a limited time horizon will generate noticeable artifacts. Where appropriate, we also suggest possible solutions, which may in some cases violate physics, but which produce noticeably less offensive motion.

Terminology Before continuing on to discuss time-reversed simulation in depth, we should pause to establish some terminology. We will use the term *forward* to refer to *increasing* time: t, $t + \Delta t$, $t + 2\Delta t$, ... *Backward* and *reverse* will refer to decreasing time: t, $t - \Delta t$, $t - 2\Delta t$, ... Similarly, with respect to a reference time t_{ref}, *before* will refer to times $t < t_{\text{ref}}$ and *after* will refer to times $t > t_{\text{ref}}$. Although we will refer to forward simulation frequently, the problems we will actually be solving will all use backward integration, so the *initial conditions* for the solver will always refer to the simulation state we provide as input to the reverse solver, which will often (but not always) consist of the bodies at rest.

Related Work

Rigid body simulation has a rich history in both the mechanics and computer graphics literature. As we will discuss in §2, our method is based on the Linear Complementarity Problem (LCP) formulation due to Baraff [1991; 1994] and Stewart and Trinkle [1996; 1997], but many other methods are available. Perhaps the simplest techniques to implement are those which handle contacts by applying instantaneous impulses to the bodies; continuous contacts are handled using a series of small impulses [Hahn 1988; Mirtich and Canny 1995]. While these methods are fast, they tend to handle stacking behavior poorly and do not accurately model friction.

Also easy to implement are penalty-based contact methods, such as Hertz, Kelvin-Voigt, or Hunt-Crossley models (see [Hunt and Crossley 1975; Lankarani and Nikravesh 1994]). As in (1), these force-based models are trivial to reverse (just negate the damping term), but can be slow compared to competing constraint-based methods, and they generally do not handle complicated stacking behavior well due to the difficulty in incorporating a proper Coulomb-like friction model (although see [Mirtich 2000]).

Guendelman and colleagues introduced a timestepping scheme which interleaves contact handling and timestepping [2003]. Contacts are handled one at a time in a relaxation scheme, but convergence is accelerated through use of the contact graph [Hahn 1988]. Weinstein and colleagues extended this method to articulated bodies with control [2006]. Kaufman and colleagues [2005] solved for the contact forces on each body separately, holding other bodies in the scene fixed. Milenkovic and Schmidl [2001] formulated the contact problem as a quadratic program. It is likely that time-reversed simulation methods could be developed based on any of these methods, although the issues that we discuss in §4 arise regardless of the underlying time-stepping scheme.

A number of optimization-based techniques have been developed for finding solutions to the boundary-value problem (BVP) in which both the object's initial and final states are specified by the user. In particular, spacetime constraints [Witkin and Kass 1988] could be interpreted as a BVP-style reverse-integration scheme. However, giving the user this level of control necessarily involves tradeoffs, and for optimization-based methods the drawbacks are performance and scalability. Popović and colleagues [2000; 2003] use gradient descent to find solutions quickly; the initial guess for the optimizer is a single forward simulation from the starting condition, or, in later work, the user's sketched motion. Chenney and Forsyth [2000] used sampling techniques to solve various boundary-value problems, while Twigg and James [2007] added user input to speed convergence. In cases with small numbers of objects and where both the start and end states are specified, any of these approaches would be superior to ours, but we demonstrate that our method can scale up to larger numbers of objects.

Adjoint methods for gradient computation step through the simulation forward and then backward [Wojtan et al. 2006; McNamara et al. 2004], but the backward stage involves performing computations on the existing simulation rather than computing new time-reversed motion. Similarly, time-warp rigid body simulation [Mirtich 2000] involves "rolling back" bodies to earlier states, but only to states that were computed previously using forward simulation.

Geometric integrators are time-stepping schemes that preserve certain invariants of the system, such as energy [Leimkuhler and Reich 2005; Kharevych et al. 2006]. These methods are ideal for simulating N-body systems where exact energy preservation is needed to prevent orbits from spiraling inward, and they guarantee that the dynamics are reversible. However, this energy preservation and reversibility does not hold across frictional collision events, and indeed there are cases (e.g., a body at rest) where the trivial solution computed by a reversible integrator will not generate the desired behavior. In a similar vein, Gear and Kevrekidis [2004] take a series of forward steps and use a polynomial fit to exterapolate backward, but this assumes that the function is smooth which does not hold in the presence of contacts.

2 Background: Forward Steps

A multitude of schemes exist for time-stepping rigid body simulations. While it may be possible to turn many of these into backward schemes, there are various trade-offs. In particular, simple penalty-based methods can afford at best a minimal amount of control over the resulting simulation, as even small oscillations in position due to numerical round-off quickly grow into large bounces due to the use of inverse damping, which makes capturing either sliding or stacking behaviors difficult.

Our scheme is based on the Linear Complementarity Problem (LCP) formulations due to Baraff [1991] and Stewart and Trinkle [1997]. Before we explain how our backward scheme differs, we will first review how the original forward scheme worked. Because we used the open-source Open Dynamics Engine (ODE) [Smith 2006] as a starting point for our solver, we will highlight a few places where that software differs from earlier work.

2.1 Review of LCP formulation of frictional contact

Each contact consists of a point $\mathbf{p}_i$ and a normal $\mathbf{n}_i$ (Figure 2, left). Suppose we have a set of contact points $\mathbf{p}_1, \ldots \mathbf{p}_n$. The scalar normal velocity $v^t_{\mathbf{n}_i}$ evaluated at time t is the relative velocity of the two bodies at $\mathbf{p}_i$ along $\mathbf{n}_i$. The convention is that if the bodies are separated at $\mathbf{p}_i$ and the normal velocity $v^t_{\mathbf{n}_i}$ at each contact point is nonnegative, then the two bodies will not interpenetrate at the given contact point in the next timestep. To maintain this invariant, we apply a normal impulse $\Delta t f_{\mathbf{n}_i}\mathbf{n}_i$ at each contact point. Solving for impulses instead of forces and formulating constraints in terms of post-timestep velocities avoids the classic Painlevé problem as noted by Stewart and Trinkle [1996]; once velocities are computed they can be used to advance positions. Conveniently, if we use an Euler timestepping scheme, the relationship between the set of normal impulses $\mathbf{f} = \{f_{\mathbf{n}_1}, f_{\mathbf{n}_2} \ldots f_{\mathbf{n}_n}\}$ applied to a pair of colliding bodies and the resulting post-collision velocities $\mathbf{v}^{t+\Delta t} = \{v^{t+\Delta t}_{\mathbf{n}_1}, v^{t+\Delta t}_{\mathbf{n}_2} \ldots v^{t+\Delta t}_{\mathbf{n}_n}\}$ at each of those points is linear, and can be written as,

$$\mathbf{v}^{t+\Delta t} = \mathbf{v}^t + \Delta t \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{f} + \Delta t \mathbf{J}\mathbf{M}^{-1}\mathbf{f}_0 \qquad (2)$$

where $\mathbf{M}$ is a generalized mass matrix, $\mathbf{f}_0$ contains body forces, and $\mathbf{J}$ is the matrix transforming object velocities into contact constraint normal velocities $\mathbf{v}^t = \{v^t_{\mathbf{n}_i}\}$ (see Baraff's course notes for details [2001]). The impulses $\Delta t f_{\mathbf{n}_i}$ and velocities $v^{t+\Delta t}_{\mathbf{n}_i}$ must satisfy three conditions,

(a) bodies must not interpenetrate at the contact: $v^{t+\Delta t}_{\mathbf{n}_i} \geq 0$;

(b) contact forces can only repel and never attract: $f_{\mathbf{n}_i} \geq 0$; and

(c) contact forces are allowed to drive the post-collision velocity to 0 but not beyond, that is, $(f_{\mathbf{n}_i})(v^{t+\Delta t}_{\mathbf{n}_i}) = 0$.

