Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2014)
Vladlen Koltun and Eftychios Sifakis (Editors)

Optimization Integrator for Large Time Steps

T. F. Gast† C. Schroeder‡

Department of Mathematics
University of California, Los Angeles

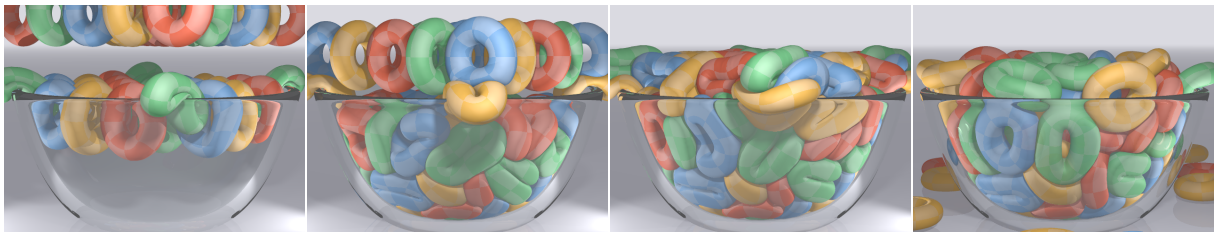

Figure 1: *125 tori are dropped into a bowl at 5 time steps per frame, resulting in significant deformation and tough collisions.*

Abstract

Practical time steps in today's state-of-the-art simulators typically rely on Newton's method to solve large systems of nonlinear equations. In practice, this works well for small time steps but is unreliable at large time steps at or near the frame rate, particularly for difficult or stiff simulations. We show that recasting backward Euler as a minimization problem allows Newton's method to be stabilized by standard optimization techniques with some novel improvements of our own. The resulting solver is capable of solving even the toughest simulations at the 24 Hz frame rate and beyond. We show how simple collisions can be incorporated directly into the solver through constrained minimization without sacrificing efficiency. We also present novel penalty collision formulations for self collisions and collisions against scripted bodies designed for the unique demands of this solver.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

The most commonly used time integration schemes in use today for graphics applications are implicit methods. Among these, backward Euler [BW98, HFL*01, VMT01, MTGG11, LBOK13] or variants on Newmark methods [Kan99, BFA02, BMF03] are the most common, though even more sophisticated schemes like BDF-2 [HE01, CK05], implicit-explicit schemes [EEH00, SG09], or even the more exotic exponential integrators [MSW13] have received consideration. Integrators have been the subject of comparison before (see for example [HE01, VMT01, PF02]), seeking good compromises between speed, accuracy, robustness, and dynamic behavior.

These integrators require the solution to one or more nonlinear systems of equations each time step. These systems are typically solved by some variation on Newton's method. Even the most stable simulators are typically run several time steps per 24 Hz frame of simulation. There is growing interest in running simulations at larger time steps [SSF13], so that the selection of Δt can be made based on other factors, such as damping or runtime, and not only on whether the simulator works at all. One of the major factors that limits time step sizes is the inability of Newton's method to converge reliably at large time steps (See Figures 3, 4, and 7), or if a fixed number of Newton iterations are taken, the stability of the resulting simulation. We address this by formulating our nonlinear system of equations as a minimization problem, which we demonstrate can be solved more robustly. The idea that dynamics, energy, and minimization are related has been known since antiquity and is commonly leveraged in variational integrators [STW92, KMO99, Kan99, LMOW04, KYT*06b, SG09, GSO10]. The idea that the nonlinear system that occurs from methods like backward Euler can be formulated as a minimization

† e-mail: tfg@math.ucla.edu
‡ e-mail: craig@math.ucla.edu

DOI: 10.2312/sca.20141120

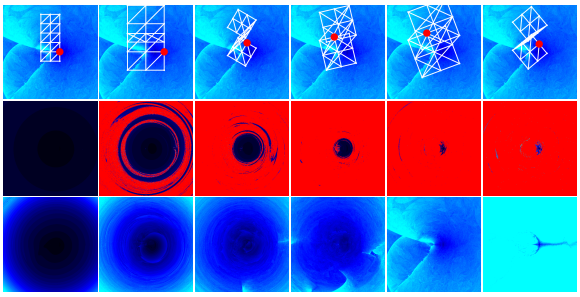

Figure 2: *Convergence of Newton's method (middle) and our stabilized optimization formulation (bottom) for a simple 36-dof simulation in 2D. The initial configuration (top) is parameterized in terms of a pixel location, with the rest configuration occurring at* $(\frac{3}{5}, \frac{1}{2})$*. Initial velocity is zero, and one time step is attempted. Time steps are (left to right) 170, 40, 20, 10, and 1 steps per* $24\,Hz$ *frame, with the rightmost image being* $\Delta t = 1\,s$*. Color indicates convergence in 0 iterations (black), 15 iterations (blue), 30 or more iterations (cyan), or failure to converge in 500 iterations (red). Note that Newton's method tends to converge rapidly or not at all, depending strongly on problem difficulty and initial guess.*

problem has appeared many times in graphics in various forms [HFL*01, KYT*06b, MTGG11, LBOK13, MSW13]. [KYT*06b] point out that minimization leads to a method that is both simpler and faster than the equivalent nonlinear root-finding problem, and [LBOK13] show that a minimization formulation can be used to solve mass-spring systems more efficiently. [KMO99] use a minimization formulation as a means of ensuring that a solution to their nonlinear system can be found assuming one exists. [GHF*07] shows that a minimization formulation can be used to enforce constraints robustly and efficiently. [HFL*01] shows that supplementing Newton's method with a line search greatly improves robustness. [MTGG11] also shows that supplementing Newton's method with a line search and a definiteness correction leads to a robust solution procedure. Following their example, we show that recasting the solution of the nonlinear systems that result from implicit time integration schemes as a nonlinear optimization problem results in substantial robustness improvements. We also show that additional improvements can be realized by incorporating additional techniques like Wolfe condition line searches which curve around collision bodies, conjugate gradient with early termination on indefiniteness, and choosing conjugate gradient tolerances based on the current degree of convergence.

2. Time Integration

The equations of motion for simulating solids are $\dot{\boldsymbol{x}} = \boldsymbol{v}$ and $\boldsymbol{M}\dot{\boldsymbol{v}} = \boldsymbol{f}$, where $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{v})$ are forces. As is common in graphics we assume $\boldsymbol{M}$ is a diagonal lumped-mass matrix. Since we are interested in robustness and large time steps, we follow a backward Euler discretization. This leads to $\frac{\boldsymbol{x}^{n+1}-\boldsymbol{x}^n}{\Delta t} = \boldsymbol{v}^{n+1}$ and $\boldsymbol{M}\frac{\boldsymbol{v}^{n+1}-\boldsymbol{v}^n}{\Delta t} = \boldsymbol{f}^{n+1} = \boldsymbol{f}(\boldsymbol{x}^{n+1}, \boldsymbol{v}^{n+1})$. Eliminating $\boldsymbol{v}^{n+1}$ using the first equation yields $\boldsymbol{M}\frac{\boldsymbol{x}^{n+1}-\boldsymbol{x}^n-\Delta t\boldsymbol{v}^n}{\Delta t^2} = \boldsymbol{f}(\boldsymbol{x}^{n+1}, \frac{\boldsymbol{x}^{n+1}-\boldsymbol{x}^n}{\Delta t})$, which is a nonlinear system of equations in the unknown positions $\boldsymbol{x}^{n+1}$. This system of nonlinear equations is normally solved with Newton's method. If we define $\boldsymbol{h}(\boldsymbol{x}^{n+1}) = \boldsymbol{M}\frac{\boldsymbol{x}^{n+1}-\boldsymbol{x}^n-\Delta t\boldsymbol{v}^n}{\Delta t^2} - \boldsymbol{f}(\boldsymbol{x}^{n+1}, \frac{\boldsymbol{x}^{n+1}-\boldsymbol{x}^n}{\Delta t})$, then our nonlinear problem is one of finding a solution to $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}$. To do this, one would start with an initial guess $\boldsymbol{x}^{(0)}$, such as the value predicted by forward Euler. This estimate is then iteratively improved using the update rule $\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} - \left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}(\boldsymbol{x}^{(i)})\right)^{-1}\boldsymbol{h}(\boldsymbol{x}^{(i)})$. Each step requires the solution of a linear system, which is usually symmetric and positive definite and solved with a Krylov solver such as conjugate gradient or MINRES.

