Better Cloth Through Unbiased Strain Limiting and Physics-Aware Subdivision

by

Konstantinos Dinos Tsiknis

B.Sc., The University of British Columbia, 2004

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

The Faculty of Graduate Studies

(Computer Science)

The University Of British Columbia

October 2006

Abstract

The ongoing challenge of cloth simulation in both interactive applications and film (where animators generally use iterative design, rerunning a simulation many times with different parameters) is speed. Current methods to allow large time steps do not have guarantees on stability and/or consistency with respect to physical parameters; we correct this with a new unbiased strain limiting scheme which is order and mesh independent. We also propose a physics-aware subdivision scheme which adds missing wrinkle and fold detail to coarse mesh simulations in a quasi-static way, making the use of faster and more easily-controlled coarse meshes more practical.

Contents

List of Figures

Acknowledgements

First and foremost I would like to thank my supervisor, Robert Bridson, for introducing me to the world of physical animation. His delightfully contagious passion for his work and tremendous enthusiasm for teaching made working with him both enjoyable and inspiring. Thanks as well to Wolfgang Heidrich for his insightful comments in the writing of this thesis. Special thanks to the Imager crew for great memories both in the lab and on the road: Tyson, Dave W., Chris, Vlady, Abhi, Trent, Derek, Ken, Llach, Aaron, Dave Y., Mike, Hagit, Kevin, anyone else I've forgotten, and especially Biff. Thanks to my brothers from other mothers, Wyatt and Steve for endless opportunities for distraction (and stress relief!). Lastly, but certainly not least, I'd like to thank my parents, without whose never-ending love and support all of this would not have been possible.

Chapter 1

Introduction

The proper portatrayal of characters on film screens, on television and in video games is essential to conveying a rich and complete story to the audience. The appearance of a character gives us instant visual insight into their personality and role in the story being told. Storytellers use a variety of visual cues to convey these characteristics to us as an audience. One very important visual cue is the clothing a character wears or interacts with and thus the ability to portray clothing and fabrics in a visually plausible manner is essential to delivering a meaningful performance. In environments where computer generated cloth is the only option this is usually not a trivial undertaking. Performing the animation of cloth such as this by hand can be a daunting task in certain situations. For these situations many productions choose to animate cloth through the simulation of the underlying physics that govern the behaviour of the fabric. Though animation of cloth through physical simulation takes some burden off the shoulders of the animator, it comes at a price.

The cost of simulation comes in several forms. A more obvious form is apparant when we speak about methods that demand a certain level of performance such as interactive applications or games. These types of systems are under strict constraints for frame rates and thus have strenuous requirements on computation time for a single frame and, consequently, the size of the timestep taken between state calculations. Generally, we start from an initial value state and move the system forward in discretized timesteps by approximating the (usu-

ally) non-linear analytic solution step by step with a time integration technique. Since we are dealing with approximations, all techniques that are used will have some degree of error. Certain techniques behave better than others and so can handle taking larger timesteps while still providing a stable solution.

Linearized implicit methods such as that of Baraff and Witkin [2] provide a framework for running simulations with larger time steps than explicit methods but still can be unstable above a certain unknown timestep size. Kane et al. [9] proved that fully nonlinear integrators can, in theory, guarantee unconditional stability. However, acquiring a solution from these techniques requires the solution of nonlinear systems, thus subjecting the system to the pitfalls surrounding the use of a Newton iterative solver, namely failure to converge for timesteps larger than an unknown threshold. In practice, this means instability is always a risk for large time steps, hence real-time applications tend to use ad hoc fixes such as strain limiting [25]. Unfortunately, these fixes are not convergent: they are order-dependent and do not reflect material properties. Order may not matter for an unconstrained cloth system with little or no applied forces however problems can definitely arise with strain limiting when constraints and external forces come into play. As Figure 1.1 shows, the order in which the limiting of springs in the 1 dimensional mass spring configuration propagates through the system can have a significant effect on the end result, severely altering the plausibility of the simulation. In a full 3 dimensional simulation with many more degrees of freedom and more constraints as well, the problem of choosing a proper direction to propagate is an even more difficult problem. Even if a good choice of direction is made, running the simulation with a different mesh can produce dramatically different results, and it can be difficult to determine how to adjust mesh structure and simulation parameters to achieve a certain look.

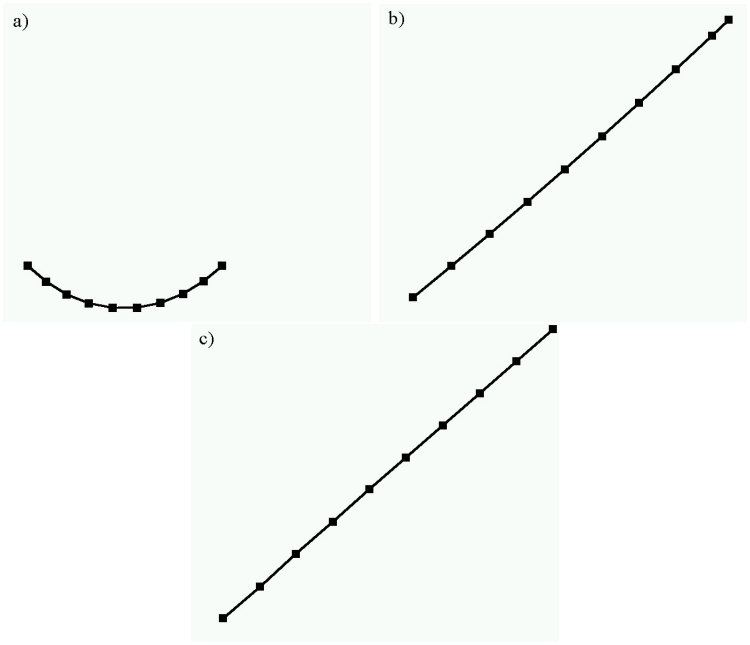

Figure 1.1: An example of the order dependence issues of standard strain limiting techniques. The top image (a) is a simple mass-spring chain in its initial configuration. The two endpoints are pinned and gravity is allowed to act on the rest of the points. In (b), the chain has been stretched by manipulating one of the constrained points. Because of the specific order of traversal, the springs are strain limited in an order that results in a non-physical result. (c) shows our method without order dependence. Note that each spring deforms a fraction of the total deformation of the chain, distributing the deformation in a more physically plausible way.

