Improved Integration for Cloth Simulation

D. Parks and D. Forsyth

Department of Electrical Engineering and Computer Science, U. C. Berkeley, Berkeley, California

Abstract

Most methods of cloth simulation produce animations that do not move very freely; the result often looks like the material is "swimming". The primary sources of this phenomenon are the non-physical dissipation terms often added to the mechanics to ensure stability in the simulation, and the numerical damping that is implicit in the choice of ODE integration schemes. By disposing of the additional dissipation terms and choosing a new integrator, the generalized-α method, *that was designed to safely integrate mechanical systems with extraneous high-frequency signals in their temporal component, we obtain more realistic results often with the about the same amount of computation.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

Recent advances in cloth simulation have produced attractive animations with a reasonable amount of computation. However, the results still fall short of physical realism. The motion of real cloth is often fast and jerky. The numerical methods that are usually employed to simulate cloth are not well suited to this sort of problem due to their inherent damping properties.

In order to see how this happens, we need to look at the core problem. All physically-based routines, regardless of whether they are particle or finite-element based, attempt to solve the standard discretized ordinary differential equation,

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \dot{\mathbf{x}}, t) \tag{1}$$

Here $\mathbf{M}$ is the (possibly lumped) mass matrix, the vector $\mathbf{x}$ contains the positions of the particles or, in finite-element codes, the nodes, of the mesh that represent the discretized cloth sheet. The function $\mathbf{F}(\mathbf{x}, \dot{\mathbf{x}}, t)$ is the sum of the conservative forces in the system, such as material stretch and gravitation, and the non-conservative forces in the system (such as friction or user-imposed constraints). We can therefore write

$$\mathbf{F}(\mathbf{x}, \dot{\mathbf{x}}, t) = -\frac{\partial E}{\partial \mathbf{x}}(\mathbf{x}) + \mathbf{F}^*(\mathbf{x}, \dot{\mathbf{x}}, t)$$

where E is the material's potential energy function and $\mathbf{F}^*$ represents the non-conservative forces.

The potential energy of cloth has two natural scales. Fabric is very strong in its plane ("stretching") – recall that the tensile strength of silk can be higher than that of steel. On the other hand, deformations out of the material plane ("bending") are virtually free. It can take orders of magnitude more work to stretch a small region of cloth than it does to cause a large fold. The result is that equation 1 is generally poorly scaled and this stiffness leads to serious issues impacting the numerical stability.

Until recently, most related dynamics research focused on solving equation 1 with *explicit integration* [4, 8]. In explicit integration, the state $(\mathbf{x}, \mathbf{v})$, where $\mathbf{v} = \dot{\mathbf{x}}$, is advanced using an explicit difference equation. This class of integration scheme is known to be unstable when applied to stiff and oscillatory problems. Conversely, most new research into the problem is based on using *implicit integration* [1, 3, 6] (additionally, some of the older research of Terzopoulos [11, 10] also used these methods). In these methods, the state at a new time step is a function of the state at both the old and new steps, or, in other words, the equation for advancing the state is implicit. Members of this class of algorithms typically supply an artificial amount of damping, which leads to a higher level of stability.

The implicit method most frequently employed to solve equation 1 is *backward Euler*,

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(\mathbf{y}_{n+1}) \tag{2}$$

where $\mathbf{y}' = \mathbf{f}(\mathbf{y})$. When applied to our problem, the result is the following set of difference equations:

$$\begin{aligned}\mathbf{x}_{n+1} &= \mathbf{x}_n + h\mathbf{v}_{n+1}\\ \mathbf{v}_{n+1} &= \mathbf{v}_n + h\mathbf{M}^{-1}\left(\mathbf{F}(\mathbf{x}_{n+1}, \mathbf{v}_{n+1}, t_{n+1})\right)\end{aligned} \tag{3}$$

Here, the subscript represents the time step and h is the length of time we are trying to integrate in this step. Techniques have been introduced in attempts to solve this system of equations quickly [3], but we will simply employ Newton's method.