Figure 3: *Cube being stretched: initial configuration (left), our method at* $t = 0.4\,s$ *and* $t = 3.0\,s$ *(middle), and standard Newton's method at* $t = 0.4\,s$ *and* $t = 3.0\,s$ *(right). Both simulations were run with one time step per* $24\,Hz$ *frame. Newton's method requires three time steps per frame to converge on this simple example.*

If the function $\boldsymbol{h}(\boldsymbol{x})$ is well-behaved and the initial guess sufficiently close to the solution, Newton's method will converge very rapidly (quadratically). If the initial guess is not close enough, Newton's method may converge slowly or not at all. For small enough time steps, the forward and backward Euler time steps will be very similar (they differ by $O(\Delta t^2)$), so a good initial guess is available. For large time steps, forward Euler will be unstable, so it will not provide a good initial guess. Further, as the time step grows larger, Newton's method may become more sensitive to the initial guess (see Figure 2). The result is that Newton's method will often fail to converge if the time step is too large. Figures 3, 4, and 7 show examples of simulations that ought to be routine but where Newton fails to converge at $\Delta t = 1/24\,s$.

Sometimes, only one, or a small fixed number, of Newton steps are taken rather than trying to solve the nonlinear equation to a tolerance. The idea is that a small number of Newton steps is sufficient to get most of the benefit from doing an implicit method while limiting its cost. Indeed, even a single Newton step with backward Euler can allow time steps orders of magnitude higher than explicit methods. Linearizing the problem only goes so far, though, and even these solvers tend to have time step restrictions for tough problems.

2.1. Minimization problem

The solution to making Newton's method converge reliably is to recast the equation solving problem as an optimization problem, for which robust and efficient methods exist. In principle, that can always be done, since solving $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}$

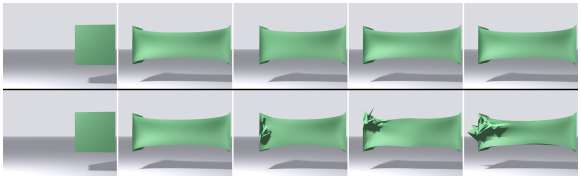

Figure 4: *Cube being stretched and then given a small compressive pulse, shown with our method (top) and standard Newton's method (bottom). Both simulations were run with one time step per 24 Hz frame. In this simulation, Newton's method is able to converge during the stretch phase, but a simple pulse of compression, as would normally occur due to a collision, causes it to fail to converge and never recover. Newton's method requires five time steps per frame to converge on this simple example.*

is equivalent to minimizing $\|\boldsymbol{h}(\boldsymbol{x})\|$ assuming a solution exists. This approach is not very convenient, though, since it requires a global minimum of $\|\boldsymbol{h}(\boldsymbol{x})\|$. Further minimization using Newton's method would require the Hessian of $\|\boldsymbol{h}(\boldsymbol{x})\|$, which involves the second derivatives of our forces. The standard approach only requires first derivatives. What we really want is a quantity E that we can minimize whose second derivatives only require the first derivatives of our forces. That is, we need to *integrate* our system of nonlinear equations $\boldsymbol{h}(\boldsymbol{x})$. It turns out that if we require our forces to come from an energy we can do this. This way of recasting the problem also requires only a local minimum be found. Most forces with symmetric force derivatives can be put into this model. We will show later how damping can also be incorporated into this model. Friction can be given an approximate potential which is valid for small Δt, see [PKMO02]. Since our examples focus on taking larger time steps we incorporate friction explicitly after the Newton solve.

Let Φ be the total potential energy of our internal forces (gravity is a special case, which we will address later). The potential Φ has a global minimum, which is typically its rest configuration. Then, we can write $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{M}\frac{\boldsymbol{x}-\boldsymbol{x}^n-\Delta t\boldsymbol{v}^n}{\Delta t^2} + \frac{\partial\Phi}{\partial\boldsymbol{x}}$. We need to express this as the gradient of some scalar function $E(\boldsymbol{x})$. Letting $\hat{\boldsymbol{x}} = \boldsymbol{x}^n + \Delta t\boldsymbol{v}^n$, we have $E(\boldsymbol{x}) = \frac{1}{2\Delta t^2}(\boldsymbol{x}-\hat{\boldsymbol{x}})^T\boldsymbol{M}(\boldsymbol{x}-\hat{\boldsymbol{x}}) + \Phi$. Then, $\boldsymbol{h} = \frac{\partial E}{\partial\boldsymbol{x}}$ as desired. Since $\boldsymbol{M}$ is symmetric and positive definite (or at least semidefinite if scripted objects are permitted), the first term is bounded from below by zero. Since Φ is also bounded from below, E is as well. Thus, E has a global minimum. If $E(\boldsymbol{x}^{n+1})$ is smooth at its minima, then this minimum will satisfy $\frac{\partial E}{\partial\boldsymbol{x}}(\boldsymbol{x}^{n+1}) = \boldsymbol{0}$ or equivalently $\boldsymbol{h}(\boldsymbol{x}^{n+1}) = \boldsymbol{0}$. Note that, although we are now doing minimization rather than root finding, we are still solving the exact same equations. The discretization and dynamics will be the same, but the solver will be more robust.

Gravity A graphics simulation would not be very useful without gravity. Gravity has the potential energy function $-\boldsymbol{M}\boldsymbol{g}^T\boldsymbol{x}$, where $\boldsymbol{g}$ is the gravitational acceleration vector, but this function is not bounded. An object can fall arbitrarily far and liberate a limitless supply of energy, though in practice this fall will be stopped by the ground or some other object. Adding the gravity force to our nonlinear system yields $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{M}\frac{\boldsymbol{x}-\boldsymbol{x}^n-\Delta t\boldsymbol{v}^n}{\Delta t^2} - \boldsymbol{M}\boldsymbol{g} + \frac{\partial\Phi}{\partial\boldsymbol{x}}$, which can be obtained from the bounded minimization objective $E(\boldsymbol{x}) = \frac{1}{2\Delta t^2}(\boldsymbol{x}-\hat{\boldsymbol{x}}-\Delta t^2\boldsymbol{g})^T\boldsymbol{M}(\boldsymbol{x}-\hat{\boldsymbol{x}}-\Delta t^2\boldsymbol{g}) + \Phi$. A more convenient choice of E, and the one we use in practice, is obtained by simply adding the effects of gravity $\Phi_g = -\boldsymbol{M}\boldsymbol{g}^T\boldsymbol{x}$ into Φ. Since all choices E will differ by a constant shift, this more convenient minimization objective will also be bounded from below.

3. Minimization

The heart of our simulator is our algorithm for solving optimization problems, which we derived primarily from [NW06], though most of the techniques we apply are well-known. We begin be describing our method as it applies to unconstrained minimization and then show how to modify it to handle the constrained case.