Given these issues, we propose a method which is applied as a post processor and can be bootstrapped with any reasonable time integration technique in order to provide an order and mesh independent solution while maintaining the stability of the system.

Taking the end state of the external time integration step as input to the system, we compute an initial guess of the internal impulse forces of the cloth system. We then apply these impulses element-by-element creating a disjoint set of elements that have moved independently from each other. The final solution is

stitched together as a least squares approximation to the set of individually time integrated elements. Element-by-element methods were pioneered by Hughes and Liu [19] for use in finite element analysis to break down complex stiffness matrices into products of smaller individual element matrices, though unlike our new method, suffered from stringent stability limits and ordering-dependent results.

The idea of stitching a model together from the individual solutions of its sub parts is also seen in the work of Sumner and Popovic [28] where the deformation of a source mesh is applied to a target mesh with different topology. The resulting deformation transfer is applied to each triangle of the target mesh separately resulting in a set of disconnected triangles much like the result of the second step of our time integration. The final vertex positions of the deformed target model are then calculated using an optimization technique.

We go back to our earlier discussion of the costs of physical simulation to look at a different, more intrinsic cost. Animating a cloth system is an iterative process. An animator generally will rig the cloth, run the simulation, observe the effects, tweak parameters as they see fit and restart the process. This is done over and over again until a satisfactory looking effect is attained. This process can be painstakingly slow, especially with systems ranging in the tens of thousands of points as is commonly seen in production environments. In order to speed up the cycle, one may want to use a lower resolution, coarsely discretized cloth system to define the movement of the cloth and revert back to a more finely discretized simulation once satisfied with the animation. The problem with this is that simply mapping animation parameters from a coarse simulation to a finer simulation does not automatically imply that the fine simulation will produce similar looking movement to the coarse simulation. In fact, the result will often look very different than the original animation since the dynamics of the system

can be sensitive to perturbations, particularly when collision and contact are involved.

To address this, our second contribution is a method for producing a high resolution cloth animation by simulating a coarse mesh and subdividing the cloth surface. Standard subdivision schemes do not provide the folding and wrinkling detail that is implied but not represented on the coarse mesh, which prompted ad hoc methods based on modulating wrinkly displacement textures [33] or sinusoids [20] according to the coarse mesh compression. This is intuitively the right approach, but is too far removed from the actual physics to provide plausible detail in general situations. We instead use a physics-based quasi-static method, similar in spirit to the flesh model of Teran et al. [29], where we compute a displacement map which minimizes the potential energy of the fine mesh. This can naturally produce correct buckling wrinkles and folds where the coarse mesh compresses. We note this provides a satisfying connection back to the original mechanical splines that inspired computational splines and subdivision schemes and so provides a physically sound but relatively inexpensive (from a simulation standpoint) method for representing cloth behaviour.

The remainder of this thesis is organized as follows. The following section discusses previous work in the area of physics-based deformable models and cloth animation schemes including methods for calculating collisions and proper models for the deformation of soft bodies and shells. Chapter 2 discusses our new time integration post processing technique for improved stability and order and mesh independence. Chapter 3 talks about our physics-aware subdivision scheme for more accurate control of cloth simulations and Chapter 4 discusses the results we have attained thus far and some future directions for further work in this area.

1.1 Related Work

Terzopoulos et al. [30] were the first to introduce physics-based deformable models to computer graphics using elasticity theory. Metaxas and Terzopoulos[23] then introduced the use of finite element methods to define the behaviour of these models. It was quickly realized that dense systems with many points were difficult and slow to manipulate and so attempts were made at using refinement based schemes. Thingvold and Cohen [31] modeled a control polygon with b-splines and refined the splines when more detail was desired. Debunne et al. [13] used an explicit finite element model applied to multiple dependent meshes of various resolutions, deciding to adaptively change resolutions at particular nodes based on both the local deformation and frame rate. Grinspun et al. [17] also used a finite element model but refined basis functions instead of elements to reduce simulation costs and Capell et al. [10] applied adaptive basis function refinement to a coarse volumetric model acting as a control lattice for an underlying high resolution mesh.

Time integration techniques are also still an area with continuing developments. Baraff and Witkin [2] provided a major stepping stone with the introduction of their implicit linear solver though their system induced excess damping. Further work with mixed implicit/explicit methods provided improvements [5, 6]. Other work has been focused on variational methods [9, 22, 27, 36] which has shown interesting results.

Focusing more on mass spring systems and with speed as the main goal, Desbrun et al. [14] make some approximations in their implicit time integration technique, achieving an increase in speed at the cost of some realism. Kang et al. [21] also make some approximations to speed up time integration of mass spring systems, using a coarser mesh and inducing finer wrinkles by using spline curves whose frequencies are adjusted according to spring compression to induce

wrinkles.

A number of articles deal with cloth collisions including [3, 6, 8, 15, 26, 35]. Bridson et al. [8] introduce a robust self-collision model and then Bridson et al. [6] present a cloth-object collision model that preserves cloth wrinkles during collisions. Baraff et al. [3] discuss a method for treating pinching of cloth between solid objects.