In [12], the explicit midpoint rule and a fifth-order explicit Runge-Kutta method are compared with backward Euler when applied to cloth simulation. The authors' results suggest what we stated above; the mechanics of complex cloth dynamics are better suited to implicit schemes. They also observe, however, that the inaccuracy of backward Euler can lead to unrealistic results. This is further discussed in section 2.

1.1. Potential Energy

For ease, we have chosen the potential energy function defined in [1]. We could easily replace it with another form for deformation energy, but the difficulties mentioned above would be the same as they are inherent to the mechanics of the problem.

The function E is the positive-definite form $E = \frac{1}{2}\mathbf{C}^T\mathbf{C}$. The vector C contains terms for stretch, shear, and bend, all of which are zero in a state of no deformation. For each triangular face in our mesh, there is a stretch and a shear contribution, and for each pair of adjacent faces, there is a bend term. Defining the piecewise-linear function $\mathbf{w}(u,v)$ as the function mapping a parameterization of the cloth surface to its three-dimensional coordinates, the stretch and shear terms are defined as:

$$C_{stretch} = a\begin{pmatrix} \|\frac{\partial \mathbf{w}}{\partial u}\| - \|\frac{\partial \mathbf{w_0}}{\partial u}\| \\ \|\frac{\partial \mathbf{w}}{\partial v}\| - \|\frac{\partial \mathbf{w_0}}{\partial v}\| \end{pmatrix} \tag{4}$$

$$C_{shear} = a\frac{\partial \mathbf{w}}{\partial u}^T \frac{\partial \mathbf{w}}{\partial v} \tag{5}$$

where a is the triangle's undeformed area and $\mathbf{w}_0$ represents the function $\mathbf{w}$ for an undeformed sheet. Given two adjacent triangles with normals $\mathbf{n}_1$ and $\mathbf{n}_2$, we charge for bending in the sheet as

$$C_{bend} = \arccos(\mathbf{n}_1{}^T\mathbf{n}_2) \tag{6}$$

In order to simulate cloth, we scale these equations so that the penalty for bending out of plane is not nearly as harsh as that for stretching in plane. Additionally, an artificial term based on these functions is introduced in order to improve numerical stability in the integration (leading to a so-called *damping force*).

In the remaining sections of this paper, we will suggest a numerical integrator that is better suited our problem. In section 2, we discuss why backward Euler is a poor choice for solving equation 1. We show that the sort of damping that is inherent to the method is ineffective for our needs, which leads to the need for the additional damping force mentioned above. In section 3, we cover the generalized-α method [2]. We detail the ways in which it is an improvement over using backward Euler and discuss how its implementation can produce codes that run with the same amount of computation. Preliminary results comparing the two algorithms are given in section 4, and we discuss the remaining issues in section 5.

2. Damping in Backward Euler

In order to see how the backward Euler method damps oscillations, we will apply it to the single DOF second-order ODE

$$\ddot{x} + \omega^2 x = 0 \tag{7}$$

The solution to this problem is easily seen to be $x(t) = a\sin(\omega t) + b\cos(\omega t)$ for some constants a and b. For our purposes, we need a method that damps periods on the order of our time step. More precisely, a desirable integrator will provide little numerical damping when $\omega h < 1$ and more when $\omega h > 1$. Applying equation 2 to equation 7, we see that we can write the solution in the convenient form,

$$\begin{aligned}\begin{pmatrix} x_{n+1} \\ v_{n+1} \end{pmatrix} &= \mathbf{A}_{BE}\begin{pmatrix} x_n \\ v_n \end{pmatrix}\\ &= \mathbf{A}_{BE}^{n+1}\begin{pmatrix} x_0 \\ v_0 \end{pmatrix}\end{aligned} \tag{8}$$

where $\mathbf{A}_{BE}$ is the *amplification matrix* for the backward Euler method,

$$\mathbf{A}_{BE} = \begin{pmatrix} \frac{1}{1+\omega^2h^2} & \frac{h}{1+\omega^2h^2} \\ \frac{-\omega^2 h}{1+\omega^2h^2} & \frac{1}{1+\omega^2h^2} \end{pmatrix} \tag{9}$$

From equation 8, we see that the amplification matrix governs the level of damping caused by the method. Clearly, unless $\mathbf{A}_{BE}$ has all eigenvalues $|\lambda| <= 1$, the method will be unstable for increasing n. In addition, any eigenvalues $|\lambda| < 1$ will cause the system to damp, with $\lim_{n\to\infty} x_n = 0$.