3.1. Unconstrained minimization

Our optimization routine begins with an initial guess, $\boldsymbol{x}^{(0)}$. Each iteration consists of the following steps:

1. * Register active set
2. Compute gradient ∇E and Hessian $\boldsymbol{H}$ of E at $\boldsymbol{x}^{(i)}$
3. Terminate successfully if $\|\nabla E\| < \tau$
4. Compute Newton step $\Delta\boldsymbol{x} = -\boldsymbol{H}^{-1}\nabla E$
5. Make sure $\Delta\boldsymbol{x}$ is a downhill direction
6. Clamp the magnitude of $\Delta\boldsymbol{x}$ to ℓ if $\|\Delta\boldsymbol{x}\| > \ell$
7. Choose step size α in direction $\Delta\boldsymbol{x}$ using a line search
8. Take the step: $\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \alpha\Delta\boldsymbol{x}$
9. * Project $\boldsymbol{x}^{(i+1)}$

Here, τ is the termination criterion, which controls how accurately the system must by solved. The length clamp ℓ guards against the possibility of the Newton step being enormous (if $\|\Delta\boldsymbol{x}\| = 10^{100}$, computing $\Phi(\boldsymbol{x}^{(i)} + \Delta\boldsymbol{x})$ is unlikely to work well). Its value should be very large. Our line search is capable of choosing $\alpha > 1$, so the algorithm is very insensitive with respect to the choice ℓ. We normally use $\ell = 10^3$. Steps beginning with * are only performed for constrained optimization and will be discussed later. A few of the remaining steps require further elaboration here.

Linear solver considerations Computing the Newton step requires solving a symmetric linear system. The obvious candidate solver for this is MINRES that can handle indefinite systems, and indeed this will work. However, there are many tradeoffs to be made here. In contrast to a normal Newton solve, an accurate estimate for $\Delta\boldsymbol{x}$ is not necessary for convergence. Indeed, we would still converge with high probability if we chose $\Delta\boldsymbol{x}$ to be a random vector. The point of using the Newton direction is that convergence will typically be much more rapid, particularly when

the superconvergence of Newton's method kicks in. (Choosing $\Delta \boldsymbol{x} = -\nabla E$ leads to gradient descent, for example, which can display notoriously poor convergence rates.) When the current estimate is far from the solution, the exact Newton direction tends to be little better than a very approximate one. Thus, the idea is to spend little time on computing $\Delta \boldsymbol{x}$ when $\|\nabla E\|$ is large and more time when it is small. We do this by solving the system to a relative tolerance of $\min(\frac{1}{2}, \sigma\sqrt{\max(\|\nabla E\|, \tau)})$. The $\frac{1}{2}$ ensures that we always reduce the residual by at least a constant factor, which guarantees convergence. The scale σ adjusts for the fact that ∇E is not unitless (we usually use $\sigma = 1$). If our initial guess is naive, we must make sure we take at least one minimization iteration, even if ∇E is very small. Using τ here ensures that we do not waste time solving to a tiny tolerance in this case.

Conjugate gradient One further optimization is to use conjugate gradient as the solver with a zero initial guess. If indefiniteness is encountered during the conjugate gradient solve, return the last iterate computed. If this occurs on the first step, return the right hand side. If this is done, $\Delta \boldsymbol{x}$ is guaranteed to be a downhill direction, though it might not be sufficiently downhill for our purposes. In practice, indefiniteness will only occur if far from converged, in which case little time is wasted in computing an accurate $\Delta \boldsymbol{x}$ that is unlikely to be very useful anyway. Indeed, if the system is detectably indefinite and $\Delta \boldsymbol{x}$ is computed exactly, it might not even point downhill. Since we are searching for a minimum of E (even a local one), the Hessian of E will be symmetric and positive definite near this solution. (Technically, it need only be positive semidefinite, but in practice this is of little consequence.) Thus, when we are close enough to the solution for an accurate Newton step to be useful, conjugate gradient will suffice to compute it. This is very different from the normal situation, where a solver like MINRES or an indefiniteness correction are employed to deal with the possibility of indefiniteness. In the case of our solver, neither strategy is necessary, and both make the algorithm slower.

Downhill direction Making sure $\Delta \boldsymbol{x}$ points downhill is fairly straightforward. If $\Delta \boldsymbol{x} \cdot \nabla E < -\kappa \|\Delta \boldsymbol{x}\| \|\nabla E\|$, then we consider $\Delta \boldsymbol{x}$ to be suitable. Otherwise, if $-\Delta \boldsymbol{x}$ is suitable, use it instead. If neither $\Delta \boldsymbol{x}$ nor $-\Delta \boldsymbol{x}$ are suitable, then we use the gradient descent direction $-\nabla E$. Note that if the conjugate gradient strategy is used for computing the Newton direction, then $-\Delta \boldsymbol{x}$ will never be chosen as the search direction at this stage. We have found $\kappa = 10^{-2}$ to work well.

Line search For our line search procedure, we use an algorithm for computing α such that the strong Wolfe Conditions are satisfied. See [NW06] for details. The line search procedure guarantees that E never increases from one iteration to the next and that, provided certain conditions are met, sufficient progress is always made. One important attribute of this line search algorithm is that it first checks to see if $\Delta \boldsymbol{x}$ itself is a suitable step. In this way, the line search is almost entirely avoided when Newton is converging properly.

Initial guess A good initial guess is important for efficient simulation under normal circumstances. Under low-Δt or low-stress conditions, a good initial guess is obtained by replacing $\boldsymbol{f}^{n+1}$ by $\boldsymbol{f}^n$ resulting in $\boldsymbol{M}\frac{\boldsymbol{x}^{n+1} - \boldsymbol{x}^n - \Delta t \boldsymbol{v}^n}{\Delta t^2} = \boldsymbol{f}(\boldsymbol{x}^n)$. Solving for $\boldsymbol{x}^{n+1}$ yields the initial guess $\boldsymbol{x}^{(0)} = \boldsymbol{x}^n + \Delta t \boldsymbol{v}^n + \Delta t^2 \boldsymbol{f}(\boldsymbol{x}^n)$. This initial guess is particularly effective under free fall, since here the initial guess is correct and no Newton iterations are required. On the other hand, this initial guess is the result of an explicit method, which will be unstable at large time steps or high stress. Under these conditions, this is unlikely to be a good initial guess and may in fact be very far from the solution. Under these situations, a better initial guess is obtained from $\boldsymbol{x}^{(0)} = \boldsymbol{x}^n + \Delta t \boldsymbol{v}^n$. In practice, we compute both initial guesses and choose the one which produces the smaller value of E. This way, we get competitive performance under easy circumstances and rugged reliability under tough circumstances.

3.2. Constrained minimization

We use constrained minimization for some of our collisions, which may result in a large active set of constraints, such as when an ball is bouncing on the ground. As the ball rises, constraints become deactivated. As the ball hits the ground, more constraints become activated. The change in the number of active constraints from iteration to iteration may be quite significant. This would render a traditional active set method impractical, since constraints are activated or deactivated one at a time. Instead, we use the gradient-projection method as our starting point, since it allows the number of active constraints to change quickly. The downside to this choice is that its reliance on the ability to efficiently project to the feasible region limits its applicability to simple collision objects.

Projections Let $P(\boldsymbol{x})$ be the projection that applies P_{bp} to $\boldsymbol{x}_p$ for all body-particle pairs (b, p) that are labeled as active or are violated ($\phi_b(\boldsymbol{x}_p) < 0$). Note that pairs such that $\phi_b(\boldsymbol{x}_p) = 0$ (as would be the case once projected) are considered to be touching but not violated. The iterates $\boldsymbol{x}^{(i)}$ obtained at the end of each Newton step, as well as the initial guess, are projected with P.