Several works have also addressed the issue of a proper (and efficient) model for the bending of cloth and deformable thin shells. Bridson et al. [6] provided an accurate physically based model capable of handling non-zero rest angles. Grinspun et al. [16] arrived at a similar model based on flexural energy and through the use of automatic differentiation. Cirak and Ortiz [12] propose an FE formulation based on the Kirchhoff-Love-type shell theories but only handle the static case. Thomaszewski et al. [32] extend this to the dynamic case using corotational subdivision elements to preserve the linearity of the strain calculations. Bergou et al. [4] use the assumption of isometric deformations to derive a bending model that is quadratic in positions, resulting in a constant Hessian and a reduction in simulation time. They rely, however, on the key assumption of inextensibility of the surface being modeled, which allows them to simplify the problem somewhat.

For a good overview of more contributions to cloth animation we refer the reader to the book of House and Breen [18], and the survey papers of Volino et al. [34] and Ng and Grimsdale [24].

Chapter 2

Unbiased Strain Limiting

Explicit time integration techniques can provide guaranteed stability given that the size of the timesteps do not exceed a certain threshold. This threshold tends to be quite small, forcing the calculation of several sub-steps per frame of animation. Attempts to increase the timestep size past this threshold will generally result in visible instabilities in the system. Even implicit solvers have a threshold above which stability is no longer a guarantee. The difficulties still lie in that we are trying to solve a large nonlinear problem with one or more linear steps. Our method takes a slightly different approach in that we attempt to break the problem down into a set of smaller subproblems, solve each of these in isolation and then find a global solution as close as possible to the combination of all the individual solutions.

Our system works as follows. First, external forces (such as gravity) are integrated forward in time, separately, using any desired method. The choice of method here is arbitrary, though to maintain convergence it should satisfy the CFL condition, which is guaranteed by any linearized implicit method for example. (Note that we use the CFL condition in the technically correct sense of ensuring numerical information propagates at least as fast as the physics dictates, a condition relating more to convergence rather than stability.) Our method is able to bootstrap itself to any time integration scheme for calculating the external forces as all that it requires are the resulting positions and velocities. These resulting positions and velocities are then fed into our system which is

composed of 3 steps: a global force estimate, element-by-element responses and, finally, a global stitching step.

2.1 Global Force Estimate

The first step is a standard step of time integration to enforce maxmimum strain constraints on edges in the mesh. Linearized Backwards Euler or, as in our examples, a linearized Lagrange multiplier system are both viable options here. We use the method of Baraff [1] for constraint satisfaction, employing constraints for limiting strain to a maximum amount (equation 2.1) as well as fixing the movement of any individual points we wish to constrain in the animation (equation 2.2).

$$c_i(\mathbf{x}) = \begin{cases} \kappa_{ij}(\frac{|x_j - x_i|}{L_{ij}} - (\epsilon_{max} + 1)) & \epsilon_{ij} > \epsilon_{max} \\ 0 & \epsilon_{ij} \leq \epsilon_{max} \end{cases} \tag{2.1}$$

$$c_i(\mathbf{x}) = |x_i - x_c| \tag{2.2}$$

In equation 2.1, κ_{ij} is the spring constant of the spring connecting mass points x_i and x_j, L_{ij} is the length of the spring at rest, ϵ_{ij} is the strain of the spring, and ϵ_{max} is the maximum allowable strain of the spring between these points. The constraint in equation 2.1 is enforced only along springs that are stretched beyond ϵ_{max}. In equation 2.2, x_i is simply a constrained point and x_c the target point in space that it is constrained to. Using these constraints, we solve the Lagrange multiplier system for a set of forces which satisfy these constraints. Essentially this means that if we assume a state $q^*(x^*_{n+1})$ after the forward time integration of external forces, we wish to apply impulses to enforce the constraints $\mathbf{c}(\mathbf{x^*_{n+1}}) = \mathbf{0}$. We do this by solving the following system for the

Lagrange multipliers, λ.

$$JM^{-1}J^T\lambda = -\mathbf{c}(x^*_{n+1}) \tag{2.3}$$

Where $J = \Delta t \frac{dC}{dx} + \frac{dC}{dv}$. The solution, λ, then gives us the impulse $\mathbf{i} = \mathbf{J^T}\lambda$ which then allows us to update the positions and velocities:

$$\begin{aligned} \mathbf{v_{n+1}} &= \mathbf{v^*_{n+1}} + M^{-1}\mathbf{i} \\ \mathbf{x_{n+1}} &= \mathbf{x^*_{n+1}} + \Delta t M^{-1}\mathbf{i} \end{aligned} \tag{2.4}$$

In our case we impose no velocity based constraints. Our only constraints are stretch and positional constraints. Thus $\frac{dC}{dv} = 0$, simplifying the Jacobian matrix calculation. This first step serves as an initial guess for the forces acting on each element (e.g. spring, or FEM triangle) for the timestep.

2.2 Element-By-Element Responses

The first step holds no guarantees of stability. The true solution to the constraint satisfaction problem posed in the global approximation step is inherently nonlinear. Since we take a linearized approach, approximating the solution with a single linear solve, it could very easily overestimate the elastic response of an element to the forces applied on it, which would lead to the simulation becoming unstable. Thus we proceed to our second step, where we stably compute the individual response of each element to the estimated forces. We purposefully use the nonspecific terminology of element here because our algorithm works independent of the basic element type. Given a definition of strain and strain rate for a particular element shape, we are able to calculate if the approximate forces from the linear solve in the previous step violate the desired thresholds set by the user. We use a value of 10% for strain limiting in all of our examples.

We then apply exactly the right impulse in response, limiting the strain and strain rate in a way that conserves linear and angular momentum. We do this on an element-by-element basis, applying the impulse forces to each individual element independently—the key observation is that computing the nonlinear response of a single element in isolation can typically be done exactly in an analytic fashion, guaranteeing stability.