The eigenvalues of $\mathbf{A}_{BE}$ are $\lambda_{1,2} = \frac{1}{1+\omega^2h^2}(1 \pm \omega h)$. A plot of ωh vs $|\lambda|$ is found in figure 1.

3. The generalized-α method

In order to improve on the low-frequency accuracy of the integration while maintaining the useful high-frequency damping, we would like an integrator that sustains its plateau near 1 when $\omega h < 1$ and drops off steeply when $\omega h > 1$. A natural conclusion would be to implement a higher order method and thereby obtain additional accuracy. This is the approach taken in [5], but the benefit typically comes with a computational cost – smaller time steps are required to enforce stability. Symplectic integration[9], which conserves the system's mechanical energy, requires an even more dramatic decrease

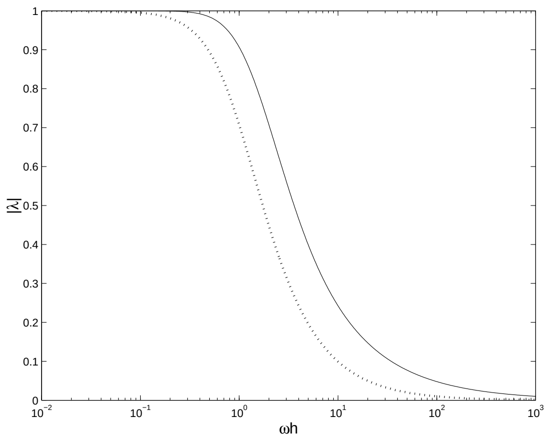


Figure 1: *A plot of ωh vs the magnitude of the largest eigenvalues of the amplification matrices ($\max|\lambda_i|$) for both backward Euler (the dotted line) and the high-frequency dissipation limit ($\rho_\infty = 0$) of the generalized-α algorithm (the solid line). The amplification matrices are found by applying each integrator to the single-DOF problem $\ddot{x} + \omega x = 0$ with time step h. To achieve the desirable properties of heavily damping frequencies that are high with respect to h, while minimizing the impact on low frequencies, an algorithm should show $|\lambda| \approx 1$ for small ωh and $|\lambda| \approx 0$ for large ωh. As the generalized-α response is always higher than that of backward Euler (at times the difference is a factor of four), we would expect that for given h, generalized-α to provide a much more accurate low-frequency response, but not damp high frequencies as aggressively. Our results suggest that the penalty paid for this damping loss is marginal while improvements in accuracy can be dramatic.*

in time step to remain stable. In order to retain the benefits of backward Euler, the chosen method should minimize any computational penalty.

The generalized-α method is a member of a class of integration schemes long popular in the mechanics literature because they possess some of these useful properties [2,7]. The method is specifically designed for second order systems and is second-order accurate *in the positions*; it is first order accurate in velocities. Its parameters are chosen in an attempt to associate dissipation with the higher frequencies, providing some numerical stability.