This last condition corresponds to the intuition that contact forces should not add energy to the system. These three conditions combined with the linear relationship (2) form a Linear Complementarity Problem, which can be written

$$0 \leq v^{t+\Delta t}_{\mathbf{n}_i} \quad \perp \quad f_{\mathbf{n}_i} \geq 0, \qquad i = 1 \ldots N, \qquad (3)$$

where the $\perp$ symbol is used to indicate the complementarity condition, $v^{t+\Delta t}_{\mathbf{n}_i} f_{\mathbf{n}_i} = 0$. Note that Stewart's and Trinkle's original formulation [1996] posed constraint (a) in terms of positions, requiring that the bodies be fully separated at the next timestep. We prefer to use velocities (as did Anitescu and Potra [1996]) as forcibly separating bodies during backward simulation imparts a nonzero velocity which (as we will see) tends to grow without bound.

Restitution In the model described above, after only a single impact the relative velocity between objects will drop to zero. Restitution is needed for objects to bounce as they do in the real world. We follow the approach described in Stewart [1998] and use a Newtonian model for impacts instead of the Poisson model which is more common in graphics [Popović et al. 2000; Guendelman et al. 2003]. Under this model, the post-impact velocity is related to the pre-impact velocity by a constant $\alpha \in [0, 1)$ termed the *coefficient of restitution*, such that $v^{t+\Delta t}_{\mathbf{n}_i} = -\alpha v^t_{\mathbf{n}_i}$. A collision is inelastic if $\alpha = 0$; for convenience, we will call it elastic if $\alpha > 0$. Under the Stewart model, elastic collisions can be incorporated into the LCP framework by modifying the allowed range of $v^{t+\Delta t}_{\mathbf{n}_i}$,

$$0 \leq v^{t+\Delta t}_{\mathbf{n}_i} + \alpha v^t_{\mathbf{n}_i} \quad \perp \quad f_{\mathbf{n}_i} \geq 0. \qquad (4)$$

The decision of whether to apply an elastic or inelastic collision response is commonly made using a cutoff velocity $\epsilon > 0$ for $v^t_{\mathbf{n}_i}$, switching to an inelastic response whenever $|v^t_{\mathbf{n}_i}| < \epsilon$ to allow objects to come to rest and avoid spending time applying barely perceptible elastic micro-collisions to bodies.

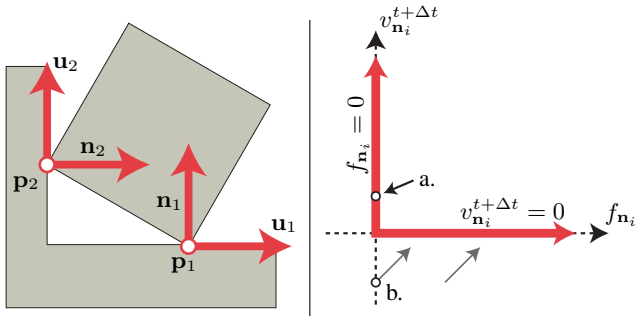


Figure 2: **Linear Complementarity Problem for the normal force: Left:** *A contact point i consists of a point $\mathbf{p}_i$ and a normal $\mathbf{n}_i$. If the normal velocity of the two objects at $\mathbf{p}_i$ is 0, then the two bodies will not penetrate. For friction we add two vectors $\mathbf{u}_i$ (above) and $\mathbf{w}_i$ (not pictured) that span the tangent plane.* **Right:** *Constraints on the acceleration and force at a contact point i correspond to values of $v_{\mathbf{n}_i}$ and $f_{\mathbf{n}_i}$ which lie on the thick red line. When solving the LCP, Lemke's algorithm begins with $f_{\mathbf{n}_i} = 0$; if $v_{\mathbf{n}_i} \geq 0$ (e.g., point* **(a)**), *we are done, but if $v_{\mathbf{n}_i} < 0$ (e.g., point* **(b)**) *then we must increase $f_{\mathbf{n}_i}$ until $v_{\mathbf{n}_i}$ moves into the valid range.*

Friction To model friction, we associate a pair of vectors $\mathbf{u}_i \perp \mathbf{n}_i$ and $\mathbf{w}_i \perp \mathbf{n}_i$ with each contact point i. These vectors lie in the tangent plane at $\mathbf{p}_i$ and oppose sliding in that plane. As in (2), the relationship between impulses $\Delta t f_{\mathbf{w}_i}\mathbf{w}_i$ applied along these vectors and changes in velocity at other contact points is linear. For friction, the constraints are slightly different,

(a) friction must always oppose velocity; that is, if $v^{t+\Delta t}_{\mathbf{w}_i} \leq 0$ then $f_{\mathbf{w}_i} \geq 0$ and vice versa,

(b) friction forces are bounded by a constant fraction μ of the normal force, $-\mu f_{\mathbf{n}_i} < |f_{\mathbf{u}_i}| + |f_{\mathbf{w}_i}| < \mu f_{\mathbf{n}_i}$, and

(c) friction must be maximal if the resulting tangential velocity is nonzero, $v^{t+\Delta t}_{\mathbf{w}_i}(\mu f_{\mathbf{n}_i} - |f_{\mathbf{w}_i}|) = 0$ (this last condition is related to the *principle of maximal dissipation* [Stewart 2000]).

For simplicity, ODE decouples the two friction directions: the single constraint in (b) above is converted to two individual constraints,

$$-\mu f_{\mathbf{n}_i} < |f_{\mathbf{u}_i}| < \mu f_{\mathbf{n}_i} \qquad -\mu f_{\mathbf{n}_i} < |f_{\mathbf{w}_i}| < \mu f_{\mathbf{n}_i} \qquad (5)$$

which approximates the classic friction cone using a pyramid. The resulting LCP constraint is

$$0 \leq \mu f_{\mathbf{n}_i} - |f_{\mathbf{u}_i}| \quad \perp \quad v^{t+\Delta t}_{\mathbf{u}_i}, \quad f_{\mathbf{n}_i} v^{t+\Delta t}_{\mathbf{u}_i} \leq 0. \qquad (6)$$

These constraints are easier to see visually and correspond to the red line in Figure 3. ODE actually computes friction forces in two passes; the first estimates normal forces for a frictionless system, and the second computes friction forces using the bounds $\mu f_{\mathbf{n}_i}$ from the first pass. In our implementation we solve for friction and normal forces simultaneously similar to Baraff [1994].

