Energy Conservation for the Simulation of Deformable Bodies

Jonathan Su, Rahul Sheth, and Ronald Fedkiw

Abstract—We propose a novel technique that allows one to conserve energy using the time integration scheme of one's choice. Traditionally, the time integration methods that deal with energy conservation, such as symplectic, geometric, and variational integrators, have aimed to include damping in a manner independent of the size of the time step, stating that this gives more control over the look and feel of the simulation. Generally speaking, damping adds to the overall aesthetics and appeal of a numerical simulation, especially since it damps out the high frequency oscillations that occur on the level of the discretization mesh. We propose an alternative technique that allows one to use damping as a material parameter to obtain the desired look and feel of a numerical simulation, while still exactly conserving the total energy - in stark contrast to previous methods in which adding damping effects necessarily removes energy from the mesh. This allows, for example, a deformable bouncing ball with aesthetically pleasing damping (and even undergoing collision) to collide with the ground and return to its original height exactly conserving energy, as shown in Figure 2. Furthermore, since our method works with any time integration scheme, the user can choose their favorite time integration method with regards to aesthetics and simply apply our method as a post-process to conserve all or as much of the energy as desired.

Index Terms—Computer Graphics, Physically-based Modeling

1 Introduction

Deformable models have been used in computer graphics for over twenty years, dating back to the early work of [1]–[3]. While our work will focus on mass-spring models, the main ideas should be extendable to finite element methods (such as those in [4] and [5]), though we do not evaluate FEMs within the scope of this paper. Many authors have explored various time integration schemes, such as fully implicit methods (e.g. [2], [6]), semi-implicit schemes (e.g. [7], [8]), and explicit schemes (e.g. [9]).

Damping plays a much larger role in a numerical simulation than one might otherwise first expect. Implicit time integration schemes possess inherent damping, which can lead to undesirable artifacts as discussed by [10]. Other authors, such as [11], stressed the desirability of schemes that add a known quantitative amount of damping independent of the time step, as opposed to an unknown quantity of scheme-inherent damping present in many implicit schemes. It is important to note that while some schemes require damping in order to achieve stability, even those that do not require it for stability use damping to achieve aesthetically pleasing results. For example, [12] states "in application practice, one generally wishes to have damping" but stresses the importance of damping independent of the time step in contrast to numerical damping, similar in vein to arguments in [10]. Similarly in the case of fluids, [13] states "completely inviscid [undamped] flows may look unnatural" and "fluid animation in computer graphics requires a small amount of viscosity to render the motion more realistic." We take this one step further. The aforementioned papers first achieve energy conservation and then add an explicitly controlled amount of time step-independent damping to achieve realistic results, but that damping leads to a loss of energy conservation. In fact, the authors of [12] show exactly this phenomena in their paper talk [14], where the energy conserving variational integrator produces high-frequency vibrations in an oscillating cube, and damping is used to achieve a more visually appealing simulation at the cost of energy conservation. Instead, our scheme allows for any

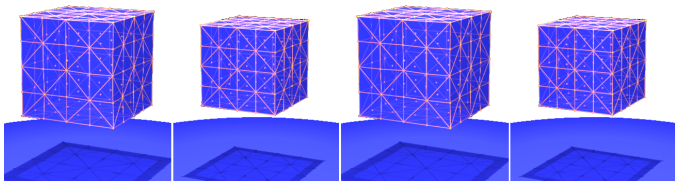

Fig. 1. Our global energy correction allows one to use damping to improve the aesthetics of a simulation while maintaining energy conservation, enabling a stretched cube to oscillate uniformly in and out while exactly conserving energy. In contrast, the variational integrator seen in [14] exhibits either high speed oscillations when exactly conserving energy or loss of energy when damping is added to make the simulation more visually appealing.

- *J. Su is with the Computer Science Department, Stanford University, Stanford, CA, 94305 and Intel Labs, Intel Corporation, Santa Clara, CA 95054.*
  E-mail: jonsu@stanford.edu.
- *R. Sheth is with the Computer Science Department, Stanford University, Stanford, CA 94305.*
  Email: rbsheth@stanford.edu.
- *R. Fedkiw is with the Computer Science Department, Stanford University, Stanford, CA 94305 and Industrial Light + Magic, San Francisco, CA, 94129.*
  E-mail: fedkiw@cs.stanford.edu.

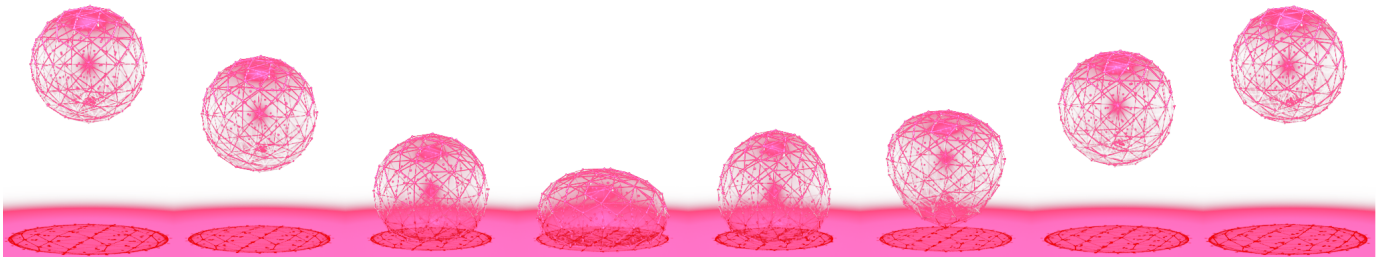

Fig. 2. Our global energy correction technique is used to restore all energy lost during evolution and collisions, allowing the sphere to bounce to its original height. It also restores energy lost from damping, as this simulation uses the semi-implicit Newmark scheme from [8].

amount of explicit or implicit damping in one's time integration scheme of choice and subsequently applies a correction that yields exact energy conservation as well - something not present in any contemporary methods in computer graphics or computational mechanics. This allows damping to be used as a material property to control the visual attractiveness of the simulation while maintaining energy conservation. Using our scheme, the cube from [14] can oscillate uniformly with the addition of damping while still conserving energy in Figure 1. Furthermore, the simulation shown in Figure 2 with an aesthetically pleasing time-integration scheme including damping (and collisions) allows for the sphere to bounce back to its original height with an effective elastic restitution coefficient of 1. Any other method that allows for any amount of implicit or explicit damping, independent of the time step or not, would dissipate energy, modeling a coefficient of restitution strictly less than 1. Any method that ignores damping (assuming it can correctly handle the collisions) in order to achieve exact energy conservation to allow the ball to return to the same height would then lose the ability to achieve the aesthetically pleasing results that damping allows.

We stress that both damping and energy conservation are important aspects of our approach. Whereas damping allows for aesthetically pleasing results, without energy conservation, the ball cannot return to its original height. Energy conservation also guarantees that simulations do not become unstable, which is especially important if one considers large time steps and real-time applications. Also pertinent to large time steps in real-time simulations as well as long-time simulations is the notion of *cumulative* damping. Whether the damping is intrinsically inherent in an implicit method or explicitly added independent of the time step in a method that otherwise conserves energy, repeated damping time step after time step will eventually lead to large amounts of energy being lost from the system. This situation is even worse for implicit damping where larger time steps mean faster energy loss in shorter periods of time. So in spite of methods that add explicit damping independent of the time step fairing better than other methods, over a long enough period of time the damping added to increase the visual fidelity of the system at each time step eventually removes significantly large amounts of energy from the system, cumulatively damping it out. A large advantage of our technique is that one can obtain the same visual fidelity with the same amount of damping yet never lose any energy from the system. Instead of removing high frequency energy via damping, we essentially move high frequency energy into lower frequencies, giving both visual fidelity and exact energy conservation.

Given the importance of energy conservation, we begin our exposition in Section 2 with a bottom-up approach to energy conservation for non-linear elastic systems similar to some of the earliest papers on the topic, such as [15] which considered the motion of a single particle and the subsequent paper [16] that considered systems of particles. This leads us one step at a time towards a conclusion similar to theirs, that an iterative approach is required (one of their methods requires iteration and the other requires dual iteration). However, our bottom-up approach looks at the problem a bit differently in that we consider it from the standpoint of forces. Our resulting scheme is really no different in spirit from any number of contemporary energy conservative time integration schemes in addition to those already mentioned above. We refer the interested reader to a selection of papers on the topic (e.g. [17]–[21]) as well as the book [22]. We note that the energy conserving time integration scheme that we propose in Section 2 is not intended to directly compete against the plethora of methods in the literature, but rather is used to illustrate a slightly different approach to driving a scheme of this type and as such lends itself to what we refer to as an energy budgeting process.