Register active set Let E' be the objective that would be computed in the unconstrained case. The objective func-

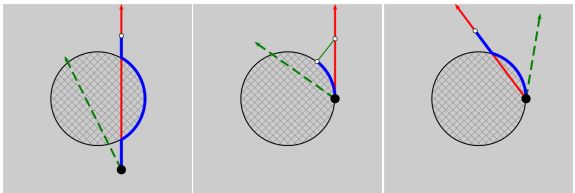

Figure 5: *Line search showing the gradient descent direction (green), Newton direction (red), and effective line search path (blue). The constraint is initially feasible (left), active (middle), and touching but inactive (right). Constraints are projected if violated or active, but only inactive constraints may separate.*

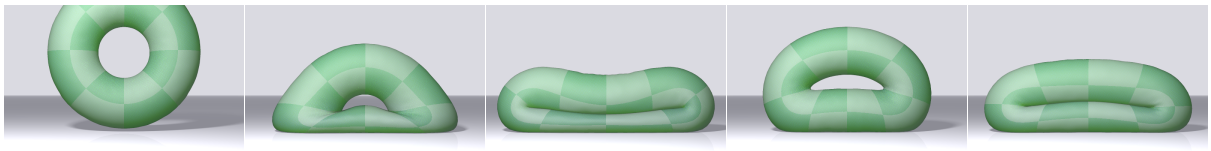

Figure 6: *A torus falls on the ground (constraint collisions) and collides with itself (penalty collisions).*

tion for constrained optimization is $E(\boldsymbol{x}) = E'(P(\boldsymbol{x}))$. Compute the gradient $\nabla E'$. Constraints that are touching and for which $\nabla E' \cdot \nabla \phi_b \geq 0$ are labeled as active for the remainder of the Newton step. All others are labeled as inactive. No constraint should be violated at this stage. Note that $E'(\boldsymbol{x}^{(i)}) = E(\boldsymbol{x}^{(i)})$ is true before and after every Newton step, since constraints are never violated there.

3.2.0.1. Curved paths Note that configurations are always projected to the feasible region before E is computed. One may interpret this as performing line searches along curved paths, as illustrated is Figure 5.

When the unprojected line search curve passes through the medial axis of an object, it is possible for the search curve to be disconnected. This causes a discontinuity in the energy as seen from the line search. If the line search does not stop at the discontinuity, the discontinuity has no effect. If it does, the constraint causing the discontinuity will be active (in which case the discontinuity is projected out) or separating (in which case we move away from the discontinuity) in the next Newton step. Thus a disconnected search curve is not a problem for our method.

3.2.0.2. Derivatives Note also that E must be differentiated twice, and that involves differentiating the projection function P twice. Since P depends on the first derivatives of ϕ_b, the Hessian $\boldsymbol{H}$ of E would seem to require third derivatives. We note, however, that the only occurrence of the third derivative of ϕ_b occurs multiplied by ϕ_b. Since $\boldsymbol{H}$ is used only at the beginning of the Newton step when the configuration is feasible, $\phi_b(\boldsymbol{x}_p) = 0$ or P_{bp} is the identity function. The third derivative term is zero either way, so only the second derivatives of ϕ_b are required.

3.3. Practical considerations

There are a few matters of practicality that are worth mentioning regarding the effective use of this method. The most important of these is that the method does not tolerate discontinuities in E, not even very minute ones, except under some special circumstances that we mention below. In practice, what tends to happen is that a line search encounters a discontinuity in E, where E rises abruptly. The line search dutifully advances the configuration right up to location of this discontinuity. If in the next Newton iteration the descent direction points into the discontinuity, no progress can be made. The solver is stuck. Discontinuities in ∇E can also cause problems and are impossible to avoid in general. As long as these kinks are not valleys of E, they are fine. Thus, the corotated constitutive model, though not completely unusable with this solver, is ill-advised (the fixed variant has no such valleys [SHST12] and is fine). In practice, we have only encountered problems when evaluating self-collision models. The self-collision model we propose works well with the method.

The second practical consideration is that E can be somewhat noisy. This is particularly true with forces that involve an SVD, since its computation involves a balance between speed and accuracy. If the Newton tolerance τ is set too low, the solver will be forced to optimize an objective E where the actual change in E is hidden by the noise. Even with our noisy SVD, we found there is typically at least a three order of magnitude range between the largest value of τ below which no change in output is visually observed and the smallest value above which E is not too noisy to optimize reliably. If we make the E computation robust, E can be optimized down to roundoff level.

Another practical consideration is that occasionally very large changes in the configuration are considered by the line search. For most forces, this is of little consequence. For self-collisions, however, this poses a major performance hazard. We note that when this occurs, the other components of E become very large, too. We first compute all contributions to E except self-collisions. Since our self-collision potential has a global minimum of zero, the real E will be at least as large as the estimate. If this partial E is larger than $E(\boldsymbol{x}^{(i)})$, we do not compute self-collisions at all. While this presents a discontinuity in E to the optimizer, it is safe to do so un-

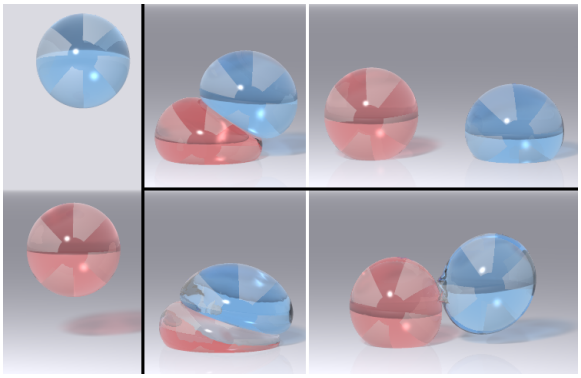

Figure 7: *Two spheres fall and collide with one another with $\Delta t = 1/24\,s$: initial configuration (left), our method (top), and Newton's method (bottom). Notice the artifacts caused by Newton not converging. Newton's method requires six time steps per frame to converge on this example.*

der these conditions, since the optimizer will avoid the large value in E by taking a smaller step along the search line.

4. Forces

Our formulation is fairly insensitive to the underlying forces, provided it has a continuous potential energy function. We use five forces in our simulations. The simplest of these is gravity, which we addressed in Section 2.1. We also employ a hyperelastic constitutive model (Section 4.1), a Rayleigh damping model (Section 4.2), and two collision penalty force models (Sections 5.2 and 5.3).

4.1. Elastic

A suitable hyperelastic constitutive model must have a few key properties to be suitable for this integrator. The most important is that it must have a potential energy function defined everywhere, and this function must be continuous. The constitutive model must be well-defined for any configuration, including configurations that are degenerate or inverted. This is true even if objects do not invert during the simulation, since the minimization procedure may still encounter such states. Examples of suitable constitutive models are those defined by the corotated hyperelasticity energy [ST08, ZSTB10, MG04, EKS03, CPSS10, MZS*11] (but see Section 3.3), and the fixed corotated hyperelasticity variant [SHST12]. Stress-based extrapolated models [ITF04, TSIF05] are unsuitable due to the lack of a potential energy function in the extrapolated regime, but energy-based extrapolation models [SHST12] are fine. We use the fixed corotated variant [SHST12] for all of our simulations for its combination of simplicity and robustness.