2.2.1 Strain Limiting on Triangle Elements

We demonstrate our approach on an unstructured triangle mesh using triangle elements as our basic building blocks. In order to calculate the amount of strain on a triangle we observe the metric tensor of a triangle element. If x_i represents a point on a triangle in world space, and p_i represents that same point in the triangle's parameter space then the two coordinate spaces are related through $x_i = Fp_i + o$ where o is some translational offset and F is the deformation gradient. The metric tensor in 3 dimensional space can now be calculated as FF^T. We then perform an eigenvalue decomposition and take the square root of the largest eigenvalue, λ_0 and its corresponding eigenvector, e_0, to represent the largest component of the strain and the direction in which it acts respectively. This gives us a calculation of the strain of a triangle that is rotationally invariant. Now we take r_i to be a triangle vertex expressed in the local coordinate system of the triangle, that is $r_i = x_i - x_{cm}$ where x_{cm} is the triangle's centre of mass, and we define the unit length direction of strain, $\mathbf{d}$ (the eigenvector e_0 as a linear combination of the triangle's r_i's).

$$\mathbf{d} = \sum_i \beta_i x_i \tag{2.5}$$

This gives us the following update rule for limiting strain and strain rate in a momentum preserving way:

$$x_i^{new} = x_i + \alpha \frac{\beta_i}{m_i} \mathbf{d} \tag{2.6}$$

$$v_i^{new} = v_i + \gamma \frac{\beta_i}{m_i} \mathbf{d} \tag{2.7}$$

Where m_i is the mass at the i^{th} point and α and γ are computed as follows:

$$\alpha = \frac{\epsilon_{des}|q| - 1}{\sum_i \frac{\beta_i^2}{m_i}} \tag{2.8}$$

$$\gamma = -\frac{\sum_i \beta_i v_i \cdot \mathbf{d}}{\sum_i \frac{\beta_i^2}{m_i}} \tag{2.9}$$

Given the above formulation it is also possible to model anisotropic fabrics by defining the strain limit as a function of the strain direction in parameter space, q. This way, elements could be allowed to stretch a greater amount in one direction than another. In fact, this direction dependent strain limiting could be defined differently for certain elements allowing the proper modeling of stretching around seams in a garment where the stiffness tends to be different than other areas.

As previously mentioned this method works for any type of element composition. Since the strain direction is broken down into a linear combination of the vertices of the element shape, we can calculate appropriate weights for an aribtrary number of vertices, enabling the same formulas to be used for 1 dimensional springs and extended for use with more complex elements such as tetrahedra. For a complete derivation of the above formulas refer to Appendix A.

2.3 Global Stitching

Because we work on each element in isolation, it is almost certain that elements will disagree on the final position of a shared node and so the system will become disjoint at this point. We are left with a set of separated elements which have each moved in isolation and in accordance only with the local forces acting on them directly. The only task that remains is to rejoin all the elements back together to form a single surface again. The third step reconnects all of the individual elements, stitching the system back together with a linear least squares solve. Though it would be easier to simply take the centroid of all of a node's end positions from the previous step, this would end up violating the stretch constraints which we were trying to enforce in the first and second steps. Thus, instead of minimizing over differences in node positions, we minimize over differences in edge vectors (equation 2.10), resulting in a final solution which does a better job of maintaining the stretch constraints.

$$\min \|\mathbf{c}\|^{\mathbf{2}} \quad \text{where} \quad c_i = [(x_i - x_j) - (\tilde{x_i} - \tilde{x_j})] \tag{2.10}$$

In equation 2.10, x_i and x_j are two mass points connected by springs and $\tilde{x_i}$ and $\tilde{x_j}$ are the corresponding endpoints for edge i as determined by the element-by-element responses. Note that this is just a linear least squares problem, which can easily be solved exactly, unlike the nonlinear equations that existing fully implicit time integrators produce.

Figure 2.1 shows an example of our method broken up into the three major steps and being applied to a mass spring system consisting of a single chain of mass points connected together in succession.

This three-step approach is physically correct and order-independent (unlike standard strain limiting) because we compute a valid global estimate of forces

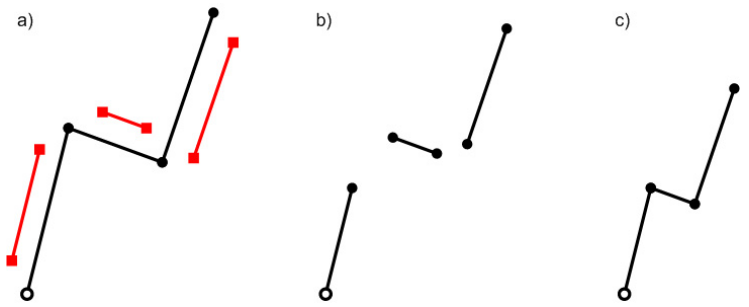

Figure 2.1: a) Mass spring system with 1 constrained point (white point). Red lines (with square endpoints) show the maximum allowable lengths of each spring. At this point an initial guess of forces is calculated. b) The impulse forces from step 1 are applied to each spring individually. Shared nodes disagree on their final positions since they are being pulled in different directions. c) Step 3 stitches the system back together, respecting the constrained points and lengths of springs.

and correct element-by-element responses. It is guaranteed to be stable since the element computations are stable and the global stitching step confines nodes to the convex hull of the element responses and is linear, hence there are no convergence issues.

Though the element-by-element solve in step 2 is a general formulation and works for all shapes of elements with arbitray numbers of vertices, this is unfortunately not the case for the first and last steps. These are still formulated in terms of edge constraints. For example, the Lagrange system in step 1 produces a force prediction based on the strain along edges and the linear least squares solve of step 3 also calculates an approximation that is based on length constraints along edges. Though the algorithm can still handle all types of shapes since all forms of element shape can be broken down into edges, it is not yet fully unbiased towards mesh topology. We are still working on a formulation for the global force approximation and global stitching steps that is mesh independent but we are quite confident a solution is well within reach.

Our framework does allow for flexibility in the details of each step. An alter-

native approach to the first 2 steps is also possible using linearized force curves instead of constraints for the first step and corrective impulse force responses for the second step. As long as the second step ensures a stable response to the initial global force approximation, the method is guaranteed to retain its stability.