This energy budgeting process is a key ingredient in our novel technique which allows one to exactly conserve energy for any time integration method, explicit or implicit, with inherent or explicit damping for aesthetics as well as collisions, self-collisions, etc. For this reason, we continue the development of this scheme from the single spring force in Section 2.1, to multiple springs in Section 2.2, gravity in Section 2.3, multiple dimensions in Section 2.4, and more complex meshes in Section 3, before stating the main result of this paper in Sections 4 and 5. Notably, the method proposed in Section 4 does not require iteration, but instead achieves exact energy conservation through solving a single quadratic formula. This is in the spirit of the projection-type method in [23], but since our approach is different our method lends

itself to the simulation of more complex phenomena such as damping, collisions, and self-collisions (see Sections 4 and 5).

2 Spring Forces

2.1 Simple Spring

We begin by considering the evolution of a single spring in one spatial dimension. For the sake of illustration, we have chosen to use a Newmark time integration scheme similar to [8], but a similar approach for energy conservation could be taken with other time integration schemes. Recall that in the context of a Newmark time integration scheme, the evolution of a single particle is defined as follows:

I. $v^{n+1/2} = v^n + \frac{\Delta t}{2m} F$
II. $x^{n+1} = x^n + \Delta t v^{n+1/2}$
III. $v^{n+1} = v^n + \frac{\Delta t}{m} F$

where F is the net force on the particle. In the context of a single spring, this will result in changes in kinetic and potential energy which look like

$$\Delta KE = \frac{1}{2} m_1 \left[(v_1^{n+1})^2 - (v_1^n)^2 \right] + \frac{1}{2} m_2 \left[(v_2^{n+1})^2 - (v_2^n)^2 \right] \quad (1)$$

$$\Delta PE = \frac{1}{2} \frac{k}{\ell_0} \left(|x_2^{n+1} - x_1^{n+1}| - \ell_0 \right)^2 - \frac{1}{2} \frac{k}{\ell_0} \left(|x_2^n - x_1^n| - \ell_0 \right)^2 \quad (2)$$

where k is the Young's modulus and l_0 is the restlength of the spring. Substituting the updates for position and velocity from steps I, II, and III in for x^{n+1} and v^{n+1} leads to

$$\Delta KE = \frac{1}{2} \frac{\Delta t^2}{\hat{m}} F^2 + \Delta t (v_1^n - v_2^n) F \quad (3)$$

$$\Delta PE = \frac{1}{2} \frac{k}{\ell_0} (2ba + b^2 + 2acF + 2bcF + c^2 F^2 - 2\ell_0 |a + b + cF| + 2\ell_0 |a|) \quad (4)$$

where

$$a = x_2^n - x_1^n \quad (5)$$

$$b = (v_2^n - v_1^n) \Delta t \quad (6)$$

$$c = -\frac{1}{2} \frac{\Delta t^2}{\hat{m}} \quad (7)$$

$$\frac{1}{\hat{m}} = \frac{1}{m_1} + \frac{1}{m_2}. \quad (8)$$

In order for energy to be conserved, $\Delta KE = -\Delta PE$ must be true. We note that using the typical elastic spring force $F = \frac{k}{l_0}(l - l_0)$, where l is the spring's current length, does not in general satisfy this equation and therefore is not guaranteed to conserve energy.

To find an energy conserving spring force, we plug equations 3 and 4 into $\Delta KE = -\Delta PE$ and solve for F, making the assumption that the spring does not change directions (which removes the absolute values). This produces the quadratic equation

$$AF^2 + BF + C = 0 \quad (9)$$

where

$$A = \frac{1}{2} \frac{\Delta t^2}{\hat{m}} + \frac{1}{2} \frac{k}{\ell_0} c^2 \quad (10)$$

$$B = \Delta t \, (v_1^n - v_2^n) + \frac{k}{\ell_0} (ac + bc - \ell_0 c) \quad (11)$$

$$C = \frac{k}{\ell_0} \left(ab + \frac{1}{2} b^2 - \ell_0 b \right). \quad (12)$$

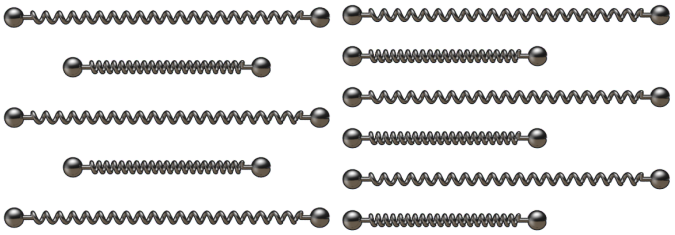

Fig. 3. (Left) A single unconstrained spring released from a stretched state and evolved using our new spring force, allowing it to continue to oscillate about the restlength. (Right) A single spring constrained at the left endpoint, released from a stretched state and evolved using our new spring force, allowing it to continue to oscillate about the restlength. Note that we chose a time step equal to the size of the framerate for both simulations.

Solving equation 9 produces two possible spring forces that exactly conserve energy given the current spring state and time step size. One root corresponds to a spring force which would result in a compressed energy-conserving final state for the spring, whereas the other root would result in an expanding final configuration for the spring. In order to choose the correct one, we compare the resulting velocity to the analytic velocity for the spring. Recall that the analytic solution for the velocity of the endpoints of a spring are

$$v_1^{n+1} = v_{cm}^n - \frac{m_2}{m_1 + m_2} \left[-\omega\alpha \sin \omega \Delta t + \omega\beta \cos \omega \Delta t \right] \quad (13)$$

$$v_2^{n+1} = v_{cm}^n + \frac{m_1}{m_1 + m_2} \left[-\omega\alpha \sin \omega \Delta t + \omega\beta \cos \omega \Delta t \right] \quad (14)$$

where v_{cm} is the velocity of the center of mass of the spring, and

$$\omega = \sqrt{\frac{k}{l_0 \hat{m}}} \quad (15)$$

$$\alpha = (x_2^n - x_1^n) - l_0 \quad (16)$$

$$\beta = \frac{1}{w} (v_2^n - v_1^n). \quad (17)$$

We choose the solution that produces velocities v_1^{n+1} and v_2^{n+1} closest to the analytic velocities.

Figure 3 (left) shows how using this new spring force allows a spring to exactly conserve its energy, oscillating back and forth about its restlength. Moreover this works for large time steps, and Figure 3 was computed using a time step equal to the framerate.

To handle constrained nodes, one should note that constrained nodes have infinite mass and a prespecified velocity. Given this, it is straightforward to handle constraints. In equation 9, any inverse mass terms corresponding to constrained nodes will become zero, and any velocity terms corresponding to constrained nodes will just be constants determined by the constrained velocity of those nodes. Figure 3 (right) shows how our new spring force correctly conserves energy even when one node of the spring is constrained.

2.2 Multiple Springs

When multiple springs are connected together, the equations from Section 2.1 are no longer correct, as the ΔKE

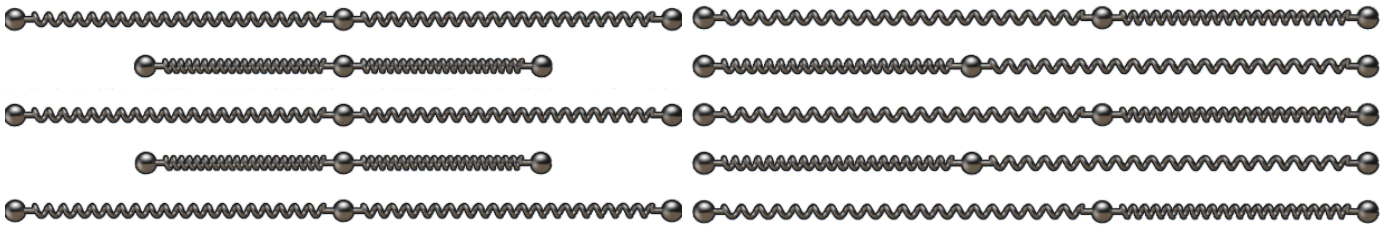

Fig. 4. (Left) Two unconstrained springs released from a stretched state and evolved using our new spring force, allowing them to continue to oscillate about their restlengths. (Right) Two identical springs are constrained on their non-shared endpoints, while the shared node is offset from the center. The shared node is then released and the springs are evolved using our new spring force, allowing the shared node to continue to oscillate back and forth. Note that we chose a time step equal to the size of the framerate for both simulations.