4.2. Damping

At first, one might conclude that requiring a potential energy may limit our method's applicability, since damping forces cannot be defined by a potential energy function. A very simple damping model is given by $\boldsymbol{f} = -k\boldsymbol{M}\boldsymbol{v}^{n+1}$. Eliminating the velocity from the equation yields $\boldsymbol{f}(\boldsymbol{x}^{n+1}) = -k\boldsymbol{M}\frac{\boldsymbol{x}^{n+1}-\boldsymbol{x}^n}{\Delta t}$, where $k > 0$. The scalar function $\Phi(\boldsymbol{x}^{n+1}) = \frac{k}{2\Delta t}(\boldsymbol{x}^{n+1}-\boldsymbol{x}^n)^T\boldsymbol{M}(\boldsymbol{x}^{n+1}-\boldsymbol{x}^n)$ has the necessary property that $\boldsymbol{f} = -\frac{\partial\Phi}{\partial\boldsymbol{x}}$. Note that this Φ looks very similar to our inertial term in E, and it is similarly bounded from below. That this Φ is not a real potential energy function is evident from its dependence on $\boldsymbol{x}^n$ and Δt, but it is nevertheless suitable for use in our integrator. This simple drag force is not very realistic, though, so we do not use it in our simulations.

A more realistic damping force is Rayleigh damping. Let ψ be an elastic potential energy function. The stiffness matrix corresponding to this force is $-\frac{\partial^2\psi}{\partial\boldsymbol{x}\partial\boldsymbol{x}}$, and the Rayleigh damping force is $\boldsymbol{f} = -k\left(\frac{\partial^2\psi}{\partial\boldsymbol{x}\partial\boldsymbol{x}}(\boldsymbol{x}^{n+1})\right)\boldsymbol{v}^{n+1}$. This integrates to $\Phi_c = \frac{k}{\Delta t}\left((\boldsymbol{x}^{n+1}-\boldsymbol{x}^n)^T\frac{\partial\psi}{\partial\boldsymbol{x}} - \psi\right)$. This candidate Φ_c has at least two serious problems. The first is that second derivatives of Φ_c involve third derivatives of ψ. The second is that

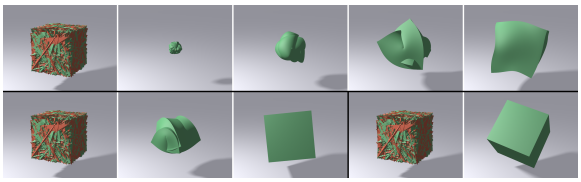

Figure 8: *Random test with $65\times65\times65$ particles simulated with $\Delta t = 1/24\,s$ for three stiffnesses: low stiffness recovering over 100 time steps (top), medium stiffness recovering over 40 time steps (bottom left), and high stiffness recovering in a single time step (bottom right). The red tetrahedra are inverted, while the green are uninverted.*

$\frac{\partial^2\psi}{\partial\boldsymbol{x}\partial\boldsymbol{x}}$ may be indefinite, in which case the damping force may not be entirely dissipative. Instead, we approximate Rayleigh damping with a lagged version. Let $\boldsymbol{D} = \frac{\partial^2\psi}{\partial\boldsymbol{x}\partial\boldsymbol{x}}(\boldsymbol{x}^n)$. Since $\boldsymbol{D}$ does not depend on $\boldsymbol{x}^{n+1}$, the lagged Rayleigh damping force $\boldsymbol{f} = -k\boldsymbol{D}\boldsymbol{v}^{n+1}$ leads to $\Phi_d = \frac{k}{2\Delta t}(\boldsymbol{x}^{n+1}-\boldsymbol{x}^n)^T\boldsymbol{D}(\boldsymbol{x}^{n+1}-\boldsymbol{x}^n)$. This solves the first problem, since the second derivative of Φ_d is just $\frac{k}{\Delta t}\boldsymbol{D}$. Since $\boldsymbol{D}$ is not being differentiated, it is safe to modify it to eliminate indefiniteness as described in [TSIF05, SHST12]. This addresses the second problem. We did not use the damping model found in [KYT*06a], which uses $\psi(\boldsymbol{x}^{n+1})$ with $\boldsymbol{x}^n$ used as the rest configuration, because it is not defined when $\boldsymbol{x}^n$ is degenerate.

5. Collisions

Collisions are a necessary part of any practical computer graphics simulator. The simplest approach to handling collisions is to process them as a separate step in the time integration scheme. This works well for small time steps, but it causes problems when used with large time steps as seen in Figure 7. Such arrangement often leads to the collision step flattening objects to remove penetration and the elastic solver restoring the flattened geometry by pushing it into the colliding object. To get around this problem, the backward Euler solver needs to be aware of collisions. A well-tested strategy for doing this is to use penalty collisions, and we do this for two of our three collision processing techniques.

5.1. Object collisions as constraints

Our first collision processing technique takes advantage of our minimization framework to treat collisions with non-simulated objects as inequality constraints. Treating collisions or contacts as constraints is not new and in fact forms the basis for LCP formulations such as [KSJP08, GZO10]. Unlike LCP formulations, however, our formulation does not attempt to be as complete and as a result can be solved about as efficiently as a simple penalty formulation.

Our constraint collision formulation works reliably when the level set is known analytically. This limits its applicability to analytic collision objects. While this approach is feasible only under limited circumstances, these circumstances

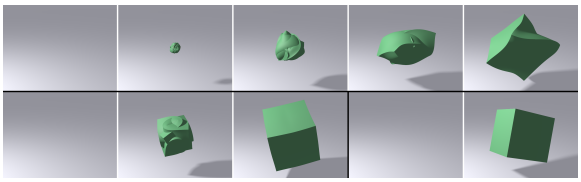

Figure 9: *Point test with $65 \times 65 \times 65$ particles simulated with $\Delta t = 1/24\,s$ for three stiffnesses: low stiffness recovering over 120 time steps (top), medium stiffness recovering in 5 time steps (bottom left), and high stiffness recovering in a single time step (bottom right).*

occur frequently in practice. When this approach is applicable, it is our method of choice, since it produces better results (e.g., no interpenetration) for similar cost. When this formulation is not applicable, we use a penalty collision formulation instead.

We begin by representing our collision objects (indexed with b) by a level set, which we denote ϕ_b to avoid confusion with potential energy. By convention, $\phi_b(\boldsymbol{x}) < 0$ for points $\boldsymbol{x}$ in the interior of the collision object b. Our collision constraint is simply that $\phi_b(\boldsymbol{x}_p^{n+1}) \geq 0$ for each simulation particle p and every constraint collision object b. With such a formulation, we can project a particle at $\boldsymbol{x}_p$ to the closest point $\boldsymbol{x}'_p$ on the constraint manifold using $\boldsymbol{x}'_p = P_{bp}(\boldsymbol{x}_p) = \boldsymbol{x}_p - \phi_b(\boldsymbol{x}_p)\nabla\phi_b(\boldsymbol{x}_p)$. We show how to solve the resulting minimization problem in Section 3.2.

We apply friction after the Newton solve. The total collision force felt by particles is $\nabla E'(\boldsymbol{x}^{n+1}) - \nabla E(\boldsymbol{x}^{n+1}) = \nabla E'(\boldsymbol{x}^{n+1}) - \nabla E'(P(\boldsymbol{x}^{n+1}))$ (See Section 3.2 for the definition of E'). Only collision pairs that are active at the end of the minimization will be applying such forces. We use the level set's normal and the collision force to apply Coulomb friction to colliding particles.