3.1. Implementation

The generalized-α algorithm begins with the user selecting a value for the high-frequency dissipation limit, $\rho_\infty = \lim_{\omega h \to \infty} |\lambda|$. $\rho_\infty = 1$ is no dissipation and $\rho_\infty = 0$ is maximum dissipation. The response of the generalized-α method with $\rho_\infty = 0$ is plotted in figure 1. Define[2, 13],

$$\alpha_m = \frac{2\rho_\infty - 1}{\rho_\infty + 1}, \quad \alpha_f = \frac{\rho_\infty}{\rho_\infty + 1}$$
$$\beta = \frac{1}{4}(1 - \alpha_m + \alpha_f)^2, \quad \gamma = \frac{1}{2} - \alpha_m + \alpha_f$$
$$\hat{\beta} = \beta\frac{1-\alpha_f}{1-\alpha_m}, \quad \hat{\gamma} = \gamma\frac{1-\alpha_f}{1-\alpha_m}$$

The nonlinear form of the generalized-α method, as detailed in [13] and applied to equation 1, is

$$\begin{aligned}\mathbf{M}(\mathbf{x}_{n+1} - \hat{\mathbf{x}}_n) - \hat{\beta}h^2\mathbf{F}(\mathbf{x}_{n+1}, \mathbf{v}_{n+1}, t_{n+1}) = 0 \\ \mathbf{M}(\mathbf{v}_{n+1} - \hat{\mathbf{v}}_n) - \hat{\gamma}h\mathbf{F}(\mathbf{x}_{n+1}, \mathbf{v}_{n+1}, t_{n+1}) = 0\end{aligned} \tag{10}$$

where $\hat{\mathbf{x}}_n$ and $\hat{\mathbf{v}}_n$ are functions of the prior state

$$\begin{aligned}\hat{\mathbf{x}}_n &= \mathbf{x}_n + h\mathbf{v}_n + h^2\left[\left(\frac{1}{2} - \frac{\beta}{1-\alpha_m}\right)\mathbf{a}_n + \frac{\beta\alpha_f}{1-\alpha_m}\mathbf{M}^{-1}\mathbf{F}_n\right] \\ \hat{\mathbf{v}}_n &= \mathbf{v}_n + h\left[\left(1 - \frac{\gamma}{1-\alpha_m}\right)\mathbf{a}_n + \frac{\gamma\alpha_f}{1-\alpha_m}\mathbf{M}^{-1}\mathbf{F}_n\right]\end{aligned} \tag{11}$$

Here the vector $\mathbf{a}_n$ represents a third component to the system's state,

$$\begin{aligned}\mathbf{a}_n = \frac{1}{1-\alpha_m}\left((1-\alpha_f)\mathbf{M}^{-1}\mathbf{F}_n + \alpha_f\mathbf{M}^{-1}\mathbf{F}_{n-1} - \alpha_m\mathbf{a}_{n-1}\right) \\ \mathbf{a}_0 = \mathbf{M}^{-1}\mathbf{F}_0\end{aligned} \tag{12}$$

Before attempting to solve equation 10, we can simplify it and hence cut the size of the nonlinear problem in half by observing that the vectors $(\mathbf{x}_{n+1} - \hat{\mathbf{x}}_n)$ and $(\mathbf{v}_{n+1} - \hat{\mathbf{v}}_n)$ must be parallel. We therefore define our new parameter $\phi = \mathbf{x}_{n+1} - \hat{\mathbf{x}}_n$ and reduce the system to one equation :

$$\mathbf{M}\phi - \hat{\beta}h^2\mathbf{F}\left(\phi + \hat{\mathbf{x}}_n, \frac{\hat{\gamma}}{\hat{\beta}h}\phi + \hat{\mathbf{v}}_n, t_{n+1}\right) = 0 \tag{13}$$

When solving this equation with Newton iteration, we use the previous state as an initial guess $\phi^{(0)} = \mathbf{x}_n - \hat{\mathbf{x}}_n$. Newton iterations require that we solve the linear system

$$\left(\mathbf{M} - \hat{\beta}h^2\frac{\partial\mathbf{F}}{\partial\mathbf{x}} - \hat{\gamma}h\frac{\partial\mathbf{F}}{\partial\mathbf{v}}\right)\Delta\phi^{(i)} = -\left(\mathbf{M}\phi^{(i-1)} - \hat{\beta}h^2\mathbf{F}\right) \tag{14}$$

and then set $\phi^{(i)} = \phi^{(i-1)} + \Delta\phi^{(i)}$. We currently use the preconditioned conjugate gradient linear solver detailed in [1]. In order to compare the method discussed here with the algorithm from that paper, our examples use one Newton step per iteration for both backward Euler and generalized-α.