of a node with multiple incident springs cannot simply be attributed to the ΔPE of any single one of the incident springs, but must instead be shared among the ΔPEs of all the incident springs. If F is the spring force under consideration, and $\sum F_i$ is the sum of all *other* forces incident on node i, then ΔKE becomes

$$\Delta KE = \frac{1}{2}m_1\left[\left(\hat{v}_1^n + \frac{\Delta t}{m_1}F\right)^2 - (\hat{v}_1^n)^2\right] + \frac{1}{2}m_2\left[\left(\hat{v}_2^n - \frac{\Delta t}{m_2}F\right)^2 - (\hat{v}_2^n)^2\right]. \quad (18)$$

where

$$\hat{v}_1^n = v_1^n + \frac{\Delta t}{m_1}\sum F_1 \quad (19)$$

$$\hat{v}_2^n = v_2^n - \frac{\Delta t}{m_2}\sum F_2 \quad (20)$$

In a sense, we are just replacing v_i^n by $\hat{v}_i^n$. This simplifies to

$$\Delta KE = \frac{\Delta t^2}{2\hat{m}}F^2 + \Delta t(v_1^n - v_2^n)F + \Delta t^2\left(\frac{\sum F_1}{m_1} - \frac{\sum F_2}{m_2}\right)F. \quad (21)$$

However, this is still not completely correct, since if one tries to sum all the ΔKE terms from all the equations together, one would notice that the cross terms $\Delta t^2\left(\frac{\sum F_1}{m_1} - \frac{\sum F_2}{m_2}\right)F$ are double counted. As each cross term involves two forces, we simply split the cross term evenly between the two equations, to give the final ΔKE

$$\Delta KE = \frac{\Delta t^2}{2\hat{m}}F^2 + \Delta t(v_1^n - v_2^n)F + \frac{\Delta t^2}{2}\left(\frac{\sum F_1}{m_1} - \frac{\sum F_2}{m_2}\right)F. \quad (22)$$

The equation for ΔPE follows a similar logic when multiple springs are present, simply replacing occurrences of v_i^n in equation 4 with $\hat{v}_i^n$.

Now that we have adapted $\Delta KE = -\Delta PE$ to the multiple spring case, we would like to solve for the actual spring forces. However, each $\Delta KE = -\Delta PE$ equation contains multiple spring forces in it. Our approach to this problem is to simply iterate over all the springs in a Gauss-Jacobi fashion. Since each equation is associated with a particular spring, we just set all the other spring forces in that equation to whatever estimate for their value was obtained in the last iteration. This leaves a set of quadratic equations with only one variable spring force in each, which we can solve the same way we did for the single spring case.

It is possible that this iterative technique takes many iterations to converge, particularly for large systems of springs. This is where the second half of the Newmark integration scheme, the velocity update, comes in handy. In the velocity update, the same $\Delta KE = -\Delta PE$ equations are set up for each spring, but as the new positions are already known, ΔPE can be computed and plugged into each equation as a constant. This allows one to handle a position update that produces erroneous positions, whether that stems from non-convergence in the method we described above, or simply from using other less accurate methods for the position update. In particular, this means that *even if the position update is erroneous, one can fix the velocity to still conserve total energy.*

As the velocity update for multiple springs is solved with the same iterative approach as the position update, it is also possible that it may not converge. If so, this means that at the end of the time step energy is not exactly conserved, with too much or too little energy existing in the system. However, we can compute exactly what this residual energy error, which we will call PE_{res}, is for each spring as $PE_{res} = \Delta KE + \Delta PE$, where ΔKE and ΔPE are computed using the spring force from the last iteration of the solve.

PE_{res} can then be used in the next time step by adding it to the ΔPE term of the $\Delta KE = -\Delta PE$ equation for the spring it corresponds to. This can be thought of as slightly compressing or expanding the spring in the next time step to account for any energy error (hence why we call it potential energy residual). Keeping track of the energy error in this fashion allows one to fix the energy error in the system over time, so that the energy error does not accumulate. This process of computing and using PE_{res} is what we refer to as *energy budgeting*. Figure 4 shows how our new spring force works to conserve energy exactly for multiple springs, even with large time steps.

2.3 Gravity

The same ideas described above for energy conserving spring forces can be extended to other forces as well, including gravity. Consider a single particle in one spatial dimension which corresponds to the gravity direction

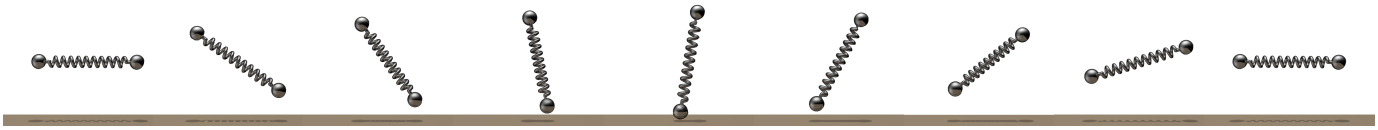

Fig. 6. A single spring rotating about its center of mass while conserving energy. The time step size is set to the framerate.

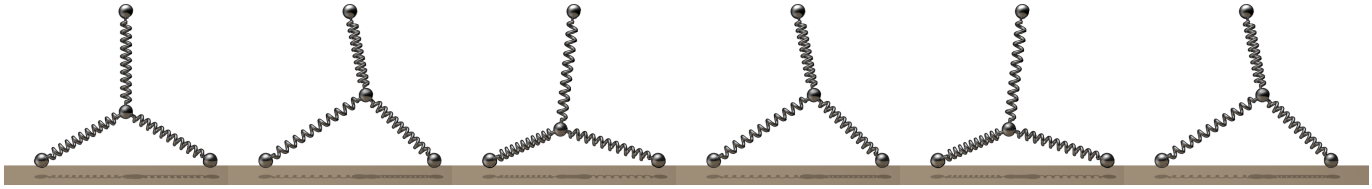

Fig. 7. Three springs constrained at their non-shared endpoints, where the shared node is given an initial velocity to the upper right. The shared node is then released and the springs are evolved using our new spring force, allowing the shared node to continue to oscillate back and forth. The time step size is set equal to the framerate of the simulation.

and where the only force on that particle is gravity. ΔKE can be written down as

$$\Delta KE = \frac{1}{2}m\left[\left(v^n + \frac{\Delta t}{m}F\right)^2 - (v^n)^2\right] \tag{23}$$

Recalling that the PE of a particle due to gravity is $-mgh$, where g is the (downward) gravity force and h is the height of the particle, ΔPE can be written down as

$$\Delta PE = -mgx^{n+1} + mgx^n \tag{24}$$
$$= -mg(x^n + v^{n+1/2}\Delta t - x^n) \tag{25}$$
$$= -mg\left(v^n + \frac{\Delta t}{2m}F\right)\Delta t \tag{26}$$

Then we simply solve $\Delta KE = -\Delta PE$ in the same manner we do for springs to get the gravity force. Gravity can be combined with springs using the same strategy for handling multiple spring forces as described in Section 2.2, replacing v^n everywhere with $\hat{v}^n$. Figure 5 shows a spring interacting with gravity while fully conserving energy.

Fig. 5. A single spring under gravity which is constrained at the top endpoint, released from a stretched state and evolved using our new spring force, allowing it to continue to oscillate about the restlength. The time step size of this simulation was set to the framerate.

2.4 Multiple Dimensions

Until this point, we have only dealt with forces in one spatial dimension. In particular, we have assumed that the spring force direction stays constant throughout the time step. Now we consider how the equations change in multiple spatial dimensions. We approximate the spring direction over the entire time step by the spring direction at the beginning of the time step. This means, that at the end of the update, when all forces are applied and new positions and velocities are computed, energy will not be exactly conserved. We can again utilize our energy budgeting framework, computing PE_{res} to keep track of this error to be fixed in subsequent time steps. This is highly desirable from the standpoint of efficiency as compared to a requirement of exact energy conservation every time step.

First, consider a single spring in isolation. ΔKE becomes

$$\Delta KE = \frac{1}{2}m_1\left[\left(\mathbf{v_1^n} + \frac{\Delta t}{m_1}F\mathbf{u}\right)^{\mathbf{T}}\left(\mathbf{v_1^n} + \frac{\Delta t}{m_1}F\mathbf{u}\right) - (\mathbf{v_1^n})^{\mathbf{T}}\mathbf{v_1^n}\right] + \frac{1}{2}m_2\left[\left(\mathbf{v_2^n} - \frac{\Delta t}{m_2}F\mathbf{u}\right)^{\mathbf{T}}\left(\mathbf{v_2^n} - \frac{\Delta t}{m_2}F\mathbf{u}\right) - (\mathbf{v_2^n})^{\mathbf{T}}\mathbf{v_2^n}\right] \tag{27}$$
$$= \frac{1}{2}\frac{\Delta t^2}{\hat{m}}F^2 + \Delta t(\mathbf{u^T}\mathbf{v_1^n} - \mathbf{u^T}\mathbf{v_2^n})F \tag{28}$$

where F is the magnitude of the spring force and $\mathbf{u}$ is the spring direction. Notice that equation 28 and the equivalent equation 3 for a single spring in one spatial dimension differ only in that all velocities are projected into the spring direction $\mathbf{u}$.