2.2 Solving the LCP Problem

One algorithm commonly used for solving LCPs is Lemke's algorithm, which is described succinctly by Baraff [1994] with more details available in Cottle, Pang and Stone [1992]. Lemke's algorithm begins with each of the $f_{\mathbf{n}_i}$, $f_{\mathbf{w}_i}$, $f_{\mathbf{u}_i}$ at zero (e.g., points (a) or (b) in Figure 2 or points (c) or (d) in Figure 3). Each step of the solver involves selecting a single $f_{\mathbf{n}_i}$, $f_{\mathbf{w}_i}$ or $f_{\mathbf{u}_i}$ and driving it toward the valid region. Let $\{f_i, v^{t+\Delta t}_i\}$ be a generic constraint (which could refer to any of the friction/normal constraints). As $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ is positive semidefinite, when we increase f_i we find that $v^{t+\Delta t}_i$

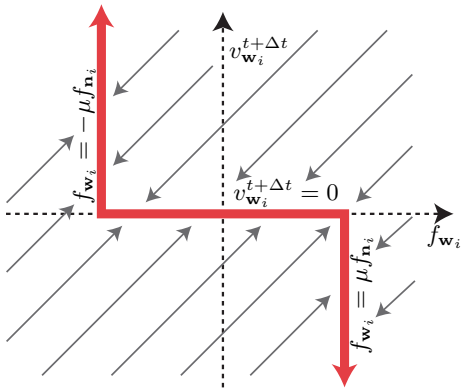


Figure 3: **Frictional LCP:** *Here, the friction force $f_{\mathbf{w}_i}$ is required to lie on the red line where the two vertical portions correspond to maximal dissipation dynamic friction and the horizontal line corresponds to static friction. The goal of the LCP solver in this case is to modify $f_{\mathbf{w}_i}$ until the point $(f_{\mathbf{w}_i}, v_{\mathbf{w}_i}^{t+\Delta t})$ lies on the red line; this corresponds to moving along the diagonal lines shown here.* (Credit: diagram inspired by one in [Smith 2006].)

increases as well, meaning that we can drive $\{f_i, v_i^{t+\Delta t}\}$ along the grey lines that can be seen in both Figure 3 and Figure 2 until we reach the valid region.

3 Backward steps

The complete state for a rigid body j at time t consists of its position $\mathbf{x}_j^t$, its orientation $\mathbf{\Theta}_j^t$, its linear velocity $\mathbf{v}_j^t$, and its angular momentum $\mathbf{L}_j^t$. In backward simulation, we are given the complete system state $\mathscr{S}_j^t = \{\mathbf{x}_j^t, \mathbf{\Theta}_j^t, \mathbf{v}_j^t, \mathbf{L}_j^t\}$ at time t and want to compute the state $\mathscr{S}_j^{t-\Delta t}$ at time $t - \Delta t$. In this section, we will consider a state $\mathscr{S}_j^{t-\Delta t}$ to be plausible if the *forward* timestepping scheme described in §2 would if given $\mathscr{S}_j^{t-\Delta t}$ as input produce $\mathscr{S}_j^t$ as output. We will examine each kind of constraint separately and break down the possible configurations at $t - \Delta t$.

3.1 Normal forces

Consider first a normal constraint i (Figure 2, left). We can easily compute the relative normal velocity $v_{\mathbf{n}_i}^t$ of the two bodies at this point. We know that $v_{\mathbf{n}_i}^t \geq 0$ (to machine precision) as this is a simulation invariant, and to prevent interpenetration we will require that $v_{\mathbf{n}_i}^{t-\Delta t} \leq 0$. Our goal is to determine a plausible value for $v_{\mathbf{n}_i}^{t-\Delta t}$. Based on the LCP formulation described in §2, there are four distinct possibilities for the state at $t - \Delta t$,

1. The relative velocity $v_{\mathbf{n}_i}^{t-\Delta t}$ was large enough that the solver chose to apply an elastic collision response: $v_{\mathbf{n}_i}^t \geq -\alpha v_{\mathbf{n}_i}^{t-\Delta t}$.
2. The relative velocity was 0 at the earlier timestep but is strictly positive in the current timestep; $v_{\mathbf{n}_i}^{t-\Delta t} = 0$ and $v_{\mathbf{n}_i}^t > 0$. In this case, we deduce that $f_{\mathbf{n}_i}^{t-\Delta t} = 0$ (from the LCP conditions).
3. There was no relative motion normal to the surfaces in either the earlier timestep or the current timestep: $v_{\mathbf{n}_i}^{t-\Delta t} = v_{\mathbf{n}_i}^t = 0$.
4. The relative velocity was strictly negative at the earlier time but is 0 in the current timestep; $v_{\mathbf{n}_i}^{t-\Delta t} < 0$ and $v_{\mathbf{n}_i}^t = 0$. In this case we cannot infer anything about $f_{\mathbf{n}_i}^{t-\Delta t}$ except that it was nonnegative; however, we can say with certainty that the prior velocity $v_{\mathbf{n}_i}^{t-\Delta t}$ was small enough in magnitude to not activate an elastic response.

The two cases with $v_{\mathbf{n}_i}^t > 0$ (1. and 2.) can easily be distinguished from the cases with $v_{\mathbf{n}_i}^t = 0$ (3. and 4.) since $v_{\mathbf{n}_i}^t$ is given as input. We can therefore break our contact handling into two paths using a small cutoff velocity $\delta \ll \epsilon$ to distinguish between the two cases, $v_{\mathbf{n}_i}^t < \delta$ and $v_{\mathbf{n}_i}^t \geq \delta$.

Elastic collisions ($v_{\mathbf{n}_i}^t > \delta$) In this case, we are required to decide whether an object's outgoing post-collision velocity resulted from (i) an elastic collision applied at that contact, or (ii) other forces in the system. This ambiguity is a stumbling block for the backward LCP solver; if we consider Figure 4 we can see the two possible lines that $\{f_{\mathbf{n}_i}^{t-\Delta t}, v_{\mathbf{n}_i}^{t-\Delta t}\}$ are allowed to lie on. Barring a fortuitous combination of body forces, the only way we will ever see both $v_{\mathbf{n}_i}^{t-\Delta t} > 0$ and $f_{\mathbf{n}_i}^{t-\Delta t} = 0$, is if some other $f_{\mathbf{n}_j}^{t-\Delta t}$ is nonzero ($j \neq i$). If we want to find cases where $f_{\mathbf{n}_i} = 0$ and $v_{\mathbf{n}_i}^{t-\Delta t}$ we will have to experiment with setting other $f_{\mathbf{n}_j}$ in the system to nonzero values, which turns our linear LCP formulation into a combinatorial optimization problem.

From another perspective, imagine we are at point (a) in Figure 4 and the LCP solver needs to decide what to do. If we try to push $f_{\mathbf{n}_i}^{t-\Delta t}$ toward 0, there is a significant risk that the slope of the line will be too steep and we will miss the valid region altogether (it is impossible to know the true slope *a priori*, since changes to the index sets will affect it). So the safe course for the solver is to simply keep $v_{\mathbf{n}_i}^{t-\Delta t}$ on the horizontal line, which it can always do by applying appropriate forces. In an LCP solver using an algorithm such as Lemke's that processes the variables one by one, variables which reach the horizontal line can never leave it, so the order we handle variables in could significantly affect the result.

For these reasons, we discard possibility 2 altogether; if we see that $v_{\mathbf{n}_i}^t \geq 0$, we assume that an elastic collision was applied, which under the Newton model means that $v_{\mathbf{n}_i}^t = -\alpha v_{\mathbf{n}_i}^{t-\Delta t}$. Because we cannot guarantee that in an arbitrary frictional system a solution exists in which each of the contacts satisfies the equality, we relax the restriction by posing it as an LCP,

$$0 \geq v_{\mathbf{n}_i}^{t-\Delta t} + v_{\mathbf{n}_i}^t/\alpha \qquad \perp \qquad f_{\mathbf{n}_i}^{t-\Delta t} \geq 0 \qquad (7)$$

We note that this backward restitution model can generate different results from forward simulation. In forward simulation, the normal velocity $v_{\mathbf{n}_i}^t$ after the collision will be *at least* as large as the Newton model predicts, that is, $v_{\mathbf{n}_i}^t \geq \alpha v_{\mathbf{n}_i}^{t-\Delta t}$. During reverse simulation, $v_{\mathbf{n}_i}^t$ is instead *at most* the predicted response, $v_{\mathbf{n}_i}^t \leq \alpha v_{\mathbf{n}_i}^{t-\Delta t}$.