We integrate this with the robust collision system of Bridson et al.[8], with an additional improvement in the handling of rigid impact zones, building on a suggestion in [7]. Previously the impact zone approach projected out the velocity of a zone of colliding elements to the space of rigid body translations and rotations, thus eliminating any possible collisions. However, we can enrich the space to include shearing as well and still make the no collision guarantee (in [7] Bridson suggests the more general space of affine motions, but we note that including scalings allows for reflections, which of course do cause collisions). Allowing the three additional shearing components can reduce the amount of energy dissipation in the projection, allowing for cloth surfaces to slide over one another instead of rigidly sticking for example. It also slightly simplifies the calculation, since the generalized inertia tensor is more sparse than the standard rigid body inertia tensor.

Chapter 3

Physics Aware Subdivision

Recall that subdivision is a model for splines which originated as the actual mechanical equilibrium of a piece of thin wood or metal subjected to point constraints, where its combination of elastic and bending energy found a (local) minimum. We bring this idea back to computer graphics, defining physics-aware subdivision as a solution to quasi-static equations of motion based on elastic and bending energy of the surface. We can show with a few steps that the minimization of these energies leads directly into the derivation of the equation for cubic splines. For more detail on this connection see Appendix B.

Our system is based on two representations of the cloth surface mesh at different levels of detail. We take a coarse mesh representation of a cloth system and subdivide it with a Loop subdivision scheme to a desired resolution. A displacement map is then layered on top of the subdivided mesh, displacing each point a certain distance in the normal direction to the surface.

Carrying on with our analogy to splines, we envision that for a given set of control points there exists one, or often many, plausible resultant smooth curves. Subdividing our coarse control polygon along the edges between control points we can grow each of these points out from the polygon surface in the normal direction. Each configuration of displacements has a potential energy. The natural tendency of cloth is to fold into a shape that minimizes the potential energy. We mimic this behaviour by optimizing over the space of displacements to find a local minimum of the potential energy and move to it at the end of

every timestep. After subdividing the coarse mesh at the beginning of each timestep, the points on the fine mesh are never moved again for the duration of the timestep. Only the scalar displacement values at the fine points change.

$$PE_{elastic}(\mathbf{d}) = \frac{1}{2}\kappa_{elastic} \sum_{springs_c} (\sum_{j=1}^{m-1} |\tilde{x}_{j+1} - \tilde{x}_j| - L_c)^2 \tag{3.1}$$

$$PE_{bending}(\mathbf{d}) = \frac{1}{2}\kappa_{bend} \sum_{i=1}^{n_f} \left((\sum_{j=1}^{m} d_j - m d_i)^2 \max\{-\epsilon_c, 0\} \right) \tag{3.2}$$

We define an energy for the displacement surface that takes into account the change of lengths from the undeformed surface as well as the bending. The elastic potential energy, shown in equation 3.1, is defined as the squared difference between the rest length of an edge on the coarse mesh (L_c) and the sum of the lengths of the corresponding subdivided edges on the subdivision surface where $springs_c$ is the set of springs in the coarse system, m is the number of subdivision mesh points lying on the particular coarse mesh edge and $\tilde{x}_j$ is the jth point on the physically subdivided mesh, that is $\tilde{x}_j = x_j + d_j\hat{n}_j$ if d_j is the displacement at point x_j in the normal direction $\hat{n}_j$. The motivation behind this being that since cloth has high resistance to in-plane deformation and low resistance to deformation in the normal direction, it prefers to buckle rather than compress when forced together at two points. Thus, the amount of material between two points or on a certain area of the surface in any buckled formation should remain the same as it would be at rest. In future work we will more correctly phrase this in terms of the metric tensor in each triangle face, rather than along the edges, to avoid mesh-dependent artifacts.

The second energy term serves to resist bending to a certain degree with the idea being that the surface is always trying to return to as smooth and flat a rest state as possible. It is defined in equation 3.2 as the perpendicular distance of a point to the least squares plane of its surrounding (1-ring) neighbors. Here d_i

is the displacement of the i^{th} point while d_j is that of each of its m neighboring points and κ_{bend} is the coefficient for bending stiffness. It is important to note the term ϵ_c on the end of equation 3.2 which scales the bending energy at each point by the strain of the corresponding triangle it is associated with on the coarse mesh. For points which sit on the boundary of more than one coarse triangle, we simply take an average of all surrounding triangles of that point. Intuitively, the bending and elastic energies attempt to drive the displacements in different directions under compressive forces. The area preserving elastic energy term will try to push the displacements away from the surface and the bending term will attempt to minimize the curvature at each point, keeping them as close to the surface as possible. At small levels of compression there is a point up until which the quasi-static bending forces are strong enough to overpower the forces driving the displacements outward. At this pivotal point there is a discontinuous bifurcation in the locus of quasi-static local minima: in other words, the cloth pops. This is nothing more than the physically correct buckling instability as discussed in the work of Choi and Ko [11] with respect to squeezing a solid rod at its two ends. To avoid objectionable (though correct, in one manner of thinking) popping, we perturb the physics to make the locus of local minima continuous, so as an edge compresses the quastistatic solution smoothly begins to crumple.

The minimization of the potential energy is performed using the Gauss-Newton method with each linear solve being done with a conjugate gradient solver. At each iteration, after solving for the change in displacements, we use a simple backtracking algorithm along the direction of the Gauss-Newton step in order to find a better solution. This is just a simplification of a line-search. A conventional line-search method here would be a better option for improved performance. The solution of displacements found at each time step

preserves the surface area of the mesh as much as possible while choosing the smoothest shape possible for the given configuration of the coarse mesh, creating the folding and buckling features one would expect of a finely discretized cloth mesh. We begin each solve with the previous frame's solution, so in the presence of multiple local minima we tend to avoid popping.

Our system is easily combined with collision aware subdivision [8] in order to handle collisions that occur as a result of the change in the displacements during the quasistatic timestep. Because of the quasistatic nature of the displacement calculations, we need not worry about repulsion velocities for collisions with the fine mesh points. We first test for intersections at the full step $d_{t+1} = d_t + \delta d$. If intersections are detected at this point we scale back the displacements to the point where the mesh is intersection free.