Now, consider what happens to ΔPE.

$$\Delta PE = \frac{1}{2}\frac{k}{\ell_0}\left[\left(||\mathbf{x_2^{n+1}} - \mathbf{x_1^{n+1}}|| - \ell_0\right)^2 - (||\mathbf{x_2^n} - \mathbf{x_1^n}|| - \ell_0)^2\right] \tag{29}$$
$$= \frac{1}{2}\frac{k}{\ell_0}(2\mathbf{b^T}\mathbf{a} + \mathbf{b^T}\mathbf{b} + 2cF\mathbf{a^T}\mathbf{u} + 2cF\mathbf{b^T}\mathbf{u} + c^2F^2 - 2\ell_0||\mathbf{a} + \mathbf{b} + cF\mathbf{u}|| + 2\ell_0||\mathbf{a}||) \tag{30}$$

where

$$\mathbf{a} = \mathbf{x_2^n} - \mathbf{x_1^n} \tag{31}$$
$$\mathbf{b} = (\mathbf{v_2^n} - \mathbf{v_1^n})\Delta t \tag{32}$$
$$c = -\frac{1}{2}\frac{\Delta t^2}{\hat{m}} \tag{33}$$

The presence of the magnitudes makes it difficult to rewrite ΔPE as a quadratic in terms of the spring force F. However, if we recall that equation 28 and equation 3 differed only in that all velocities and positions were projected into the spring direction, projecting the problem back into one spatial dimension, we will do the same for ΔPE. Therefore, we can simply use equation 4 for ΔPE, where c represents the same value, and $\mathbf{a}$ and $\mathbf{b}$ change to

$$\mathbf{a} = (\mathbf{x_2^n})^\mathbf{T}\mathbf{u} - (\mathbf{x_1^n})^\mathbf{T}\mathbf{u} \tag{34}$$

$$\mathbf{b} = \left[(\mathbf{v_2^n})^\mathbf{T}\mathbf{u} - (\mathbf{v_1^n})^\mathbf{T}\mathbf{u}\right]\Delta t \tag{35}$$

Though this is again an approximation of ΔPE, the energy error that occurs can be put into PE_{res} to be fixed in subsequent time steps. Figures 6 and 7 show how our approach conserves energy for springs in multiple dimensions.

3 More Complex Energy Budgeting

The method proposed in Section 2 focuses on energy conservation from a force based perspective in that one aims to solve for the force that gives exact energy conservation. While this force is simple to determine for a simple spring in Section 2.1, iteration was required for multiple springs in Section 2.2. Thus we introduce the concept of energy budgeting, where computing $PE_{res} = \Delta KE + \Delta PE$ accounts for errors in energy conservation. This is useful if one does not desire to perform Gauss-Jacobi iterations to convergence. Although this leads to non-conservation of energy in a single time step, those errors are accounted for and accumulated so that they can be accounted for in future time steps, producing a scheme that is energy conserving in the long run. We take advantage of this in Section 2.4 in order to approximate the direction of rotating springs as constant over a time step. In this section, we consider more complex meshes constructed from basic elements such as triangles and tetrahedra and explain the energy budgeting process for more complex altitude and bending springs. The intent is to give the reader a broader view of energy budgeting so that they may use it to account for their own favorite elements.

3.1 Altitude Springs

Following the work of [24] and [25], we use altitude springs when simulating triangles and tetrahedra for volume preservation. For simplicity, we will consider just triangle altitude springs, as the extension to tetrahedra is straightforward. An altitude spring is placed between each particle of the triangle, and a virtual node projected onto the plane of the opposite edge. An equal and opposite spring force is applied to the particle and the virtual node, and the virtual node distributes its forces barycentrically to the particles of the edge. As discussed in [26], the inverse mass and velocity of this virtual node are defined as

$$\frac{1}{m_v} = \left(\frac{w_1^2}{m_1} + \frac{w_2^2}{m_2}\right) \tag{36}$$

$$\mathbf{v_v} = w_1\mathbf{v_1} + w_2\mathbf{v_2} \tag{37}$$

where nodes 1 and 2 form the edge along which the virtual node lies, and w_1 and w_2 are the barycentric weights of those two nodes. If one considers a single altitude spring in isolation where the altitude goes from node 3 to the edge formed by nodes 1 and 2, then the equation for ΔKE becomes

$$\begin{aligned}\Delta KE =& \frac{1}{2}m_v\left[(\mathbf{v_v^{n+1}})^\mathbf{T}(\mathbf{v_v^{n+1}}) - (\mathbf{v_v^n})^\mathbf{T}(\mathbf{v_v^n})\right] + \\ & \frac{1}{2}m_2\left[(\mathbf{v_3^{n+1}})^\mathbf{T}(\mathbf{v_3^{n+1}}) - (\mathbf{v_3^n})^\mathbf{T}(\mathbf{v_3^n})\right] \qquad (38)\\ =& \frac{1}{2}\Delta t^2\left(\frac{1}{m_v} + \frac{1}{m_3}\right)F^2 + \Delta t((\mathbf{v_v^n})^\mathbf{T}\mathbf{u} - (\mathbf{v_3^n})^\mathbf{T}\mathbf{u})F \qquad (39)\end{aligned}$$

which we note is exactly what one gets with a normal spring, except that one of the endpoints is replaced by the weighted sum of two nodes. The equation for ΔPE is easily modified in a similar fashion. However, as noted in [27], if all three altitude springs in a triangle are used at once, one might encounter negative barycentric weights. To prevent these degenerate cases, we only ever use the shortest altitude spring, which also means we only need to keep track of one PE_{res} per triangle/tetrahedron. However, as the shortest altitude spring will switch throughout the course of a simulation, and because the restlength and Young's modulus may be different for each altitude, simply switching springs while using the same PE_{res} is not sufficient. To handle

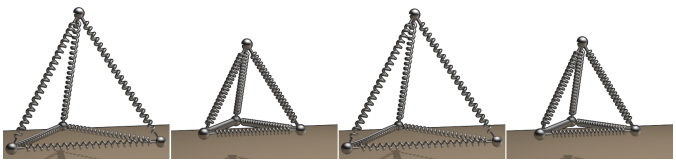

Fig. 8. A single tetrahedron with edge and altitude springs is stretched and then released. It is evolved using our new spring force, which allows it to oscillate back and forth continuously about its rest state. The time step size of this simulation is set equal to the framerate.

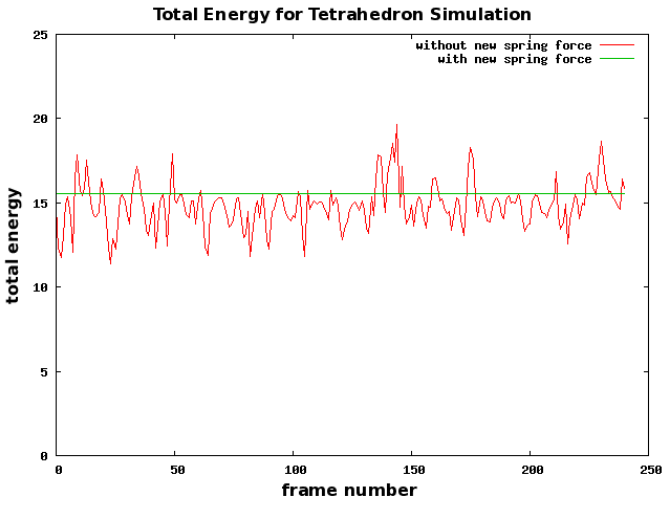

Fig. 9. A graph showing the total energy at the end of each frame in the simulation of a single tetrahedron with altitude springs shown in Figure 8. The green line shows the energy using our new energy conserving spring forces, and the red line shows the total energy if normal spring forces were used for that frame.