Resting contact ($v_{\mathbf{n}_i}^t < \delta$) For any contact i where $v_{\mathbf{n}_i}^t \approx 0$, we know as discussed above that a forward solver could not have processed this as an elastic collision (the elastic collision model guarantees a nonzero post-collision velocity), we can treat it as if an inelastic collision happened at time $t - \Delta t$. In this case, we know that $v_{\mathbf{n}_i}^{t-\Delta t} \leq 0$ and $f_{\mathbf{n}_i}^{t-\Delta t} \geq 0$. We have one other constraint, which is that $-v_{\mathbf{n}_i}^{t-\Delta t} < \epsilon$. Unfortunately, it is impossible to guarantee that this will hold due to the influence of other $f_{\mathbf{n}_j}^{t-\Delta t}$ for $j \neq i$, as these may push $v_{\mathbf{n}_i}^{t-\Delta t}$ over the ϵ threshold. In addition, we want to allow the user control over whether the object was (subject to other forces) sitting at rest at time $t - \Delta t$ or whether one or more contacts had a nonzero velocity. We therefore allow users to select a goal velocity $v_{\mathbf{n}_i}^{goal}$, where setting $v_{\mathbf{n}_i}^{goal} = 0$ means that the solver will attempt (subject to other constraints) to maintain resting contact, while setting $0 < v_{\mathbf{n}_i}^{goal} < \epsilon$ instructs the solver to set $-v_{\mathbf{n}_i}^{t-\Delta t} \geq v_{\mathbf{n}_i}^{goal}$. The result is another LCP condition,

$$0 \geq v_{\mathbf{n}_i}^{t-\Delta t} + v_{\mathbf{n}_i}^{goal} \qquad \perp \qquad f_{\mathbf{n}_i}^{t-\Delta t} \geq 0 \qquad (8)$$

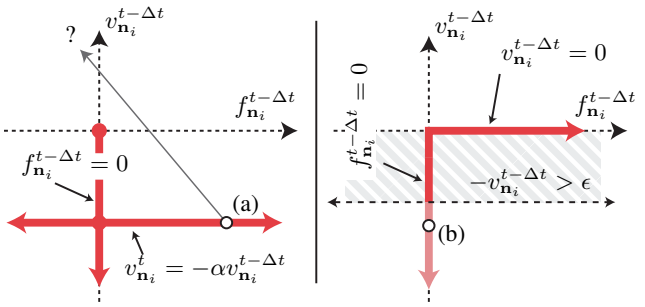


Figure 4: **LCP conditions when simulating backwards: Left:** *If we find ourselves at time t with a strictly positive $v_{\mathbf{n}_i}^t$, there are two possibilities for what could have happened at time $t - \Delta t$: either we applied an elastic collision response (corresponding to the horizontal line here) or we applied an inelastic response but nonetheless saw a nonzero post-timestep velocity $v_{\mathbf{n}_i}^t$ (corresponding to the vertical line here). Given the other forces in the system, it is trivial to determine whether a nonzero $f_{\mathbf{n}_i}^{t-\Delta t}$ is needed to explain why $v_{\mathbf{n}_i}^t > 0$; with multiple constraints, however, it is unclear how we can quickly determine* which *of the $f_{\mathbf{n}_i}^{t-\Delta t}$ lay on the vertical line and which lay on the horizontal line. (Note: Point (a) is discussed in the text.)* **Right:** *Under the restitution model we use during forward simulation, the only way that $v_{\mathbf{n}_i}^t$ can be zero is if $|v_{\mathbf{n}_i}^{t-\Delta t}| < \epsilon$. However, if we add this constraint the LCP solver will not converge in many cases, as if we are at point (b) we are not allowed to drive $f_{\mathbf{n}_i}^{t-\Delta t}$ into the negative region.*

3.2 Friction

In the model described in §2.1, a nonzero tangential velocity at time t implied that the friction force applied was maximal. That is, if $v_{\mathbf{w}_i}^t > 0$, then $|f_{\mathbf{w}_i}^{t-\Delta t}| = \mu f_{\mathbf{n}_i}^{t-\Delta t}$. On the other hand, if $v_{\mathbf{w}_i}^t = 0$, then $f_{\mathbf{w}_i}^{t-\Delta t}$ could lie anywhere inside the friction cone; this is underconstrained, as we would expect it to be, since damping forces tend to remove information from the system. While any answer returned by the solver is guaranteed to be feasible, the user can "request" a particular $v_{\mathbf{w}_i}^{t-\Delta t}$ by providing a goal velocity $v_{\mathbf{w}_i}^{goal}$; the resulting LCP is

$$0 \le \mu f_{\mathbf{n}_i}^{t-\Delta t} - |f_{\mathbf{w}_i}| \quad \perp \quad v_{\mathbf{w}_i}^{t-\Delta t} - v_{\mathbf{w}_i}^{goal} \ge 0 \tag{9}$$

$$(f_{\mathbf{w}_i})(v_{\mathbf{w}_i}^{t-\Delta t} - v_{\mathbf{w}_i}^{goal}) \le 0 \tag{10}$$

In this case, the user should also specify the friction direction $\mathbf{w}_i$, which could be in any direction. By specifying different friction directions for different contact points we get behaviors such as the object spinning in. Setting $v_{\mathbf{w}_i}^{goal} = -\infty$ or $v_{\mathbf{w}_i}^{goal} = \infty$ here is equivalent to requiring that the friction force be maximal, although in this case since we know *a priori* the relationship between the friction and normal forces we could just as easily fold friction into the system matrix similar to how Baraff handled dynamic friction [1994].

3.3 Implementation

Our timestepping scheme for backward simulation is just a mirror of the Euler scheme we use for forward timestepping; after computing the forces $\mathbf{f}_j^{t-\Delta t}$ and torques $\tau_j^{t-\Delta t}$, we advance the velocities,

$$\mathbf{v}_j^{t-\Delta t} = \mathbf{v}_j^t - \Delta t \mathbf{M}^{-1} \mathbf{f}_j^{t-\Delta t} \tag{11}$$

$$\mathbf{L}_j^{t-\Delta t} = \mathbf{L}_j^t - \Delta t \tau_j^{t-\Delta t} \tag{12}$$

which are then used to advance the positions in the normal way.

Solving for contact forces requires only a working LCP solver. Our system uses ODE's implementation of Lemke's algorithm, which we have modified to more tightly couple the friction and normal forces. The number of constraints in the system is identical to those used in forward simulation, so performance is similar. The runtime performance of Lemke's algorithm is dominated by the cost of repeated matrix solves, and if fast $O(n^2)$ matrix updates are used the total cost is roughly cubic in practice. However, like the simplex method on which it is based, Lemke's algorithm has exponential worst-case performance [Baraff 1994].