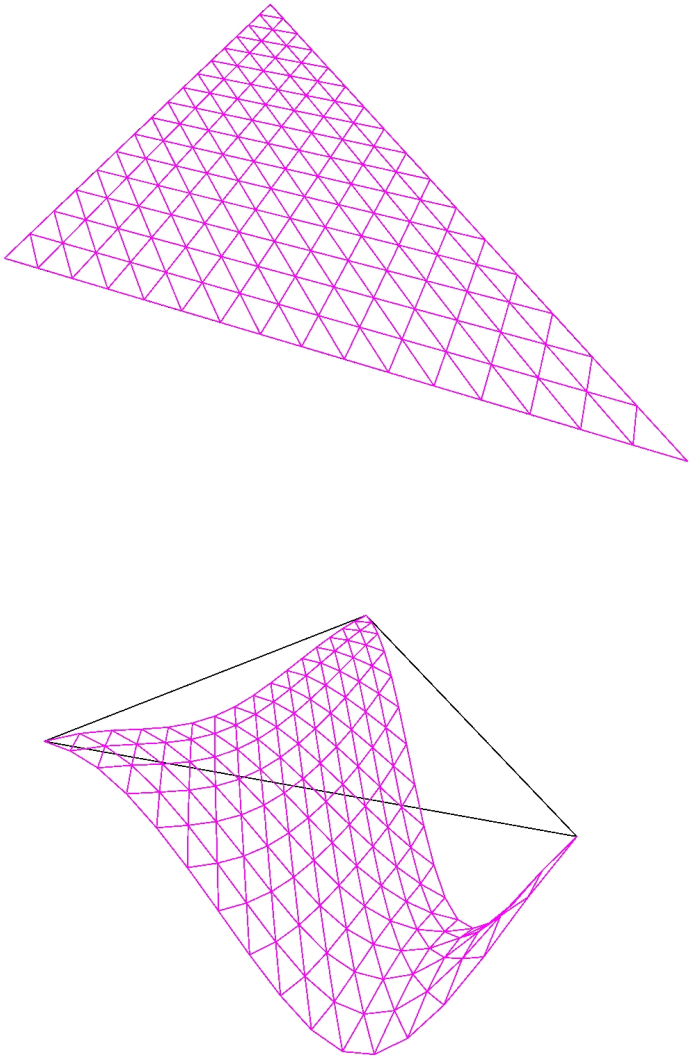

Figure 3.1: Demonstration on a single triangle. As the coarse triangle compresses, the subdivided triangle buckles outward into a configuration that minimizes the potential energy of the system. An equilibrium is found between the requirement for a smooth surface and one that preserves surface area.

Chapter 4

Results and Future Work

Our system is able to successfully model the buckling and folding of a finely discretized mesh while keeping the motion true to the underlying coarse mesh. Figure 4.1 shows a 2-dimensional representation with a chain of mass points. It can be seen that in places where the coarse mesh compresses, the physics-aware subdivision mesh buckles outward. Note as well that even as the coarse mesh wraps around itself, the use of collision-aware subdivision ensures that an intersection-free solution always results.

Figure 4.2 shows a few frames of a full 3-dimensional implementation, showing first the coarse mesh which defines the motion (blue), the undeformed Loop subdivision mesh (green) and, finally, the fine mesh with physics aware subdivision applied (purple). As triangles deform in the coarse mesh, folds begin to appear in the physically subdivided fine mesh.

The coarse mesh consists of roughly 100 triangles, enough to be easily manipulated and animated interactively. The resulting fine mesh displays detailed folds and wrinkles that ordinarily would be impossible to show with a mesh of such low resolution.

Certain mesh artifacts are visible at times along edges of triangles in the coarse mesh when there is significantly more deformation in one direction. This is a result of the fact that in our current implementation, the calculation of the area preservation term is only done along the edges of triangles. Thus, the buckling shape is driven by the points on these edges. The proper way to handle

this is to use metric tensors to measure the deformation across the entire area of each triangle. The use of tensors gives us a smooth deformation calculation across the entire face of each triangle allowing for a more precise displacement calculation for points on the interior of a triangle. The result should be a more consistent and convincing looking fold without any mesh artifacts.

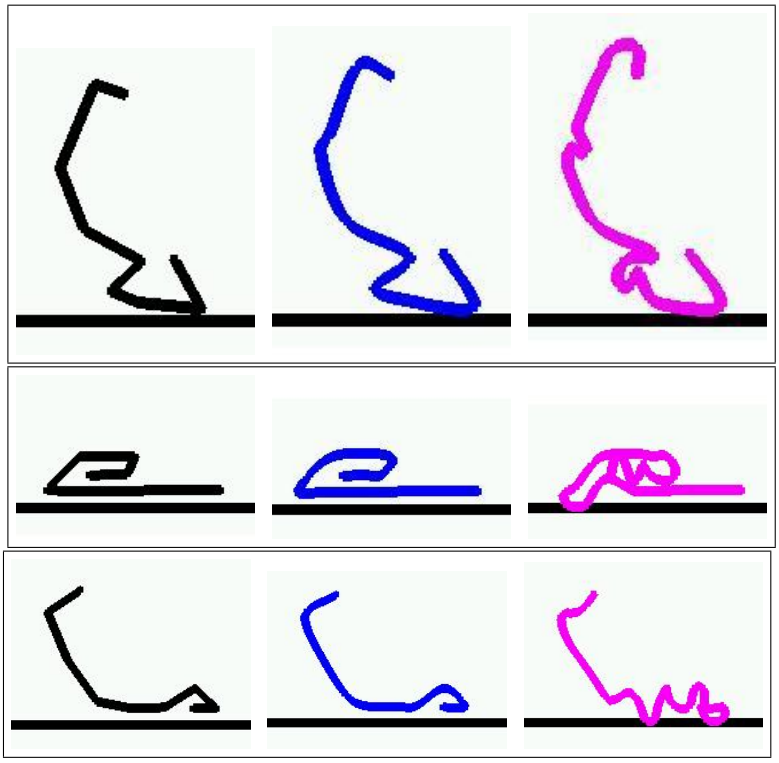

Figure 4.1: A set of 3 frames of a single chain of masses and springs. Each outlined row of images represents a single frame of animation. In each box, the left-most lines represent the coarse mesh, the centre lines the subdivision mesh, and the right-most the fine mesh with physics-aware subdivision. Note that the physics-aware subdivision mesh collides with the ground as well as itself. Different collision threshold distances have been used for the coarse and fine mesh collisions for clarity of display only.