Our constraint collision formulation is not directly applicable to grid-based level sets, since we assume that $P_{bp}(P_{bp}(\boldsymbol{x}_p)) = P_{bp}(\boldsymbol{x}_p)$ and $P_{bp}(x)$ is continuous. Continuity of $P_{bp}(x)$ can be achieved, for example, with C^1 cubic spline level set interpolation. However, it will not generally be true that $P_{bp}(P_{bp}(\boldsymbol{x}_p)) = P_{bp}(\boldsymbol{x}_p)$. Alternatively, the projection routine can be modified to iterate the projection to convergence, but then continuity is lost.

5.2. Object penalty collisions

When a collision object is not analytic, as will normally be the case for characters for instance, we use a penalty formulation instead. As in the constraint formulation, we assume our collision object is represented by a level set ϕ_b. The elastic potential energy $\Phi_{bp}(\boldsymbol{x}_p)$ of our penalty force is $\Phi_{bp}(\boldsymbol{x}) = 0$ if $\phi_b(\boldsymbol{x}_p) > 0$ and $\Phi_{bp}(\boldsymbol{x}_p) = k\phi_b(\boldsymbol{x}_p)^3$ otherwise. Since Φ_{bp} is a potential energy, we must differentiate it twice for our solver. It is important to compute the derivatives of ϕ_b exactly by differentiating the interpolation routine rather than approximating them using central differences. While a C^1 cubic spline interpolation is probably a wiser interpolation strategy since it would avoid the energy kinks that may be caused by a piecewise linear encoding of the level set, we found linear interpolation to work well, too, and we use linear interpolation in our examples.

As in the constraint case, we apply friction after the Newton solve. The total collision force felt by a particle due to object penalty collisions is obtained by evaluating the penalty force at $\boldsymbol{x}^{n+1}$. We compute the component of the discrete acceleration $\frac{\boldsymbol{x}^{n+1} - \boldsymbol{x}^n - \Delta t \boldsymbol{v}^n}{\Delta t^2}$ perpendicular to the collision force and apply Coulomb friction in the opposite direction to the colliding particle.

5.3. Penalty self-collisions

We detect self-collisions by performing point-tetrahedron inclusion tests, which we accelerate with a bounding box hierarchy. If a point is found to be inside a tetrahedron but not one of the vertices of that tetrahedron, then we flag the particle as colliding.

Once we know a particle is involved in a self collision, we need an estimate for how close the particle is to the boundary. If this particle has collided before, we use the primitive it last collided with as our estimate. Otherwise, we compute the approximate closest primitive in the rest configuration using a level set and use the current distance to this surface element as an estimate.

Given this upper bound estimate of the distance to the boundary, we perform a bounding box search to conservatively return all surface primitives within that distance. We check these candidates to find the closest one. Now we have a point-primitive pair, where the primitive is the surface triangle, edge, or vertex that is closest to the point being processed. Let d be the square of the point-primitive distance. The penalty collision energy for this point is $\Phi = kd\sqrt{d+\varepsilon}$, where ε is a small number (10^{-15} in our case) to prevent the singularities when differentiating. Note that this penalty function is approximately cubic in the penetration depth. This final step is the only part that must be differentiated.

As with the other two collision models, we apply friction after the Newton solve. In the most general case, a point n_0 collides with a surface triangle with vertices n_1, n_2, and n_3. As with the object penalty collision model, collision forces are computed by evaluating $\Phi(\boldsymbol{x}^{n+1})$ and its derivative. The force applied to n_0 is denoted $\boldsymbol{f}$; its direction is taken to be the normal direction $\boldsymbol{n}$. The closest point on the triangle to n_0 has barycentric weights w_1, w_2, and w_3. Let $w_0 = -1$ for convenience. Let $\boldsymbol{Q} = \boldsymbol{I} - \boldsymbol{n}\boldsymbol{n}^T$. If we apply a tangential impulse $\boldsymbol{Q}\boldsymbol{j}$ to these particles, their new velocities will be $\hat{v}_{n_i} = v_{n_i} + w_i m_{n_i}^{-1} \boldsymbol{Q}\boldsymbol{j}$, and total kinetic energy will be $KE = \sum_{n=0}^{3} \frac{1}{2} m_{n_i} \hat{v}_{n_i}^T \hat{v}_{n_i}$. We want to minimize this kinetic energy to prevent friction from causing instability. Since M is positive definite, we see that KE is minimized when $\nabla KE = \boldsymbol{Q}\bar{v} + \bar{m}^{-1}\boldsymbol{Q}\boldsymbol{j} = 0$, where $\bar{v} = \sum_{n=0}^{3} w_i v_{n_i}$

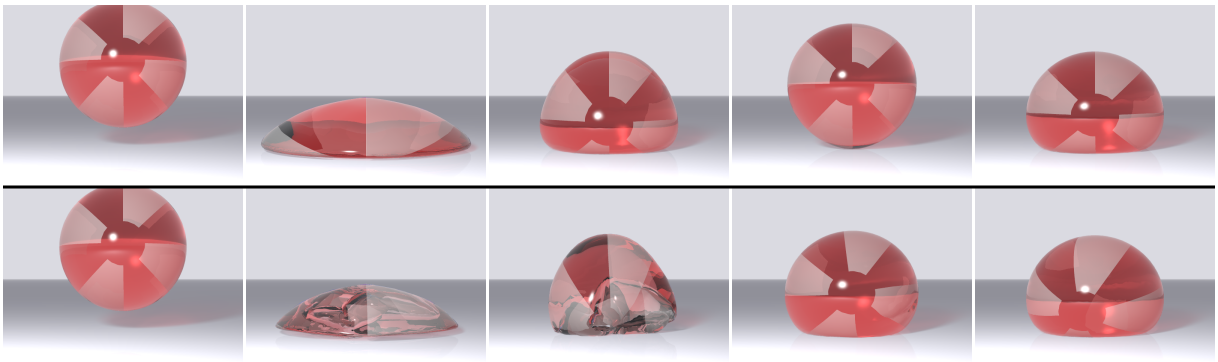

Figure 10: *Sphere dropping hard on the ground with $\Delta t = 1/24\,s$ with constraint collisions (top) and collisions as a post-process (bottom). Penalty collisions produce a result very similar to constraint collisions, though some penetration with the ground occurs. Note that the post-processing approach leads to inversion during recovery from the collision.*

and $\overline{m}^{-1} = \sum_{n=0}^{3} w_i m_{n_i}^{-1} w_i$. Thus if we let $\boldsymbol{j} = -\overline{m}\boldsymbol{Q}\overline{v}$ then $\nabla KE = 0$, and $\boldsymbol{Q}\boldsymbol{j} = \boldsymbol{j}$. If $\|\boldsymbol{j}\| < \mu\|\boldsymbol{f}\|$, then we choose $\boldsymbol{j}' = \boldsymbol{j}$ as our friction impulse. Otherwise, $\boldsymbol{j}' = \mu\|\boldsymbol{f}\|\frac{\boldsymbol{j}}{\|\boldsymbol{j}\|}$. Finally, the new velocities are $v'_{n_i} = v_{n_i} + w_i m_{n_i}^{-1} \boldsymbol{j}'$. Note that all three friction algorithms decrease kinetic energy but do not modify positions, so none of them can add energy to the system, and thus stability ramifications are unlikely even though friction is applied explicitly. This approach to friction can have artifacts, however, since friction will be limited to removing kinetic energy from colliding particles. This limits the amount of friction that can be applied at large time steps. An approach similar to the one in [KSJP08] that uses successive Quadratic Programming solves could possibly be applied to eliminate these artifacts. However [ZJ11] found existing large-scale sparse QP solvers to be insufficiently robust, and thus we did not use this method.