The linear relationship (2) must be changed due to the backward timestepping scheme,

$$\mathbf{v}^{t-\Delta t} = \mathbf{v}^t - \Delta t \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{f} - \Delta t \mathbf{J}\mathbf{M}^{-1}\mathbf{f}_0 \tag{13}$$

Since most LCP solvers are designed to operate in terms of positive definite rather than negative definite matrices, we can flip signs,

$$-\mathbf{v}^{t-\Delta t} = -\mathbf{v}^t + \Delta t \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{f} + \Delta t \mathbf{J}\mathbf{M}^{-1}\mathbf{f}_0 \tag{14}$$

Fortunately, our constraints are most naturally posed in terms of negative velocities, so no changes are needed to the solver to support (14). Now, we observe that the constraints for backward simulation all include a "goal velocity" term $v_{\mathbf{w}_i}^{goal}$ or $v_{\mathbf{n}_i}^{goal}$. These can be easily added to (14),

$$-(\mathbf{v}^{t-\Delta t} + \mathbf{v}^{goal}) = -\mathbf{v}^t + \Delta t \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{f} + \Delta t \mathbf{J}\mathbf{M}^{-1}\mathbf{f}_0 \tag{15}$$

Selecting these goal velocities will depend on the criteria we discussed in the previous section, which we can write in pseudocode (more details can be found in [Twigg 2008]):

HANDLE-CONTACT($\mathbf{v}_i^{t-\Delta t}, \mathbf{n}_i$)

$v_{\mathbf{n}_i}^t \leftarrow \mathbf{v}_i^t \cdot \mathbf{n}_i$ *Splitting velocity into components*
$\mathbf{w}_i \leftarrow (\mathbf{v}_i^{t-\Delta t} - v_{\mathbf{n}_i}^t \mathbf{n}_i) / \left\|\mathbf{v}_i^{t-\Delta t} - v_{\mathbf{n}_i}^t \mathbf{n}_i\right\|$
$v_{\mathbf{w}_i}^t = \mathbf{v}_i^t \cdot \mathbf{w}_i$
if $v_{\mathbf{w}_i}^t < \delta$ *Setting user's friction direction*
  then $\mathbf{w}_i \leftarrow$ user preference
switch *Normal velocity needs bounce?*
  case $v_{\mathbf{n}_i}^t > \delta$:
    $v_{\mathbf{n}_i}^{goal} \leftarrow -v_{\mathbf{n}_i}^t / \alpha$
  case $v_{\mathbf{n}_i}^t \le \delta$:
    $v_{\mathbf{n}_i}^{goal} \leftarrow 0$ or user preference
switch *Friction force needs to be maximal?*
  case $v_{\mathbf{w}_i}^t \le \delta$:
    $\mathbf{w}_i \leftarrow$ user-preferred direction
    $v_{\mathbf{w}_i}^{goal} \leftarrow$ user-preferred speed
  case $v_{\mathbf{w}_i}^t > \delta$:
    $v_{\mathbf{w}_i}^{goal} \leftarrow \infty$

4 Addressing Challenges and Limitations

One limitation of backward simulation as described here is that while it is always *possible* in a local sense to step a given simulation in the backward direction, naïvely applying to a given initial condition can generate motion that is both qualitatively and quantitatively different from motion generated by a forward simulation. It is important to note that this is *not* a numerical artifact of the solver; that is, even if we could guarantee that in all situations taking a backward step followed by a forward step (or vice versa) would bring us back exactly to where we started, our backward simulations will still be highly sensitive to the problems we list here because in general we provide the simulator with initial conditions that are not the result of a plausible forward simulation.

We will illustrate these problems with a combination of simple thought experiments and numerical results.

Rapid velocity growth: Backward simulation can lead to rapid growth in simulation energy as a result of restitution-related collisional and frictional dissipation effects when run in reverse. In practice, simulations with collisions can quickly become excited, and have useful time horizons which are short. One strategy for alleviating this problem, but not entirely removing it, is to limit the rate of energy gain by using restitution coefficients closer to one, and by not having large friction coefficients. Another possibility, although it is nonphysical, is to introduce backward damping terms to limit the rate at which simulations gain energy.

Unchecked micro-collision events can also lead to rapid energy gains for objects that can rattle between two other objects; for example, an object could be trapped between panes of glass. We can reduce the gain by increasing the restitution coefficient when we see many collisions between the same two objects in a short time period while running backward. In our system, the restitution coefficient between two objects is set to 1 after an elastic collision is processed between those objects and then smoothly blended back to 0 over a short time period (0.2 seconds).

High angular velocities: Multiple collisions can also lead to rapid increases in angular momentum. However, a more subtle and distracting issue is that angular velocity may be unnaturally increased relative to linear velocity. For example, backward simulation may lead to objects that bounce around with typical linear velocities, yet spin rapidly with high angular velocities. To address this we apply a small damping torque during backward simulation which counteracts rotational velocity. This is not a panacea, however, as many collisions can occur in a short time period while the damping force is only effective over longer time scales.

Importance of initial conditions: Consider the following simple experiment: we place a small object inside a large box that has a single exit barely large enough for the first object to fit. When we run this simulation backward, we find that the smaller box bounces around inside the larger one indefinitely while experiencing exponential energy growth due to the vanishingly small probability of finding the exit. While this is (mathematically) a valid simulation, it does not register to us as plausible because it seems unlikely that the smaller box could have started with such a large amount of kinetic energy. Certainly, low-probability events can be a problem for forward simulation as well, but the response (rapid energy growth) is qualitatively different.

Given that we know that there exist forward simulations that enter the box through the hole and settle to rest, the question becomes one of picking an initial condition for our backward simulation that when stepped backward will produce one of these forward simulations. In many cases, however, if we pick the initial conditions in a random or pseudo-random fashion we have an arbitrarily small probability of selecting an initial condition that is the result of some valid forward simulation. To see this, consider Figure 5, where it is evident that even a minimum of geometry can lead to a very nonlinear mapping from the input space to the output space.

A related problem involves entropy loss; when reverse-simulation motion is played back, we see a collection of objects go from being totally disordered to completely organized. This looks unnatural to us because in the real world (loosely speaking) entropy should increase as time advances. We can help a bit by choosing initial conditions for reverse simulation that are less than perfectly aligned.

Stacking: One example we would like to generate involves having objects bounce around and then coincidentally assemble into some kind of structure. To generate this, our simulator must be able to handle stacking behavior. Any kind of stacking immediately implies a causal order to the simulation, since in forward-simulated objects cannot fall into place until objects below them in the pile