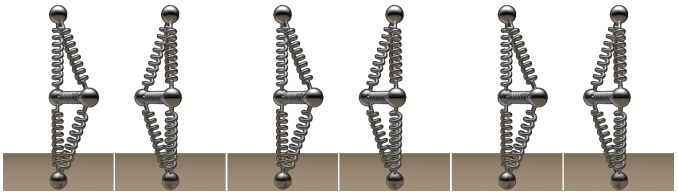

Fig. 10. Two connected triangles with edge, altitude, and bending springs are constrained at their non-shared nodes. The shared nodes are given an initial velocity, and the triangles are evolved using our new spring force, allowing the triangles to oscillate back and forth about their rest state. The time step size of this simulation is set equal to the framerate.

this, whenever we switch altitude springs in a triangle or tetrahedra, we update the PE_{res} associated with that element as follows

I. $PE_{total} = PE_{old\_altitude} - PE_{res}$
II. $PE_{res} = PE_{new\_altitude} - PE_{total}$

This ensures that energy is conserved when altitude springs are added and removed from the simulation. As future work, it would be interesting to try to adapt this scheme to a method that handles remeshing, such as [28]. Figure 10 shows a simulation with energy conserving triangle altitude springs, and Figure 8 shows the extension of this method to energy conserving tetrahedron altitude springs. Figure 9 shows the total energy at the end of each frame in the simulation, where the green line shows the energy using our new energy conserving spring forces, and the red line shows the total energy if normal spring forces were used for that frame.

3.2 Bending Springs

To simulate thin deformable materials such as cloth, we add a simple bending model composed of two springs to resist bending motion. One of these springs connects opposite vertices across an edge shared by two triangles and can be made energy conserving by simply considering what is done for normal edge springs, as it just connects two nodes in the mesh. However, if the two triangles become coplanar, this spring cannot recover the rest curvature as it lies in-plane with the triangles. A second axial bending spring is added to alleviate this problem by connecting a virtual node on the shared edge to a virtual node on the first bending spring.

To make the axial bending spring energy conserving, we apply a similar approach to handling altitude springs, except that instead of only one endpoint being a virtual node, both endpoints are now virtual nodes. However, one will notice that this axial bending spring is actually a zero-length spring. To handle this in our energy conserving framework, we first recall that the potential energy of a zero-length spring is

$$PE_{zero-length} = \frac{1}{2}\frac{k}{l_0}(l - \hat{l}_0)^2 \tag{40}$$

where the only difference with a normal spring is the use of the visual restlength $\hat{l}_0$ in addition to the normal restlength l_0, which prevents division by zero. Replacing l_0 by $\hat{l}_0$ in equation 4 allows us to handle zero-length springs in an energy conserving way. Figure 10 shows a simple example of these energy conserving bending springs. It is feasible that these ideas could be extended to other bending models, such as [8], [29], [30]. Moreover, we imagine one could extend these energy conservation techniques to other models based on mass-spring systems, such as the hard and soft constraints in [26] (see also [31]).

4 Our New Scheme

In Section 2 we outlined a force based iterative approach to conserving energy in a mass-spring system which relied on an energy budgeting process where exact changes in kinetic and potential energy were computed and the net change in energy was denoted $PE_{res} = \Delta KE + \Delta PE$. Using an iterative approach one can drive PE_{res} to zero, which is the typical goal of contemporary energy conserving time integration schemes. However, we instead exploited the ability to carry non-zero values of PE_{res} forward in order to simplify the iterative schemes in multiple dimensions in Section 2.4 and in order to cut down on the number of iterations required. In this section we propose our novel approach to energy conservation which allows one to set PE_{res} to zero *exactly* conserving energy by solving a simple quadratic equation requiring *no iteration*. In fact one could use their favorite time integration scheme with their favorite forces, including elasticity and damping(!), and simply insert their value of F into the formulas for ΔKE and ΔPE in order to compute PE_{res} for their scheme. Then our new global correction method (explained below) would allow them to exactly conserve the energy in post process for that scheme. Because one only needs to know PE_{res} for their favorite scheme, we dedicated Section 3 to a few more complex elements such as altitude and bending springs to give the reader some insight into how they might compute PE_{res} for various elements.

4.1 Motivation

First, for the sake of motivation, let us consider ether drag, which usually takes the velocity and scales it back based on some ether drag coefficient. Now, imagine if a mesh had too much energy as indicated by PE_{res}. One could simply add some ether drag to every node to slow everything down, such that the overall change in kinetic energy due to ether drag accounts for the total PE_{res} still present in the mesh. We write this down as

$$KE_{after} - KE_{before} = -\sum PE_{res} \tag{41}$$

$$\sum \frac{1}{2} m \left[(\mathbf{v^{n+1}})^{\mathbf{T}} \mathbf{v^{n+1}} - (\mathbf{v^n})^{\mathbf{T}} \mathbf{v^n} \right] = -\sum PE_{res} \tag{42}$$

where

$$\mathbf{v^{n+1}} = \mathbf{v^n} - \frac{\Delta t}{m} \epsilon \mathbf{v^n} \tag{43}$$

This simplifies to

$$\sum \Delta t \left[-\epsilon + \frac{1}{2}\frac{\Delta t}{m}\epsilon^2 \right] (\mathbf{v^n})^{\mathbf{T}} \mathbf{v^n} = -\sum PE_{res} \tag{44}$$

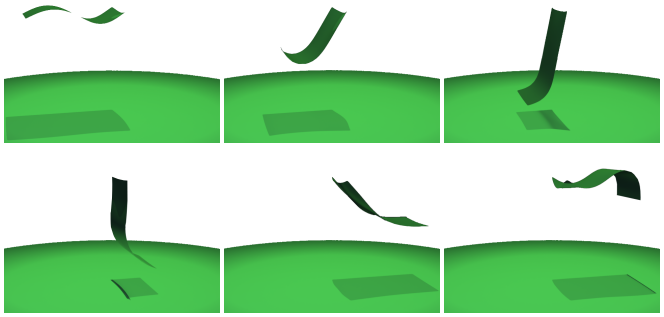

Fig. 11. A piece of cloth is constrained by two points and let go from its horizontal initial state. Our energy conservation framework enables it to continue swinging back and forth and achieve its initial height.

where ϵ is the global ether drag coefficient. This quadratic equation can be solved for ϵ, which produces an ether drag force on each node that will decrease the global energy to be closer to the correct amount. Of course one can only decrease the KE to zero, so there could still be PE_{res} left over to be fixed in the next time step, but applying this global correction will bring the total energy closer to the correct amount. If instead the mesh did not have enough energy, one could do the opposite of ether drag and increase all the velocities slightly to get the correct total energy.

Since ether drag does not conserve momentum we do not propose using it as a method to fix the global energy, but it is useful in conceptualizing the main idea. We will now introduce two momentum preserving forces that can be used in a similar way to correct the total energy while also conserving momentum.

4.2 Momentum Conserving Correction Forces

The motivational exposition in Section 4.1 for ether drag actually extends in a straightforward way to other forces that conserve momentum. The first force we will consider is the elastic spring force used in the standard velocity update. Replacing the elastic force for the ether drag force in equation 42 gives

$$\sum \frac{1}{2} m \left[(\mathbf{v^{n+1}})^{\mathbf{T}} \mathbf{v^{n+1}} - (\mathbf{v^{n}})^{\mathbf{T}} \mathbf{v^{n}} \right] = - \sum PE_{res} \tag{45}$$

where

$$\mathbf{v^{n+1}} = \mathbf{v^{n}} - \frac{\Delta t}{m} \epsilon \sum \mathbf{F_e} \tag{46}$$

This simplifies to

$$\sum \Delta t \left[-\epsilon (\mathbf{v^{n}})^{\mathbf{T}} \sum \mathbf{F_e} + \frac{\Delta t}{2m} \epsilon^2 \sum \mathbf{F_e^T} \sum \mathbf{F_e} \right] = - \sum PE_{res} \tag{47}$$

where $\sum \mathbf{F_e}$ is the sum of all elastic forces felt on a specific node. Again, we have a quadratic in ϵ, which we can solve in the same manner as when using ether drag. Using the elastic force instead of an ether drag force allows the global energy correction to be momentum conserving.

Instead of using the elastic force for the global correction, one could also use the damping force from the velocity update. One arrives at the same quadratic equation in ϵ as for the elastic force case, except with $\sum \mathbf{F_e}$ replaced by $\sum \mathbf{F_d}$, where $\sum \mathbf{F_d}$ is the sum of all damping forces felt by a specific node. As with using elastic forces, using a damping force for the global energy correction allows for momentum conservation.

Fig. 12. The same cloth as in Figure 11, except run using a small time step of the semi-implicit Newmark time integration scheme from [8]. (Left) No *physical* damping is added, causing the cloth to oscillate on the scale of the grid. (Right) A small amount of *physical* damping is added, alleviating the high-frequency oscillations while resulting in near-energy preservation, at the cost of a impractically small time step.

Note that what we are doing here is we have a momentum that we like, but we want a better total energy. This is similar to the energy correction in [32], where angular momentum is conserved but a new rigid body orientation is obtained that conserves energy. Through various tests, we found the best results are produced when damping forces are used in the cases where energy needs to be taken out of the mesh (something damping forces are known to be good for) and elastic forces are used in the cases where energy needs to be put back into the mesh. That is, we use damping when PE_{res} is positive and elastic forces when PE_{res} is negative.