6. Results

We begin by demonstrating how robust our solver is by considering the two most difficult constitutive model tests we are aware of: total randomness and total degeneracy. The attributes that make them tough constitutive model tests also make them tough solver tests: high stress, terrible initial guess, tangled configurations, and the need to dissipate massive amounts of unwanted energy. Figure 8 shows the recovery of a $65 \times 65 \times 65$ cube (824k dofs) from a randomized initial configuration for three different stiffnesses with $\Delta t = 1/24\,s$. Figure 9 repeats the tests with all points starting at the origin. The recovery times vary from about $3\,s$ for the softest to a single time step for the stiffest. We were surprised to find that a single step of backward Euler could untangle a randomized cube, even at high resolution.

Figure 6 is a classical torus drop demonstrating that our self collisions are effective at stopping collisions at the torus's hole. Figure 12 uses constraints for all collision body collisions and demonstrates that our constraint collisions are effective with concave and convex constraint manifolds. Figure 14 demonstrates our method with stiffer deformable bodies with sharp corners. Figure 13 demonstrates our constraint collisions are effective for objects with sharp corners. Finally, Figure 1 shows a more practical example which uses all three types of collisions: self collisions, constraint collisions (with ground) and penalty collisions (against a bowl defined by a grid-based level set).

7. Conclusions

We have demonstrated that backward Euler solved with Newton's method can be made more robust by recasting the

Figure	Ours?	$\frac{\text{Steps}}{\text{frame}}$	$\frac{\text{Time}}{\text{frame}}$ (s)	# dofs	$\frac{\text{Solves}}{\text{step}}$
1	Y	5	200	984k	2.2
3 mid	Y	1	0.51	18.5k	2.8
3 rt	N	1/3	8.7/1.1	18.5k	15/0.7
4 top	Y	1	0.52	18.5k	2.9
4 bot	N	1/5	3.8/1.3	18.5k	6.6/0.6
7 top	Y	1	4.25	28.0k	8.1
7 bot	N	1/6	33/7.3	28.0k	26/0.8
6	Y	5	1.13	7.9k	2.1
8 top	Y	1	68.0^{+}	824k	12.3
8 lt	Y	1	1470^{+}	824k	236.8
8 rt	Y	1	667^{+}	824k	109.6
9 top	Y	1	43.1^{+}	824k	10.7
9 lt	Y	1	831^{+}	824k	155.9
9 rt	Y	1	444^{+}	824k	88.8
10 top	Y	1	0.42	14.0k	3.8
10 bot	N	1^{*}	1.13	14.0k	9.8
12	Y	1	0.45	7.9k	8.6
13	Y	1	46.1	73.8k	34.7
14	Y	1	17.1	138k	6.9

Figure 11: *Time step sizes and average running times for the examples in the paper. The last column shows the average number of linear solves per time step. Each of the Newton's method examples fails to converge at the frame rate. For fairer comparison, timing information for all but the one marked * is shown at the frame rate and the stable time step size. The stress tests marked $^{+}$ spend the majority of their time on the first frame or two due to the difficult initial state.*

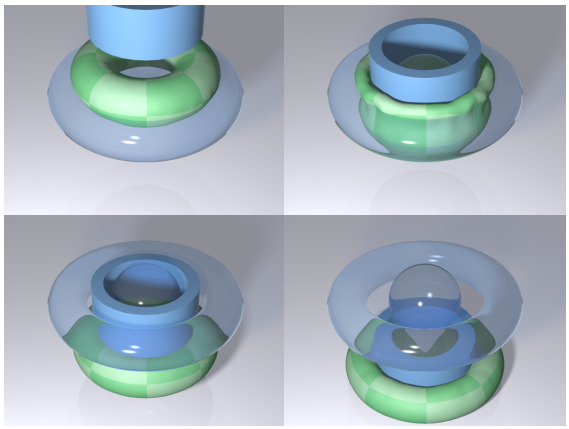

Figure 12: *A torus is pushed through a hole (constraint collisions).*

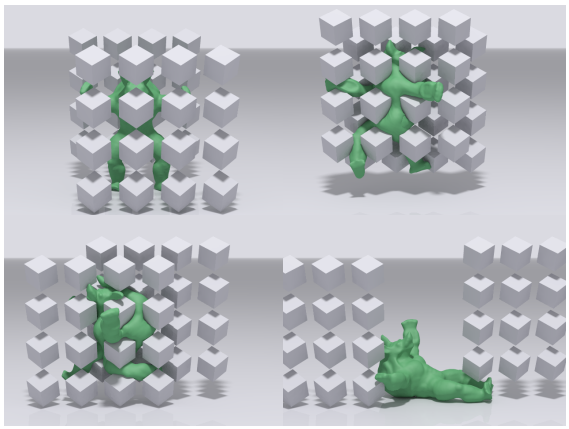

Figure 13: *An armadillo is squeezed between 32 rigid cubes (constraint collisions) with $\Delta t = 1/24\,s$. When this torture test is run at 1, 2, 4 and 8 steps per frame the average runtime per frame is 46, 58, 88, and 117 seconds respectively.*

resulting system of nonlinear equations as a nonlinear optimization problem so that robust optimization techniques can be employed. The resulting method is extremely robust to large time step sizes, high stress, and tangled configurations.

Runtimes and other performance-related information for all of our sims are provided in Figure 11. All simulations were run single-threaded on a $3.1-3.5\,GHz$ Xeon core. Our solver's performance is competitive with a standard Newton solver for those examples where both were run. In general, we take more Newton steps but spend less time on each, and the resulting runtime for typical examples is about the same for the two solvers, though our solver is faster for all of the difficult examples in this paper. Taking a large time step size can actually be slower than taking a smaller one, even with the same solver. For time integrators (like backward Euler) that have a significant amount of damping at large time steps, constitutive models are often tuned to take into account the numerical damping. If the integrator is forced to simulate a portion of a simulation at a smaller time step, the dynamic behavior can change noticeably. Solving with constraints is about the same speed as using penalty collisions.

Note that Figures 1 and 6 were run with smaller time steps sizes to avoid collision artifacts. This indicates that a self-collision scheme that is more tolerant of large time steps is required. The scheme does not have problems with collisions between different objects at the frame rate as long as they are not too thin. Continuous collision detection could perhaps be used. We leave both of these problems for future work.

The current method has a couple disadvantages compared with current techniques. It requires a potential energy to exist (which is how most constitutive models are defined anyway) and is sensitive to discontinuities in this energy. The method also occasionally fails to make progress due to valley shaped kinks in our collision processing. In practice, this only occurs when the system is already fairly close to a solution, since otherwise any energy kinks are overwhelmed by the strong gradients in the objective. From a practical perspective, this means this sort of breakdown can be dealt with by simply ignoring it. This does, however, prevent the method from being absolutely robust. We leave this weakness to be addressed in future work.

Our method was derived and implemented on top of a backward Euler integrator, which is known for being very stable but quite damped. The nonlinear system of equations for other A-stable integrators such as trapezoid rule and BDF-2 can also be readily converted into minimization form and solved similarly. Being second order schemes, their use would reduce damping at large time steps, though trapezoid rule's oscillatory properties should be taken into account.

8. Acknowledgments

We would like to acknowledge Shunsuke Saito and Yuwei Jiang for their suggestions regarding optimization. All authors were partially supported by NSF (DMS-0502315, DMS-0652427, CCF-0830554), DOE (09-LR-04-116741-BERA), ONR (N000140310071, N000141010730, N000141210834) and Intel STCVisual Computing Grant (20112360).