4. Results

Our simulations were done using a flag made of a mesh of 116 points. In order to better compare the two algorithms, we do not add external forces, such as air drag or internal friction. Some of the animations may seem to imitate these effects, but they are artifacts of the numerical damping and therefore poor approximations to these physical properties. As we have seen in figure 1 that the degree to which the methods damp is a function of the chosen step size. Were

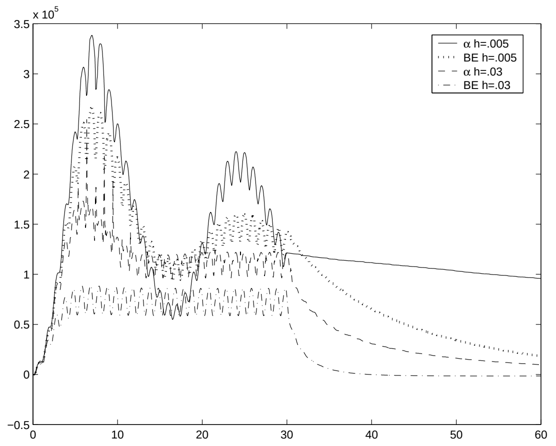


Figure 2: *A plot of the total energy over the course of the 60 second animations produced using backward Euler and generalized-*α*, each with step sizes* $h = .005s$ *and* $h = .03$*. In the simulation, the sheet is driven by moving two corner points in a sinusoidal motion. As seen in the figure, this creates two frequency components in the total energy (although as h increases, particularly with backward Euler, the lowest frequency component is squashed). At* $T = 30s$*, the moving constraints are stopped and the points are held static with a workless force. We fit the curves from* $T = 30s$ *to* $T = 60s$ *to the function* $\exp^{-\sigma t}$*; we would have energy conservation if* $\sigma = 0$*. Backward Euler and generalized-*α *with* $h = .005$ *result in* $\sigma \approx .069$ *and* $\sigma \approx .008$ *respectively, a 9-fold difference. With* $h = .03$*,* $\sigma \approx .358$ *and* $\sigma \approx .08$*, a factor of 6 difference. This is seen clearly in the animations in figures 3-6. Related results for backward Euler, implicit midpoint, and some explicit methods can be found in* [5].

the inherent numerical damping a fair replacement for drag or friction, it would not be such a strong function of the step size. It is clearly better to add these terms using actual physically-based approximations.

The flag (shown in figures 3-6) is driven by displacing the two top corner points in a sinusoidal motion with a period of 2s. After 30 seconds, the motion is stopped and the flag is held stationary at those two points. As this constraint is workless, perfect integration would cause the system to lose no energy once the motion had stopped. From the discussion above, it is clear that the system will instead suffer a steady loss of energy. This is seen in figure 2.

The amount of computation spent on generalized-α is on the same scale as backward Euler for a given time step. Both methods spend most of their time building the linear systems and solving them. As we have seen, there is some mild overhead involved in building the linear system for generalized-α, but we have found that it is offset by the improved convergence rate in the conjugate gradient solver; in the $\rho = 0$ high-frequency limit, the Jacobian matrix tends to be better conditioned than the system found in backward Euler.

As the methods share similar run times for a given time step, it is natural to ask how they perform with increasing time step. In our examples, with the increase in time step from $h = .005$s to $h = .03$s, both methods bleed energy in order to maintain stability. We see from the energy plots and simulations, however, that even with a large step size, the generalized-α simulation continues to act long after backward Euler has settled into a virtually static state.

5. Conclusions

The generalized-α integrator for second-order mechanical systems has been applied to the cloth dynamics problem. The method has been shown to provide results that are more accurate than that of backward Euler, due to its ability to damp high frequencies aggressively while respecting the more important low-frequencies. The implementation of the method has been simplified to the point where it takes little to no more work than backward Euler.