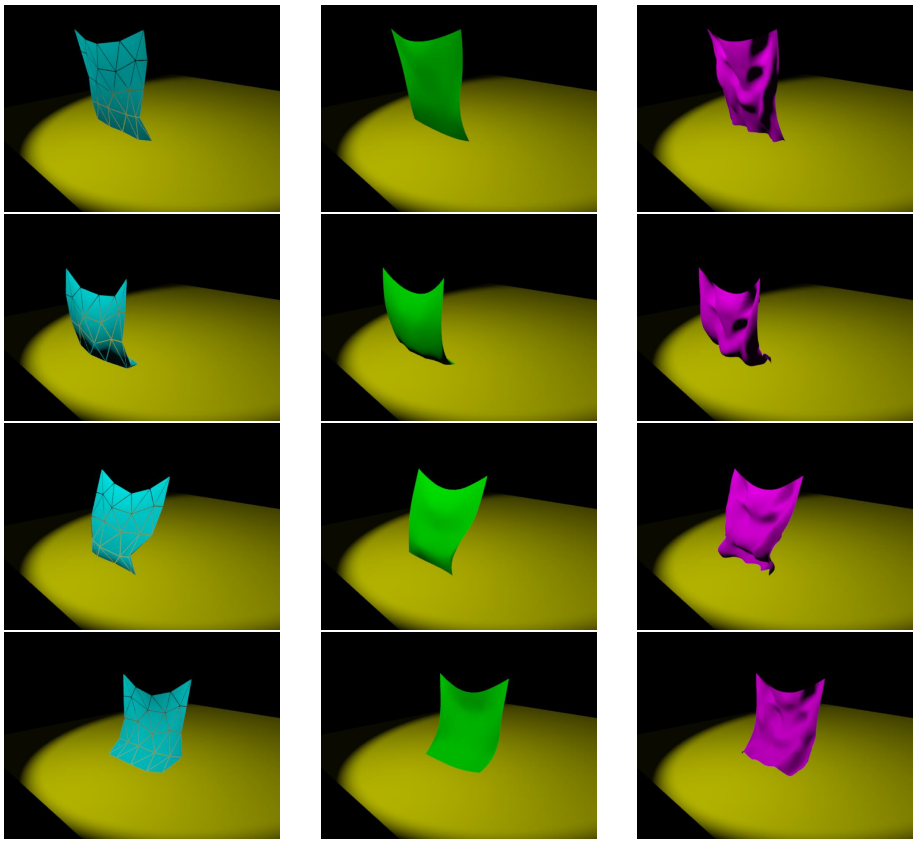

Figure 4.2: A set of 4 frames displaying the coarse (left column), Loop subdivision (centre column) and physics-aware subdivision (right column) meshes.

4.1 Future Work

As previously mentioned the introduction of an alternate representation of the membrane energy formulation is needed to avoid the current artifacts. An attempt was made at employing a formulation of the area preservation term as a minimization of the strain on each triangle element of the surface mesh. However, given the fact that our resulting surface is extended in the normal direction to the base surface mesh, the distance between extended points that are uniform on the Loop subdivision mesh may not be uniform on the physics-aware subdivision mesh. Thus, a formulation such as this will needlessly constrain certain

triangles on the physically subdivided mesh from stretching, even if stretching those triangles could result in a proper result. Instead we desire a metric for the membrane energy in 3 dimensional form that minimizes a difference between the surface area of each coarse mesh triangle and the sum of areas of its subdivision triangles. This would provide the closest analogy to the functional 1 dimensional solution. However, a description of membrane energy for an extensible surface such as this would require differentiating the area term in the context of our method which is extremely complex and undesirable. As a result we list this as future work and are still attempting to progress towards a concrete solution to this part of the problem.

Another area for future direction is that of performance improvements especially those that could lend themselves to the benefits of parallel architectures. The element-by-element approach used in the second step of the time integration technique is set up in such a way that easily lends itself to parallelism. Since each element solve is done completely independent of the others these could easily be done simultaneously given the proper hardware architecture. This would certainly serve to provide some sort of performance benefit though it has not been tested at this time.

Bibliography

[1] David Baraff. Linear-time dynamics using lagrange multipliers. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 137–146, New York, NY, USA, 1996. ACM Press.

[2] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annural conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, USA, 1998. ACM Press.

[3] David Baraff, Andrew Witkin, and Michael Kass. Untangling cloth. *ACM Trans. Graph.*, 22(3):862–870, 2003.

[4] Miklos Bergou, Max Wardetzky, David Harmon, Denis Zorin, and Eitan Grinspun. A quadratic bending model for inextensible surfaces. In *Eurographics Symposium on Geometry Processing*, 2006.

[5] Eddy Boxerman. Speeding up cloth simulation. Master's thesis, University of British Columbia, 2003.

[6] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 28–36, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[7] Robert Bridson. *Computational aspects of dynamic surfaces*. PhD thesis, Stanford University, 2003.

[8] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 594–603, New York, NY, USA, 2002. ACM Press.