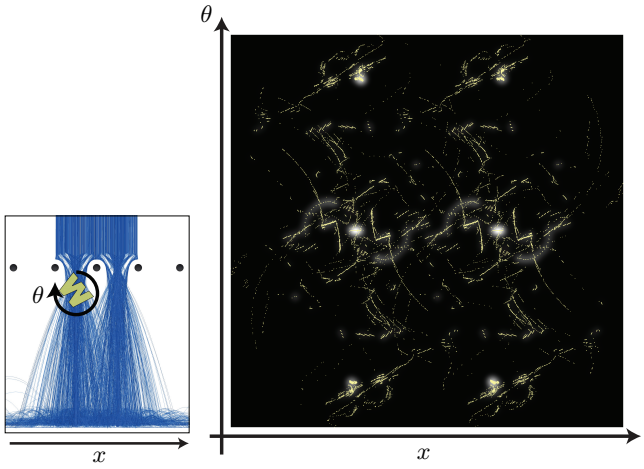


Figure 5: **Difficulty of choosing initial conditions:** *We construct a collision-constrained example* **(Left)** *that illustrates the potentially sparse set of final states that result from plausible falling motion. If we uniformly sample 10,000 initial conditions, we can compute the resulting motion under forward simulation and examine the final object configurations. To reduce quantization artifacts in the rotational component we actually look at the position and rotation when the object passes through a fixed height $y = h$.* **Right:** *These position/rotation pairs can then be plotted on a graph as shown* right *(points are in yellow and overlaid on top of a density plot). The set of dynamically achievable configurations is thus a very sparse subset of the object's configuration space. Consequently, if we randomly sample an end configuration to begin our backward simulation, the odds that we will pick a point on that sparse set, and can reach a feasible starting configuration, are vanishingly small.*

are there to fall on. The corresponding requirement when simulating backwards is that we only cause an at-rest object to slide, roll, or bounce if it is at the top of the pile. Although determining which objects should be free to bounce is a difficult problem in general, we use the simple heuristic of examining all the contact normals for a particular object; if the dot product of every pair of normals is positive, then the object is considered free to bounce, whereas if any pair of normals are opposing, the object is considered constrained. The user can override this heuristic if necessary using the sketching interface described later.

Even after we ensure causality through this heuristic, however, we note that stacks of bodies are extremely prone to flying apart under the reverse simulation (this corresponds to a forward simulation where all the bodies arrive at their precise positions in the stack simultaneously, which can limit the range of motions). The cause of this phenomenon can be seen in Figure 6; basically, to ensure that the stack remains balanced when running backward, we must anticipate collisions before they occur and ensure that the stack has the exact velocity at the time of the collision to cancel out the impulse generated by the impact. Ensuring this for every impact would necessitate a substantial amount of lookahead that would make backward simulation less practical. We take an alternative approach which is not physically valid but produces reasonable-looking motion: we allow the normal force to drop *below* 0 for objects that are constrained according to the heuristic in the previous paragraph. This corresponds to allowing stacked objects to attract each other, which prevents blocks at the bottom of the stack from transitioning from rest to bouncing.

Joints and constraints: One of the advantages of using the LCP formulation is that joints are easy to add; we simply add extra variables to $\mathbf{f}$ in (2) for the Lagrange multipliers (see Witkin's course notes for details [2001]). To generate backward simulations involv-

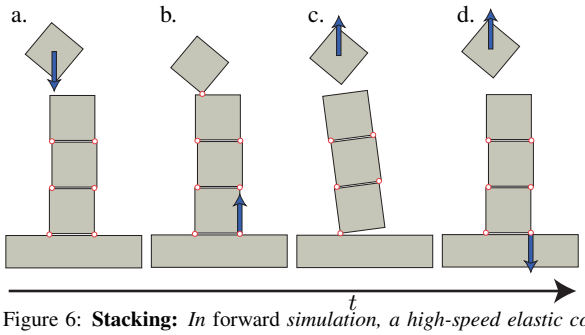


Figure 6: **Stacking:** *In* forward *simulation, a high-speed elastic collision between a single block and a stack of blocks* **(a)** *may result in a torque applied to the entire stack* **(b)** *which causes a corner of the stack to lift slightly off the ground* **(c)** *before settling down via an inelastic collision* **(d)**. *If we imagine trying to simulate this behavior* backward*, however, we notice that at step* **(d)** *before any collisions have occurred we must apply our inelastic collision to cause a nonzero velocity for the stack; failure to do so means that an opposing torque will be applied at step* **(b)** *that, unless it is somehow damped out through other low-probability events, will cause the stack to disintegrate almost immediately.*

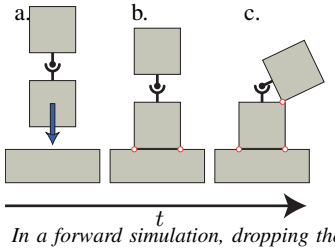


Figure 7: **Joints:** *In a forward simulation, dropping these two connected bodies results in the resting configuration* **(c)**. *If we begin our backward simulation from this resting configuration, however, we quickly run into problems; from* **(c)***, we must apply precisely the right impulse to the upper block to push it into the vertical configuration, and at that exact moment* **(b)** *the bottom block must lift off the surface. If we fail to match this exactly, there will be some residual angular motion about the joint at* **(a)** *which will quickly get multiplied through inter-block collisions.*

ing jointed and otherwise constrained bodies, this formulation holds without modification save reversing the signs on any Baumgarte stabilization terms (we want joint constraint error to decrease as the simulation steps backward, not increase).

Despite the convenience of implementation, generating *plausible* time-reversed simulations of articulated bodies remains challenging: for short periods, the generated motion is reasonable, but eventually the attached bodies begin rebounding off each other with exponentially growing velocities. This is similar to backward simulation of deforming bodies; where vibrations damp out during forward simulations, they grow without bound during reverse simulation. To understand why this problem arises in the first place, consider Figure 7; here, the only way to avoid a nonzero velocity at the joint (and hence inter-body collisions) is if we experience the low probability event that the simulation chooses to cause the resting bottom block to rise up off the surface at exactly the right moment.

One heuristic that can ameliorate the effect somewhat for articulated bodies is to set the coefficient of restitution to unity between bodies connected by a series of joints, but we recognize that this is a highly unappealing solution and hope that future work will suggest better alternatives.

Rolling: Rolling behavior is essential for generating realistic behavior for balls and cars. However, without a rolling friction model, it is impossible to generate rolling behavior during backward simulation *unless* the object is rolling without slipping in the initial conditions we provide to the solver. This is an unintuitive result, but it follows from the following two observations:

1. The principle of maximal dissipation ensures that any slipping grows without bound as the simulator steps in reverse. Suppose, for example, that our scene consists of a single sphere resting on the ground with a rotational velocity ω_i. In forward simulation, friction opposes this angular velocity, so in backward simulation it must reinforce it; the result is a sphere that spins ever faster in one direction while accelerating in the opposite.

2. In the absence of rolling friction, if an object is sitting at rest then there exists no forward simulation involving only rolling which could have concluded in that state. This means that if we provide our backward simulator with the initial condition that our sphere is sitting unmoving then it is only through the application of rolling friction that we can start the ball rolling.

This means that a working rolling friction model is an essential part of backward simulation. Ours is quite simple: we add terms to the LCP that (1) oppose rotational motion along all three axes, and (2) are proportional to the normal force. By including them in the solver itself, we guarantee that the rolling forces are proportional to the normal force, which would be difficult if we computed rolling friction before we solved for the normal forces, and that rolling friction does not cause violation of any of the other constraints, which would be difficult to ensure if rolling friction was added in after the LCP solver step.

Sliding, skidding, and bouncing: When the initial conditions for the backward simulation specify that the object is at rest, we have an enormous amount of leeway in what the object should do. As described in §3, we can provide hints to the simulator on a per-contact basis specifying that the object begin sliding or that a small nonzero normal impulse be applied.

One possibility is to give the user direct control over these possibilities. In our interactive system, the user can use gestures to distinguish between sliding and bouncing (see Figure 8).