One avenue that we have yet to investigate is the impact of the ρ_∞ parameter. While the choice of ρ_∞ as zero seems natural for our needs of speed vs accuracy, but we know little about how the amount of work increases as the parameter is changed. We plan to look into the way that the work required by the algorithm changes as a function of this parameter.

References

1. D. Baraff and A. Witkin. Large steps in cloth simulation. In *Computer Graphics Proceedings, Annual Conference Series*, volume 32, pages 43–54. SIGGRAPH, 1998. 1, 2, 3

2. J. Chung and G. M. Hulbert. A time-integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method. *Journal of Applied Mechanics*, 60:371–375, June 1993. 2, 3

3. M. Desbrun, P. Schröder, and Alan Barr. Interactive animation of structured deformable objects. In *Graphics Interface*, pages 1–8, 1999. 1, 2

4. B. Eberhardt, A. Weber, and W. Strasser. A fast, flexible, particle-system model for cloth draping. *IEEE Computer Graphics and Applications*, 16(5):52–9, 1996. 1

5. M. Hauth and O. Etzmuß. A high performance solver for the animation of deformable objects using advanced numerical methods. In *Proc. Eurographics*, pages 319–28, 2001. 2, 4

6. M. Meyer, G. Debunne, M. Desbrun, and A. H. Barr. Interactive animation of cloth-like objects in virtual reality. *Journal of Visualization and Computer Animation*, 12(1):1–12, 2001. 1

7. L. R. Petzold, L. O. Jay, and J. Yen. Numerical solution of highly oscillatory ordinary differential equations. *Acta Numerica*, 6:437–83, 1997. 3

8. X. Provot. Deformation Constraints in a Mass-spring Model to Describe Rigid Cloth Behavior. *Proc. of Graphics Interface*, pages 147–54, 1995. 1

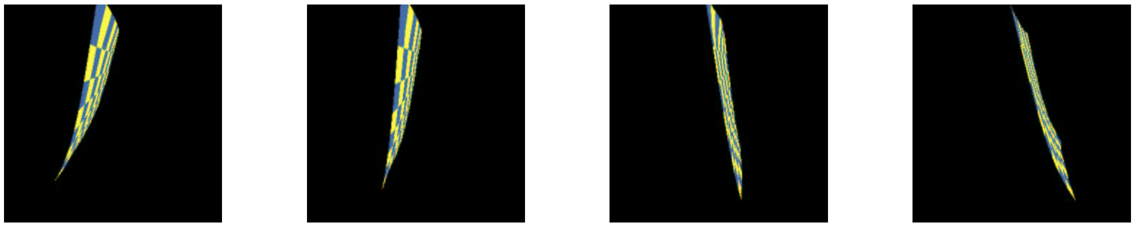
9. S. Serna and J. Calvo. Numerical hamiltonian problems, 1994. 2

10. D. Terzopoulos and K. Fleischer. Deformable Models. *The Visual Computer*, 4:306–31, 1988. 1

11. D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Computer Graphics Proceedings, Annual Conference Series*, volume 22, pages 269–78. SIGGRAPH, 1988. 1

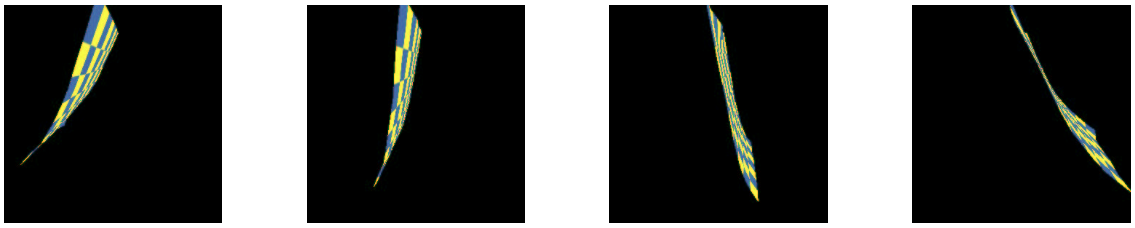
12. P. Volino and N. Magnenat-Thalmann. Comparing efficiency of integration methods for cloth animation. In *Computer Graphics International Proceedings*, pages 265–72, 2001. 2