4.3 A Note on Damping

It is important to keep in mind that we are not simulating a mass-spring network with the intention of simulating a mass-spring network, but rather in order to approximate what a deformable object would do as it experiences real-world forces. This is especially important for inviscid (undamped) models where a disturbance in the mass-spring network can excite frequencies at all scales, all the way up to the Nyquist frequency. This can appear rather disturbing, and one typically adds damping to remove energy at the highest frequencies.

Damping does two things in a simulation; it provides homogenization of the mass-spring network, limiting the highest frequency (or smallest scale) that visible vibrations occur and removes energy from the simulation. The first of these is a necessity for any high-fidelity simulation, whereas the second is undesirable, and our approach allows one to keep the first, desirable aspect of damping while discarding the second. At a high-level one can think of the first use of damping as a material parameter that can be used to adjust the look and feel of

a simulation, whereas the second use is strictly to remove energy.

Furthermore it is important to note that for a given simulation on a given mesh, damping can be minimized but not removed entirely. Take Figure 11, where we show that a highly deformable piece of cloth is able to recover its initial height when using our energy conservation framework. We reran it using a very small time step of the semi-implicit Newmark integration scheme of [8], which made the simulation take quite a long time to run. A time step this small could never be used in practice but we did it to minimize *numerical* dissipation as much as possible. Then to minimize *physical* dissipation we set the damping to zero; Figure 12 (Left) shows the result. The cloth mesh starts to oscillate on the scale of the grid - as we pointed out earlier in Section 1, this is similar to what is achieved by a variational method without damping. Energy is preserved but the simulation does not mimic an expected material behavior but rather shows frequencies particular to the scale of the mesh. If we add just a very small amount of *physical* damping, increasing it to .0001, we get the result in Figure 12 (Right), with rather nice energy preserving behavior. This is about the best we can do preserving energy in this example, using an impractically small time step and a hand-tuned, very small amount of damping. Now for a different mesh, simulation example, time integration scheme, and mass-spring model one might do better, but the issues are symptomatic. Numerical damping can be driven very low with impractically small time steps, and physical damping can be set very low but still needs to be big enough to "hide" the mesh discretization. This is the same conclusion that those doing variational methods come to without having stated it. One can think of our method in some ways as a better constitutive model for the mesh - one that does a better job modeling a real material without discretization artifacts while not requiring the removal of energy from the system (either with numerical or with physical damping).

4.4 Correcting Arbitrary Time Integrators

In summary, the key idea of our method is to compute a PE_{res} and then apply the correction forces in Section 4.2 to obtain energy conservation. Those correction forces can be used on any mesh, whether it be discretized with masses and springs or even finite elements. Whereas the correction forces are proposed in a mass-spring formulation, the original forces used in the simulation could come from a finite element model as long as PE_{res} is properly computed. In fact one could use a time integration scheme of their choice, whether it be implicit, semi-implicit, explicit, variational, symplectic, geometric, etc., and as long as PE_{res} can be computed via an energy budgeting process, one can use the correction forces in Section 4.2 to achieve energy conservation. Moreover, the original time integration method can include *both* elastic and damping forces, with damping being especially important because it gives aesthetic appeal to a simulation. In fact we chose the method from [8] with visually appealing damping parameters for Figures 2, 11, 13, 15, and 16. In each simulation, we simply computed PE_{res} in each time step using the methods outlined in Sections 2 and 3 and subsequently calculated correction forces along the lines of Section 4.2.

In Figure 14, the red line shows the results using the standard scheme from [8] to simulate the bouncing sphere shown in Figure 2. However, using our energy budgeting process to compute PE_{res} and then apply our correction forces leads to the green graph, where energy is obviously much better conserved. Note that while the fact that one can only decrease the KE to zero indicates that one cannot always exactly zero out PE_{res} every time step with the correction forces, any left over PE_{res} in each time step accumulates and is fixed in future time steps. As can be seen by comparing the green and red lines on the graph, the correction makes an enormous

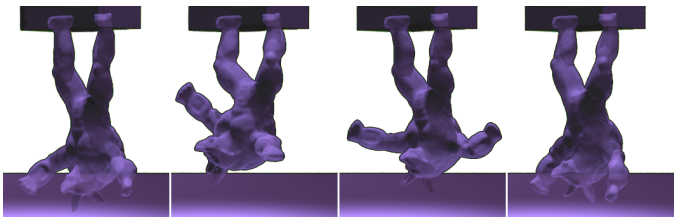

Fig. 13. A complex deformable armadillo is stretched and released with its feet constrained. Our energy conservation framework allows the armadillo to maintain its energy and continue bouncing over time.

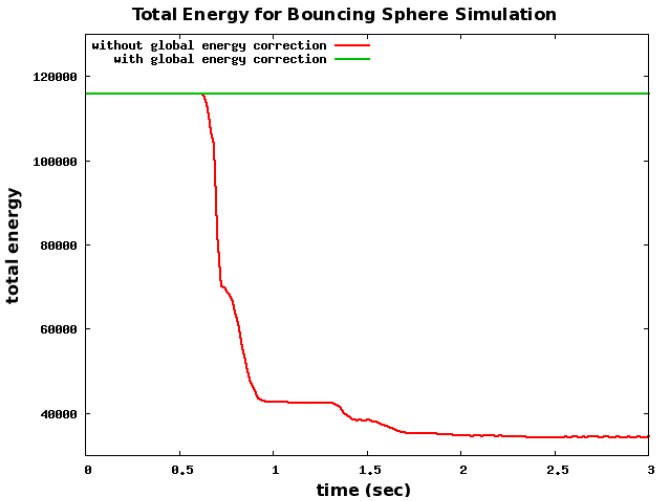

Fig. 14. A graph showing the total energy over time in the simulation of the coarse tetrahedralized sphere shown in Figure 2. The red line is the total energy when our energy conservation scheme is not used, whereas the green line is the total energy when utilizing our new scheme. The sudden declines in the red line correspond to collisions between the sphere and the ground.

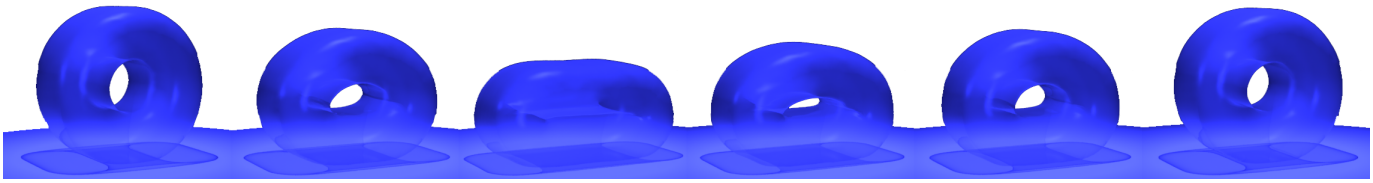

Fig. 15. A deformable wheel is dropped from a height and collides with the ground and itself, producing both rigid/deformable and self-collisions. Our global energy correction technique is used to restore all energy lost during evolution and collisions, allowing the wheel to bounce to its original height.

difference in the simulation and produces a line that appears constant to the naked eye.

5 Collisions

An obvious benefit of our energy budgeting process is that we can readily incorporate both collisions and self-collisions. Collisions are handled using the integration scheme

I. $\mathbf{v^{n+1/2}} = \mathbf{v^n} + \frac{\Delta t}{2m}\mathbf{F}$
II. $\mathbf{x^{n+1}} = \mathbf{x^n} + \Delta t \mathbf{v^{n+1/2}}$
III. Collisions: Modify $\mathbf{v^n}$ and $\mathbf{x^{n+1}}$
IV. Self-Collisions: Modify $\mathbf{v^n}$ and $\mathbf{x^{n+1}}$
V. $\mathbf{v^{n+1}} = \mathbf{v^n} + \frac{\Delta t}{m}\mathbf{F}$ (collision constraints in CG solve)

First, let us consider what happens during collisions with rigid bodies. In step III above, when a particle collides with a rigid body, its velocity is set to the velocity of the body, resulting in a change in kinetic energy. Moreover, the particle is moved to the surface of the body, resulting in a change in potential energy for all springs attached to that particle. If energy is to be conserved, the ΔKE due to collisions must equal $-\Delta PE$ due to collisions. Typically, energy is not conserved during collisions and often increases. We stress that stiffer objects and larger time steps result in a more drastic ΔPE due to collisions. As a result, large energy errors will be made, which could potentially result in unstable simulations. To ensure energy is conserved throughout the simulation, we compute the energy error $\Delta KE + \Delta PE$ due to collisions, and use our global correction from Section 4 to correct it. Even if one does not intend to use our algorithm for energy conservation at all, monitoring the energy difference during collisions and identifying any time steps where the energy increases would be important.