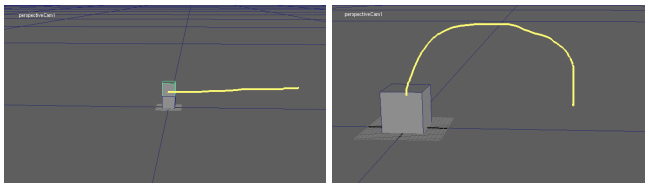


Figure 8: **User-specified motion suggestions** *can provide hints to the backward integrator to use when the resulting motion may be nonunique. An arced line indicates bouncing while a straight line indicates sliding with the direction of the line hinting the sliding direction. We use a simple method to distinguish between the two possibilities by fitting a 2nd-degree polynomial to the sketched curve and comparing the quadratic coefficient to a threshold.*

If the user fails to provide any particular direction, the simulator itself must make these decisions, preferrably in a suitably random fashion. In developing this, our goal is to generate motion that is as similar as possible to motion generated using a forward simulation. To investigate this quantitatively, we generated over 1000 forward simulations of a block landing on a plane, randomizing both the initial angular and linear velocities. We recorded the contact state at each step of the simulation. In Figure 9, we have plotted the number of contacts recorded in each frame of the simulation (only a subset of the simulations are shown here). Note that this data depends on both the collision geometry and the sampling method used to generate initial states; for more general geometries this could easily be run as a pre-processing step as single-body simulations can be computed quite quickly.

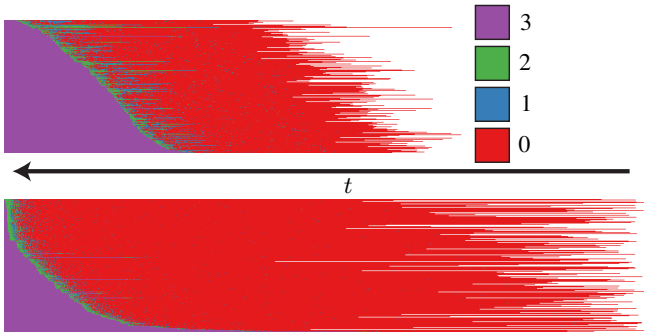


Figure 9: **Contact state transitions: Top:** *Contact state in rigid body simulation is complex, as we see transitions between* 0 *(free-flight),* 1*,* 2*, and* 3 *(flat on a face) contacts for a box thrown against a plane. Time runs from right to left, so at the far left of the plot the box is at rest; the simulations have been sorted lexicographically by contact configuration for easy viewing.* **Bottom:** *200 simulations computed using backward simulation; our goal is to get* qualitatively *similar simulations.*

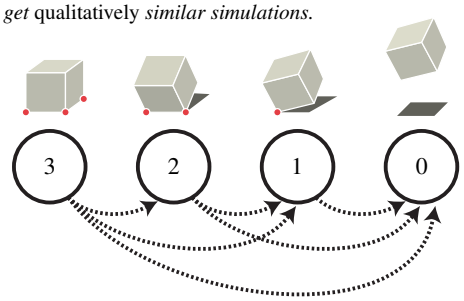


Figure 10: **Contact transitions:** *We use a simple model for determining when to transition between contact states during backward simulation. Because we can only apply impulses out of the plane, we can only transition to states with fewer contacts, as seen here. Probabilities for transitioning between contact states are computed from the data in Figure 9.*

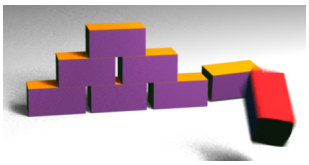
Ideally, we would like our backward simulation to generate a plot that looks as much like this as possible for simulated motions. To this end, we developed the simple state-space model that can be seen in Figure 10. We can use the $v_{\mathbf{n}_i}^{goal}$ parameter to induce a separation at a given contact point; however, we cannot force an already separating contact to "stick," so our model only includes transitions from more contacts to fewer contacts. We propose a simple Markov model; if we let η^t be the number of contacts at time t, then the probability $\mathbb{P}(\eta^t \mid \eta^{t-\Delta t})$ of transitioning from η^t contacts to $\eta^{t-\Delta t}$ is independent of all the other states $\eta^{t+\Delta t}, \eta^{t+2\Delta t}, \ldots$ (this is the Markov condition). These state transition probabilities are computed directly from the data in Figure 10. At runtime, we count the number of active contacts at each timestep and use a random-number generator to decide whether to transition to a different contact state; if so, an additional step selects randomly which of the contacts should be separated, and the appropriate $v_{\mathbf{n}_i}^{goal}$ is set. The resulting contact behavior can be seen in Figure 9. While there are significant differences between the plots, if we examine the resulting motion it seems plausible; see our video results.

5 Results

To test the usefulness and scalability of our approach, we applied it to several different examples.

Spelling balls: Chenney and Forsyth [2000] showed several examples of dropping balls into grid cells to spell out words. This is a nearly ideal example for our approach, since we care much more about the resulting shape than we do about exactly where the balls come from. The tradeoff we make here is that we have very little control over where the balls fall from, while Chenney and Forsyth incorporated this into their prior. In our version we dropped 3037 balls into square bins to spell out the classic typesetting dummy text "Lorem ipsum dolor sit amet…" (see Figure 1). Although our example has 100 times as many balls as the ACM example, it runs in just under an hour on our test machine, which has a 2.66GHz Intel Core 2 processor and 2GB of RAM. Note that in this example good modelling of rolling and sliding is essential to getting realistic behavior toward the end of the simulation.

Spelling boxes: Since we are running the simulation backward, we can make the final state as unrealistic as we want. In this case, we spell "$t \rightarrow -t$" without using bins to catch the objects (see Figure 11). Note that there is no requirement that the objects' motion be independent; in fact, many of the objects interact extensively. This example was computed in under a minute and exhibits some of the angular velocity growth noted in §4; we apply a reverse damping force to tame velocity growth somewhat.



Stacking: We built this stack out of bricks. This example is particularly prone to the microcollisions/rattling problem described in §4 due to the proximity of the bricks as they settle into the stack. As a result, many of the simulations were discarded due to rapid energy growth. Sorting the resulting simulations by energy, however, we could quickly find relatively low-energy simulations for rendering, and several are shown in the accompanying video.

Collision scenario design: Backward simulation can also be useful for specifying intermediate frames of a simulation. For example, the collision scenario shown in Figure 12 was simulated by using the intermediate frame as initial conditions for both forward and backward rigid body simulators. The objects appear to fly in, collide and bounce up, spelling "CRASH," then fall.

6 Conclusion

While backward simulation has long been considered a potential tool in our animation toolbox, there has been little in-depth examination of its implications. We have highlighted a number of the issues in this paper and provided partial solutions, but we hope that continued investigation will solve these problems outright. Our method is particularly useful for specifying final (or intermediate) states, and running them backward in time, and has computational complexity similar to traditional LCP solvers for forward-time rigid body simulation.

Notwithstanding numerous issues and limitations (§4), many interesting challenges remain. There are numerous behaviors that are not handled by our model such as stick-slip effects. Deformation would be an additional challenge, as vibrations will tend to diverge to infinity. Given the ill-posedness of backward simulation, there appears to be significant potential for user-guided motion planning and optimization. Future work should also investigate strategies for generating more natural motion. Combining forward and backward simulation techniques might leverage the strengths of both methods. Bidirectional simulation could be useful in motion optimization and sampling for boundary value problems. We expect that more sophisticated backward integration strategies might better exploit the naturalness of forward simulation to explore reverse times, similar to work on manifold-based integration [Gear and Kevrekidis 2004], but in the presence of complex frictional collisions and contact, and animator concerns.