13. J. Yen, L. R. Petzold, and S. Raha. A time integration algorithm for flexible mechanism dynamics: The DAE-α method. *Computer Methods In Applied Mechanics and Engineering*, 158:341–55, 1998. 3



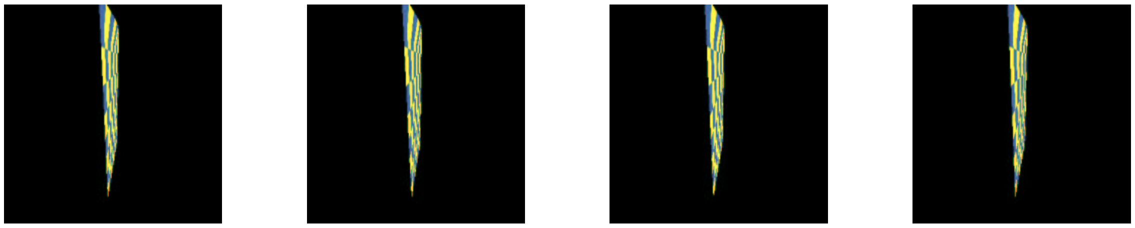
(a) T=56s (b) T=56.36s (c) T=56.72s (d) T=57.1s

Figure 3: *Frames from one-half of an oscillation of a waving flag simulated with no air drag or friction. The sheet was shaken for 30 seconds and then held. About 26 seconds later, much of the energy has been lost. This animation was done using* ***backward Euler*** *integration with a step size $h = .005s$. As the damping is a function of the integrator and the choice of step size, we judged the period of the oscillation by visually marking frames where the motion seemed to reverse direction. The entire video can be seen at http://www.cs.berkeley.edu/∼davidp/clothintegration.html.*



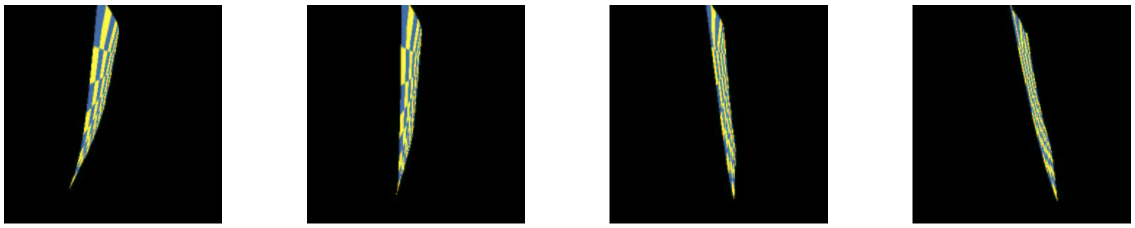
(a) T=56.7s (b) T=56.93s (c) T=57.16s (d) T=57.4s

Figure 4: *Frames from one-half of an oscillation of the same simulation but replacing backward Euler with the* ***generalized-***α *integration scheme. When compared with the frames above, both the amplitude and period of the swinging motion are more accurate. In addition, the curling of the sheet at the ends of the sequence are a noteworthy visual artifact missing in figure 3.*



(a) T=56s (b) T=56.36s (c) T=56.72s (d) T=57.1s

Figure 5: *Another 'half-period' of an oscillation of the same simulation, done using* ***backward Euler*** *with step size $h = .03s$. Due to heavy damping of low frequencies, there is no real period to analyze as the motion has effectively stopped.*



(a) T=55.9s (b) T=56.2s (c) T=56.5s (d) T=56.8s

Figure 6: *A half-period of an oscillation of the same simulation, done using* ***generalized-***α *with step size $h = .03s$. Despite a more than five-fold increase in the time step, the simulation resembles the one produced by backward Euler with $h = .005s$. The frequency of the swing remains more accurate.*