As in [6], [26], [33], we use collisions as constraints on $\mathbf{v^{n+1}}$ in the conjugate gradients solve during the velocity update in step V above. In this case, we would again like to determine the ΔKE due to using these collision constraints, as this will allow us to determine the energy gained or lost due to this part of collisions. However, as these constraints are enforced as projections in the solve, which includes elastic and damping forces as well, one cannot simply measure the kinetic energy before and after the solve. In order to determine the ΔKE due to just the collision constraints, we take the ΔKE computed after the CG solve, and subtract off the ΔKE that would have resulted by applying just the elastic and the damping forces. Once this is determined, any energy gained or lost can be corrected using the global correction from Section 4.

Figure 2 shows an example of an energy conserving simulation including collisions. We note our method makes it possible to restore only part of the energy lost to collisions, allowing an animator to explicitly control how much energy collisions remove from the simulation. It is important to note that similar energy conservation approaches can be applied to other collision handling schemes. For more on collision handling techniques, we refer the interested reader to the recent paper of [34].

5.1 Self-Collisions

As the amount of deformation in a mesh becomes more significant, it becomes important to handle self-collisions, which we handle using the method of [7]. We first note that self-collisions change both positions and velocities, resulting in changes to both PE and KE. Moreover, no energy should be lost to self-collisions, as all forces are exchanged between elements of the mesh. Therefore, to conserve energy, we simply measure both the kinetic and potential energy before and after self-collisions are applied and fix any error $\Delta KE + \Delta PE$ with the global correction described in Section 4. In Figure 15, we show an energy conserving simulation with self collisions.

We note that there are many other techniques for handling self-collisions, referring the interested reader to [35], and stress that similar energy conservation techniques could be applied to these other methods. In particular, [36] showed that under various situations, such as when cloth is pinched in the armpit of a character (between two bodies), it is sometimes desirable to allow interpenetration of cloth, which can be untangled afterwards. Note that this untangling requires changing of both positions and velocities of the cloth mesh, leading to changes in potential and kinetic energy. If these changes are undesired, they can be accounted for and corrected with our global correction method.

6 Examples

In Figure 2, we show a deformable sphere bouncing on the ground, and by using our global energy correction framework, it is able to recover its initial height while maintaining interesting oscillations due to the collision

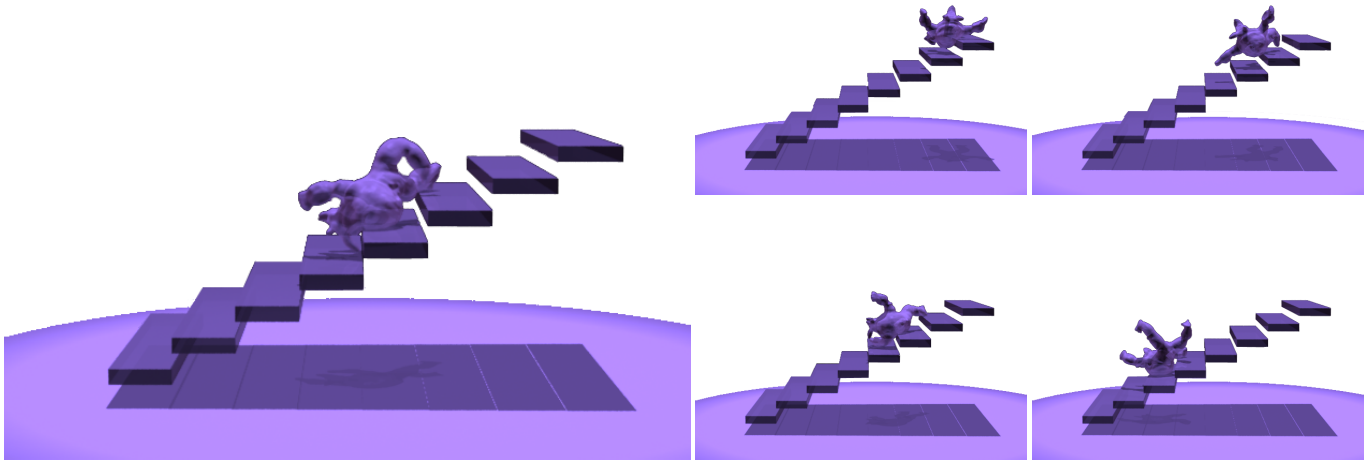

Fig. 16. A complex deformable armadillo model is dropped down a flight of stairs, exhibiting both rigid/deformable and self-collisions.

with the ground. We show that the same global energy correction can be applied to simulations with self-collisions in Figure 15, where we drop a deformable wheel on the ground. Even with both rigid/deformable and self-collisions, the wheel is able to regain its initial height. In Figure 11, we show that a highly deformable piece of cloth also is able to recover its initial height when using our energy conservation framework. It would be interesting to see if we could extend our methods, especially the energy conserving global fix in Section 4, to consider other models for cloth, such as the yarn based model seen in [37]. In the attached videos, a slight artifact is apparent in the bouncing sphere videos when the stiffness is high, causing the sphere to bounce off-center. We hypothesize that this is due to the discretization of the sphere, as it is noticeably worse when a coarse discretization is used, and when the sphere is not allowed to deform significantly, allowing the initial point of impact to play a significant role in determining the direction of the sphere's bounce. A similar artifact is seen in the video of the hanging cloth, where the swing of the cloth changes towards the end of the video. We again believe this bias is due to the coarseness of the discretization, with the cloth containing roughly only 5500 triangles. It would be interesting to see if increasing the resolution of the meshes and improving the symmetry of the discretization removes these simulation artifacts.

Finally, we show that our energy conservation framework works on complex meshes as well. In Figure 13, we stretch a complex armadillo mesh out and, while constraining its feet, we release it and let it bounce around. Using a typical deformable mesh solver, one would see this motion damp out over time; using our framework the armadillo continues to move around while conserving energy. In Figure 16, we drop the armadillo down a flight of stairs, showing that rigid/deformable collisions and self-collisions continue to work with complex meshes.

Energy budgeting, i.e. calculating PE_{res}, is rather straightforward and adds very little overhead to the cost of the simulation. Similarly, the calculation of the correction forces is also straightforward and approximately equivalent to one explicit time step. Overall in our simulations, the simulation time was dominated by the conjugate gradient solve to implicitly handle viscosity and our energy budgeting correction forces were similar in cost to the explicit part of the time step.

7 Conclusion

We introduced a novel technique for energy conserving simulation of deformable bodies regardless of the underlying time integration scheme. Our scheme allows the incorporation of collisions, self-collisions, and most importantly damping, which, as authors of other energy conserving integration schemes have pointed out, is crucial to the aesthetics of a simulation. However, unlike other energy conserving schemes where the addition of visually appealing damping necessarily removes the energy conservation property, our approach allows one to maintain energy conservation while using damping as a material parameter to improve the aesthetics of the simulation. Finally, the energy budgeting approach that our method takes allows it to be applied as a lightweight post-process energy fix to the time integration method of one's choice, allowing one to conserve as much or as little energy as is visually desirable.

One of the main limitations with the global correction scheme we proposed is that it does not provide control over where in the mesh the energy error is distributed. In future work, we would like to explore different momentum conserving correction forces similar to those presented in Section 4.2, but which allow the energy error to be distributed less uniformly or targeted to specific parts of the mesh. Furthermore, we are interested in exploring whether other less simplistic forces could be used with our global correction, and whether these forces produce more visually appealing results as compared to simply scaling the damping or elastic forces.

In Figures 2, 15, and 16, we show how our global energy correction works in the presence of collisions with kinematic objects. In future work, we would like to explore how our global energy correction extends to the interaction of multiple dynamic objects. Namely, one could keep track of the energy error due only to the collision of two bodies, and distribute that error back to the two bodies by applying an epsilon-scaled version of the original collision forces, much like the other global correction schemes we introduced in Sections 4 and 5. This will ensure that energy is conserved during multiple object interaction, while maintaining conservation of momentum.

Finally, we have only experimentally tested our global correction for a mass-spring based system using a semi-implicit Newmark integration scheme. Nothing in our global correction scheme is specific to mass-spring systems or semi-implicit Newmark integration, and in future work we would like to explore the generality of our correction scheme in the context of other solvers.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Craig Schroeder for his help with rendering. Research supported in part by NSF IIS-1048573, ONR N00014-09-1-0101, ARL AHPCRC W911NF-07-0027 and the Intel Science and Technology Center for Visual Computing. J.S. was supported in part by an NSF Graduate Research Fellowship.