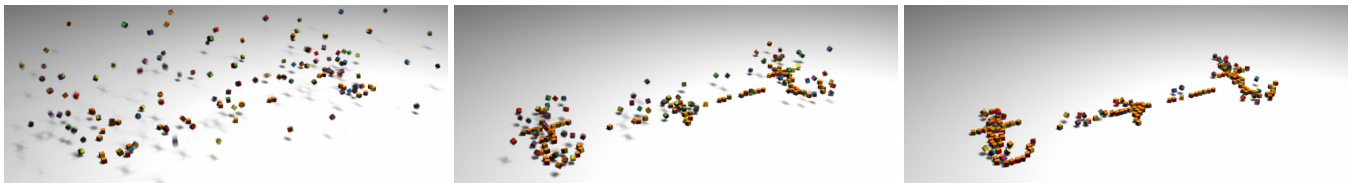


Figure 11: **Time-reversal spelled backwards**

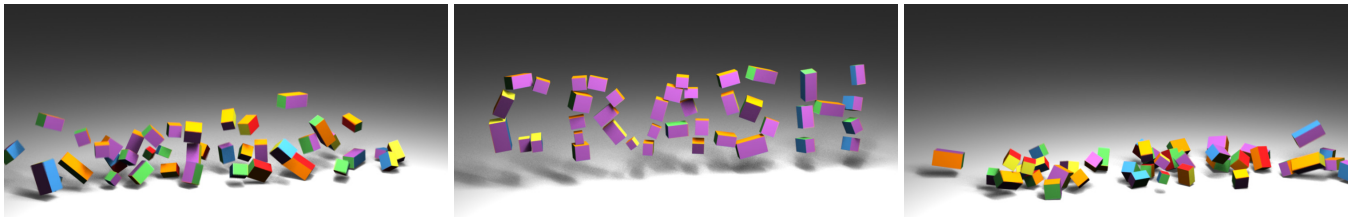


Figure 12: **Collision scenario** *designed by time-stepping forwards and backwards from one intermediate frame.*

Acknowledgements The authors would like to thank the anonymous reviewers for their many helpful comments. This work was sponsored in part by a NVIDIA Graduate Fellowship, National Science Foundation Grant CAREER-0430528, Pixar, the Alfred P. Sloan Foundation, hardware donations by NVIDIA and Intel, and Maya licenses donated by Autodesk.

References

ANITESCU, M., AND POTRA, F. A. 1996. Formulating dynamic multi-rigid-body contact problems with friction as solvable Linear Complementarity Problems. *Nonlinear Dynamics 14*, 3 (Oct.), 231–247.

BARAFF, D. 1991. Coping with friction for non-penetrating rigid body simulation. In *Computer Graphics (Proc. SIGGRAPH 91)*, 31–40.

BARAFF, D. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *Proc. of ACM SIGGRAPH 1994*, 23–34.

BARAFF, D. 2001. Rigid body simulation. In *Physically Based Modeling: SIGGRAPH 2001 Course 25*.

CHENNEY, S., AND FORSYTH, D. A. 2000. Sampling plausible solutions to multi-body constraint problems. In *Proc. of ACM SIGGRAPH 2000*, 219–228.

COTTLE, R. W., PANG, J.-S., AND STONE, R. E. 1992. *The Linear Complementarity Problem*. Academic Press.

GEAR, C. W., AND KEVREKIDIS, I. G. 2004. Computing in the past with forward integration. *Physics Letters A 321*, 5, 335–342.

GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. P. 2003. Nonconvex rigid bodies with stacking. *ACM Trans. on Graphics 22*, 3, 871–878.

HAHN, J. K. 1988. Realistic animation of rigid bodies. In *Computer Graphics (Proc. of SIGGRAPH 88)*, 299–308.

HUNT, K., AND CROSSLEY, F. 1975. Coefficient of restitution interpreted as damping in vibroimpact. *Trans. ASME, Journal of Applied Mechanics*, 440–445.

KAUFMAN, D. M., EDMUNDS, T., AND PAI, D. K. 2005. Fast frictional dynamics for rigid bodies. *ACM Trans. on Graphics 24*, 3 (Aug.), 946–956.

KHAREVYCH, L., YANG, W., TONG, Y., KANSO, E., MARSDEN, J. E., SCHRÖDER, P., AND DESBRUN, M. 2006. Geometric, variational integrators for computer animation. In *2006 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 43–52.

LANKARANI, H. M., AND NIKRAVESH, P. E. 1994. Continuous contact force models for impact analysis in multibody systems. *Nonlinear Dynamics*, 193–207.

LEIMKUHLER, B., AND REICH, S. 2005. *Simulating Hamiltonian Dynamics*. Cambridge University Press.

MCNAMARA, A., TREUILLE, A., POPOVIĆ, Z., AND STAM, J. 2004. Fluid control using the adjoint method. *ACM Trans. on Graphics 23*, 3 (Aug.), 449–456.

MILENKOVIC, V. J., AND SCHMIDL, H. 2001. Optimization-based animation. In *Proc. of ACM SIGGRAPH 2001*, 37–46.

MIRTICH, B., AND CANNY, J. 1995. Impulse-based simulation of rigid bodies. In *1995 Symposium on Interactive 3D Graphics*, 181–188.

MIRTICH, B. 2000. Timewarp rigid body simulation. In *Proc. of ACM SIGGRAPH 2000*, 193–200.

POPOVIĆ, J., SEITZ, S. M., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. P. 2000. Interactive manipulation of rigid body simulations. In *Proc. of ACM SIGGRAPH 2000*, 209–218.

POPOVIĆ, J., SEITZ, S. M., AND ERDMANN, M. 2003. Motion sketching for control of rigid-body simulations. *ACM Trans. on Graphics 22*, 4 (Oct.), 1034–1054.

REICHENBACH, H. 1999. *The Direction of Time*. Dover Publications.

SMITH, R. 2006. *Open Dynamics Engine v0.5 Users Guide*, Feb.

STEWART, D. E., AND TRINKLE, J. C. 1996. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *Intl. J. Num. Meth. Eng. 39*, 2673–2691.

STEWART, D. E., AND TRINKLE, J. C. 1997. Dynamics, friction, and complementarity problems. In *Complementarity and Variational Problems*, M. C. Ferris and J. S. Pang, Eds. SIAM, 425–439.

STEWART, D. 1998. Convergence of a time-stepping scheme for rigid body dynamics and resolution of Painlevé's problems. *Archive Rational Mechanics and Analysis 145*, 3, 215–260.

STEWART, D. E. 2000. Rigid body dynamics with friction and impact. *SIAM Review 42*, 1, 3–39.

TWIGG, C. D., AND JAMES, D. L. 2007. Many-worlds browsing for control of multibody dynamics. *ACM Trans. on Graphics (SIGGRAPH 2007) 26*, 3 (Aug.).

TWIGG, C. D. 2008. *Controlling Rigid Multibody Dynamics via Browsing and Time Reversal*. PhD thesis, Carnegie Mellon University.

WEINSTEIN, R., TERAN, J., AND FEDKIW, R. 2006. Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE Trans. on Visualization and Computer Graphics 12*, 3 (May), 365–374.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *Computer Graphics (Proc. of SIGGRAPH 88)*, 159–168.

WITKIN, A. P. 2001. Constrained dynamics. In *Physically Based Modeling: SIGGRAPH 2001 Course 25*.

WOJTAN, C., MUCHA, P. J., AND TURK, G. 2006. Keyframe control of complex particle systems using the adjoint method. In *Proc. 2006 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 15–23.