References

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. In *ACM Trans. Graph. (ToG)* (2002), vol. 21, ACM, pp. 594–603. 1

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *Proc. Symp. Comp. Anim.* (2003), pp. 28–36. 1

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proc. SIGGRAPH* (1998), pp. 43–54. 1

[CK05] CHOI K.-J., KO H.-S.: Stable but responsive cloth. In *ACM SIGGRAPH 2005 Courses* (2005), ACM, p. 1. 1

[CPSS10] CHAO I., PINKALL U., SANAN P., SCHRÖDER P.: A simple geometric model for elastic deformations. *ACM Trans. Graph. 29* (2010), 38:1–38:6. 6

[EEH00] EBERHARDT B., ETZMUSS O., HAUTH M.: *Implicit-explicit schemes for fast animation with particle systems*. Springer, 2000. 1

[EKS03] ETZMUSS O., KECKEISEN M., STRASSER W.: A fast finite element solution for cloth modeling. In *Proc. Pac. Graph.* (2003), pp. 244–251. 6

[GHF*07] GOLDENTHAL R., HARMON D., FATTAL R., BERCOVIER M., GRINSPUN E.: Efficient simulation of inextensible cloth. In *ACM Trans. on Graph. (TOG)* (2007), vol. 26, ACM, p. 49. 2

[GSO10] GONZALEZ M., SCHMIDT B., ORTIZ M.: Force-stepping integrators in lagrangian mechanics. *Intl. J. for Num. Meth. in Engng. 84*, 12 (2010), 1407–1450. 1

[GZO10] GASCÓN J., ZURDO J. S., OTADUY M. A.: Constraint-based simulation of adhesive contact. In *Proc. Symp. Comp. Anim.* (2010), pp. 39–44. 6

[HE01] HAUTH M., ETZMUSS O.: A high performance solver for the animation of deformable objects using advanced numerical methods. In *Comp. Graph. Forum* (2001), vol. 20, pp. 319–328. 1

[HFL*01] HIROTA G., FISHER S., LEE C., FUCHS H., ET AL.: An implicit finite element method for elastic solids in contact. In *Comp. Anim., 2001.* (2001), IEEE, pp. 136–254. 1, 2

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. In *Proc. Symp. Comp. Anim.* (2004), pp. 131–140. 6

[Kan99] KANE C.: *Variational integrators and the Newmark algorithm for conservative and dissipative mechanical systems*. PhD thesis, Caltech, 1999. 1

[KMO99] KANE C., MARSDEN J. E., ORTIZ M.: Symplectic-energy-momentum preserving variational integrators. *J. Math. Phys. 40* (1999), 3353. 1, 2

[KSJP08] KAUFMAN D. M., SUEDA S., JAMES D. L., PAI D. K.: Staggered projections for frictional contact in multibody systems. In *ACM Trans. Graph. (TOG)* (2008), vol. 27, ACM, p. 164. 6, 8

[KYT*06a] KHAREVYCH L., YANG W., TONG Y., KANSO E., MARSDEN J., SCHRÖDER P.: Geometric, variational integrators for computer animation. In *Proc. Symp. Comp. Anim.* (2006), pp. 43–51. 6

[KYT*06b] KHAREVYCH L., YANG W., TONG Y., KANSO E., MARSDEN J. E., SCHRÖDER P., DESBRUN M.: Geometric, variational integrators for computer animation. In *Proc. Symp. Comp. Anim.* (2006), pp. 43–51. 1, 2

[LBOK13] LIU T., BARGTEIL A. W., O'BRIEN J. F., KAVAN L.: Fast simulation of mass-spring systems. *ACM Trans. Graph. (TOG) 32*, 6 (2013), 214. 1, 2

[LMOW04] LEW A., MARSDEN J., ORTIZ M., WEST M.: Variational time integrators. *Intl. J. Num. Meth. Engng. 60*, 1 (2004), 153–212. 1

[MG04] MÜLLER M., GROSS M.: Interactive virtual materials. In *Proc. Graph. Intl.* (2004), pp. 239–246. 6

[MSW13] MICHELS D., SOBOTTKA G., WEBER A.: Exponential integrators for stiff elastodynamic problems. In *ACM Trans. Graph. (TOG)* (2013), ACM. 1, 2

[MTGG11] MARTIN S., THOMASZEWSKI B., GRINSPUN E., GROSS M.: Example-based elastic materials. In *ACM Trans. Graph. (TOG)* (2011), vol. 30, ACM, p. 72. 1, 2

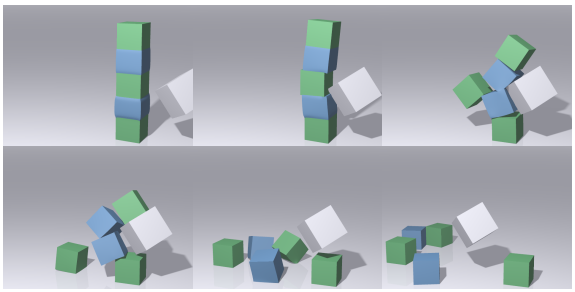

Figure 14: *A stack of deformable boxes of varying stiffness is struck with a rigid kinematic cube (constraint collisions) with $\Delta t = 1/24\,s$. The green boxes are 10 times as stiff as the blue boxes.*

[MZS*11] MCADAMS A., ZHU Y., SELLE A., EMPEY M., TAMSTORF R., TERAN J., SIFAKIS E.: Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph. 30* (2011), 37:1–37:12. 6

[NW06] NOCEDAL J., WRIGHT S.: *Numerical Optimization*. Springer series in operations research and financial engineering. Springer, 2006. 3, 4

[PF02] PARKS D., FORSYTH D.: Improved integration for cloth simulation. In *Proc. of Eurographics* (2002). 1

[PKMO02] PANDOLFI A., KANE C., MARSDEN J., ORTIZ M.: Time-discretized variational formulation of non-smooth frictional contact. *Intl. J. Num. Meth. Engng. 53* (2002), 1801–1829. 3

[SG09] STERN A., GRINSPUN E.: Implicit-explicit variational integration of highly oscillatory problems. *Multiscale Modeling & Simulation 7*, 4 (2009), 1779–1794. 1

[SHST12] STOMAKHIN A., HOWES R., SCHROEDER C., TERAN J. M.: Energetically consistent invertible elasticity. In *Proc. Symp. Comp. Anim.* (2012), pp. 25–32. 5, 6

[SSF13] SU J., SHETH R., FEDKIW R.: Energy conservation for the simulation of deformable bodies. *Visualization and Computer Graphics, IEEE Transactions on 19*, 2 (2013), 189–200. 1

[ST08] SCHMEDDING R., TESCHNER M.: Inversion handling for stable deformable modeling. *Vis. Comp. 24* (2008), 625–633. 6

[STW92] SIMO J. C., TARNOW N., WONG K.: Exact energy-momentum conserving algorithms and symplectic schemes for nonlinear dynamics. *Computer methods in applied mechanics and engineering 100*, 1 (1992), 63–116. 1

[TSIF05] TERAN J., SIFAKIS E., IRVING G., FEDKIW R.: Robust quasistatic finite elements and flesh simulation. In *Proc. Symp. Comp. Anim.* (2005), pp. 181–190. 6

[VMT01] VOLINO P., MAGNENAT-THALMANN N.: Comparing efficiency of integration methods for cloth simulation. In *Comp. graph. Intl. 2001 Proc.* (2001), IEEE, pp. 265–272. 1

[ZJ11] ZHENG C., JAMES D. L.: Toward high-quality modal contact sound. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 38. 8

[ZSTB10] ZHU Y., SIFAKIS E., TERAN J., BRANDT A.: An efficient and parallelizable multigrid framework for the simulation of elastic solids. *ACM Trans. Graph. 29* (2010), 16:1–16:18. 6