REFERENCES

[1] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically deformable models," *Comput. Graph. (Proc. SIGGRAPH 87)*, vol. 21, no. 4, pp. 205–214, 1987.

[2] D. Terzopoulos and K. Fleischer, "Deformable models," *The Vis. Comput.*, vol. 4, no. 6, pp. 306–331, 1988.

[3] ——, "Modeling inelastic deformation: viscoelasticity, plasticity, fracture," *Comput. Graph. (SIGGRAPH Proc.)*, pp. 269–278, 1988.

[4] J. O'Brien and J. Hodgins, "Graphical modeling and animation of brittle fracture," in *Proc. of SIGGRAPH 1999*, 1999, pp. 137–146.

[5] G. Irving, J. Teran, and R. Fedkiw, "Invertible finite elements for robust simulation of large deformation," in *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2004, pp. 131–140.

[6] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *ACM SIGGRAPH 98*. ACM Press/ACM SIGGRAPH, 1998, pp. 43–54.

[7] R. Bridson, R. Fedkiw, and J. Anderson, "Robust treatment of collisions, contact and friction for cloth animation," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 594–603, 2002.

[8] R. Bridson, S. Marino, and R. Fedkiw, "Simulation of clothing with folds and wrinkles," in *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2003, pp. 28–36.

[9] M. Hauth and O. Etzmuss, "A high performance solver for the animation of deformable objects using advanced numerical methods," in *Computer Graphics Forum*, vol. 20, 2001, pp. 319–328.

[10] K.-J. Choi and H.-S. Ko, "Stable but responsive cloth," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 21, pp. 604–611, 2002.

[11] L. Kharevych, W. Yang, Y. Tong, E. Kanso, J. Marsden, P. Schröder, and M. Desbrun, "Geometric variational integrators for computer animation," *ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, pp. 43–51, 2006.

[12] I. Chao, U. Pinkall, P. Sanan, and P. Schröder, "A simple geometric model for elastic deformations," in *Proc. of ACM SIGGRAPH 2010*, 2010, pp. 38:1–38:6.

[13] P. Mullen, K. Crane, D. Pavlov, Y. Tong, and M. Desbrun, "Energy-preserving integrators for fluid animation," in *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, 2009, pp. 1–8.

[14] I. Chao, U. Pinkall, P. Sanan, and P. Schröder, "A simple geometric model for elastic deformations," Supplemental Materials Video, Proc. of ACM SIGGRAPH 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1778775

[15] R. A. LaBudde and D. Greenspan, "Energy and momentum conserving methods of arbitrary order for the numerical integration of equations of motion i. motion of a single particle," *Numer. Math.*, vol. 25, pp. 323–346, 1976.

[16] ——, "Energy and momentum conserving methods of arbitrary order for the numerical integration of equations of motion i. motion of a system of particles," *Numer. Math.*, vol. 26, pp. 1–16, 1976.

[17] C. Kane, J. E. Marsden, M. Ortiz, and M. West, "Variational integrators and the newmark algorithm for conservative and dissipative mechanical systems," *International Journal for Numerical Methods in Engineering*, vol. 49, pp. 1295–1325, 2000.

[18] A. Lew, J. E. Marsden, M. Ortiz, and M. West, "Variational time integrators," *International Journal for Numerical Methods in Engineering*, vol. 60, pp. 152–212, 2004.

[19] W. Fong, E. Darve, and A. Lew, "Stability of asynchronous variational integrators," *J. Comput. Phys.*, vol. 227, pp. 8367–94, 2008.

[20] D. Harmon, E. Vouga, B. Smith, R. Tamstorf, and E. Grinspun, "Asynchronous contact mechanics," in *ACM SIGGRAPH 2009*. ACM Press/ACM SIGGRAPH, 2009, pp. 1–12.

[21] M. Gonzalez, B. Schmidt, and M. Ortiz, "Force-stepping integrators in lagrangian mechanics," *International Journal for Numerical Methods in Engineering*, vol. 84, pp. 1407–1450, 2010.

[22] E. Hairer, G. Wanner, and C. Lubich, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer Series in Computational Mathematics, 2006, vol. 31.

[23] J. C. Simo, N. Tarnow, and K. K. Wong, "Exact energy-momentum conserving algorithms and symplectic schemes for nonlinear dynamics," *Comput. Methods Appl. Mech. Eng.*, vol. 100, pp. 63–116, October 1992.

[24] N. Molino, R. Bridson, J. Teran, and R. Fedkiw, "A crystalline, red green strategy for meshing highly deformable objects with tetrahedra," in *12th Int. Meshing Roundtable*, 2003, pp. 103–114.

[25] R. Bridson, J. Teran, N. Molino, and R. Fedkiw, "Adaptive physics based tetrahedral mesh generation using level sets," *Eng. w. Comp.*, 2005.

[26] E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw, "Hybrid simulation of deformable solids," in *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2007, pp. 81–90.

[27] A. Selle, M. Lentine, and R. Fedkiw, "A mass spring model for hair simulation," *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 64.1–64.11, Aug. 2008.

[28] M. Wicke, D. Ritchie, B. Klingner, S. Burke, J. Shewchuk, and O'Brien, "Dynamic local remeshing for elastoplastic simulation," in *Proc. of ACM SIGGRAPH 2010*, 2010, pp. 49:1–49:11.

[29] E. Grinspun, A. Hirani, M. Desbrun, and P. Schröder, "Discrete shells," in *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2003, pp. 62–67.

[30] P. Volino and N. Magnenat-Thalmann, "Simple linear bending stiffness in particle systems," in *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2006, pp. 101–105.

[31] C. Twigg and Z. Kačić-Alesić, "Point cloud glue: Constraining simulations using the procrustes transform," in *Proc. of the 2010 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2010, pp. 45–54.

[32] J. Su, C. Schroeder, and R. Fedkiw, "Energy stability and fracture for frame rate rigid body simulations," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2009, pp. 155–164.

[33] T. Shinar, C. Schroeder, and R. Fedkiw, "Two-way coupling of rigid and deformable bodies," in *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2008, pp. 95–103.

[34] J. Allard, F. Faure, H. Courtecuisse, F. Falipou, C. Duriez, and P. Kry, "Volume contact constraints at arbitrary resolution," in *Proc. of ACM SIGGRAPH 2010*, 2010, pp. 82:1–82:10.

[35] J. Barbič and D. James, "Subspace self-collision culling," in *Proc. of ACM SIGGRAPH 2010*, 2010, pp. 81:1–81:9.

[36] D. Baraff, A. Witkin, and M. Kass, "Untangling cloth," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 22, pp. 862–870, 2003.
[37] J. Kaldor, D. James, and S. Marschner, "Efficient yarn-based cloth with adaptive contact linearization," in *Proc. of ACM SIGGRAPH 2010*, 2010, pp. 105:1–105:10.

Jonathan Su received his Ph.D. in Computer Science from Stanford University in 2011, during which he was awarded the National Science Foundation Graduate Research Fellowship. He is currently a Research Scientist in the Parallel Computing Lab at Intel Corporation where he has been studying the scalability of physical simulation algorithms on current and future multi-core/many-core architectures.

Rahul Sheth received his B.S.E. in Electrical and Computer Engineering from Rutgers University in 2010. While there, he worked on various research projects in visualization and computer vision. He is currently pursuing a Ph.D. in Computer Science at Stanford University and is interested in developing algorithms for real-time physical simulation.

Ron Fedkiw received his Ph.D. in Mathematics from UCLA in 1996 and did postdoctoral studies both at UCLA in Mathematics and at Caltech in Aeronautics before joining the Stanford Computer Science Department. He was awarded an Academy Award from The Academy of Motion Picture Arts and Sciences, the National Academy of Science Award for Initiatives in Research, a Packard Foundation Fellowship, a Presidential Early Career Award for Scientists and Engineers (PECASE), a Sloan Research Fellowship, the ACM Siggraph Significant New Researcher Award, an Office of Naval Research Young Investigator Program Award (ONR YIP), the Okawa Foundation Research Grant, the Robert Bosch Faculty Scholarship, the Robert N. Noyce Family Faculty Scholarship, two distinguished teaching awards, etc. Currently he is on the editorial board of the Journal of Computational Physics, Journal of Scientific Computing, and he participates in the reviewing process of a number of journals and funding agencies. He has published over 80 research papers in computational physics, computer graphics and vision, as well as a book on level set methods. For the past ten years, he has been a consultant with Industrial Light + Magic. He received screen credits for his work on "Terminator 3: Rise of the Machines", "Star Wars: Episode III - Revenge of the Sith", "Poseidon" and "Evan Almighty".