[9] M. Ortiz C. Kane, J. E. Marsden and M. West. Variational integrators and the newmark algorithm for conservative and dissipative mechanical systems. *International Journal for Numerical Methods in Engineering*, 49(10):1295–1325, 2000.

[10] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. Interactive skeleton-driven dynamic deformations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 586–593, New York, NY, USA, 2002. ACM Press.

[11] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 604–611, New York, NY, USA, 2002. ACM Press.

[12] Fehmi Cirak and Michael Ortiz. Fully c1-conforming subdivision elements for finite deformation thin-shell analysis. *International Journal for Numerical Methods in Engineering*, 51(7):813–833, 2001.

[13] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer*

graphics and interactive techniques, pages 31–36, New York, NY, USA, 2001. ACM Press.

[14] Mathieu Desbrun, Peter Schröder, and Alan Barr. Interactive animation of structured deformable objects. In *Proceedings of the 1999 conference on Graphics interface '99*, pages 1–8, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[15] Naga K. Govindaraju, David Knott, Nitin Jain, Ilknur Kabul, Rasmus Tamstorf, Russell Gayle, Ming C. Lin, and Dinesh Manocha. Interactive collision detection between deformable models using chromatic decomposition. *ACM Trans. Graph.*, 24(3):991–999, 2005.

[16] Eitan Grinspun, Anil N. Hirani, Mathieu Desbrun, and Peter Schröder. Discrete shells. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 62–67, Aire-la-Ville, Switzerland, 2003. Eurographics Association.

[17] Eitan Grinspun, Petr Krysl, and Peter Schröder. Charms: a simple framework for adaptive simulation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 281–290, New York, NY, USA, 2002. ACM Press.

[18] Donald H. House and David E. Breen. *Cloth modeling and animation*. "A.K. Peters, Ltd.", Natick, MA, 2000.

[19] T. J. R. Hughes and W. K. Liu. Implicit explicit finite elements in transient analysis: I. stability theory; ii. implementation and numerical examples. *Journal of Applied Mechanics*, 45:371–378, 1978.

[20] Y. M. Kang and H. G. Cho. Bilayered approximate integration for rapid and plausible animation of virtual cloth with realistic wrinkles. In *Computer Animation, IEEE Computer Society*, pages 203–214, 2002.

[21] Y.-M. Kang, J.-H. Choi, H.-G. Cho, and C.-J. Park. An efficient animation of wrinkled cloth with approximate implicit integration. *The Visual Computer*, 17(3):147–157, 2001.

[22] L. Kharevych, Weiwei, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and Mathieu Desbrun. Geometric, variational integrators for computer animation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2006.

[23] Dimitri Metaxas and Demetri Terzopoulos. Dynamic deformation of solid primitives with constraints. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 309–312, New York, NY, USA, 1992. ACM Press.

[24] Hing N. Ng and Richard L. Grimsdale. Computer graphics techniques for modeling cloth. *IEEE Comput. Graph. Appl.*, 16(5):28–41, 1996.

[25] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface '95*, pages 147–154, 1995.

[26] Xavier Provot. Collision and self collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation '97*, pages 177–189, 1997.

[27] Ari Stern and Mathieu Desbrun. Discrete geometric mechanics for variational integrators (a primer for cs). ACM SIGGRAPH '06 Course Notes on Discrete Differential Geometry, 2006.

[28] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, 2004.

[29] Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. Robust quasistatic finite elements and flesh simulation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 181–190, New York, NY, USA, 2005. ACM Press.

[30] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, New York, NY, USA, 1987. ACM Press.

[31] Jeffrey A. Thingvold and Elaine Cohen. Physical modeling with b-spline surfaces for interactive design and animation. In *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics*, pages 129–137, New York, NY, USA, 1990. ACM Press.

[32] B. Thomaszweski, M. Wacker, and W. Straßer. A consistent bending model for cloth simulation with corotational subdivision finite elements. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2006.

[33] P. Volino and N. Magnenat-Thalmann. Fast geometric wrinkles on animated surfaces. In *Seventh International Conference in Central Europe on Computer Graphics and Visualization (Winter School on Computer Graphics)*, 1999.

[34] Pascal Volino, Frederic Cordier, and Nadja Magnenat-Thalmann. From early virtual garment simulation to interactive fashion design. *Computer-Aided Design Journal*, 37:593–608, 2005.

[35] Pascal Volino and Nadia Magnenat Thalmann. Collision and self-collision detection: Efficient and robust solutions for highly deformable surfaces. In Dimitri Terzopoulos and Daniel Thalmann, editors, *Computer Animation and Simulation '95*, pages 55–65. Springer-Verlag, 1995.

[36] Matthew West. *Variational Integrators*. PhD thesis, California Institute of Technology, 2004.

Appendix A

Derivation of Strain Limiting Equations

A.1 Position and Velocity Update Formulas

The following is the derivation for the position and velocity updates used in the element-by-element strain limiting portion of our time integration technique.

Given a vector, $\hat{d}$ of unit length representing the direction of strain across an element and element vertices $\hat{r}_i = \hat{x}_i - \hat{x}_{cm}$ where $\hat{x}_i$ is the position of the ith point in world space and $\hat{x}_{cm}$ is the centre of mass, we can express $\hat{d}$ as a linear combination of the $\hat{r}_i$'s.

$$\hat{d} = \sum_i \beta_i \hat{r}_i \tag{A.1}$$

With $\sum_i \beta_i = 0$ and $\sum_i m_i \hat{r}_i = 0$. Now we wish to choose an α such that:

$$\sum_i m_i \alpha_i = 0 \tag{A.2}$$

and

$$\sum_i m_i \alpha_i \cdot \hat{d} \times \hat{r}_i = 0 \tag{A.3}$$

Substituting A.1 into A.3 gives:

$$\sum_i m_i \alpha_i (\sum_j \beta_j \hat{r}_j) \times \hat{r}_i = 0 \tag{A.4}$$

$$\sum_{ij} m_i \alpha_i \beta_j (\hat{r}_i \times \hat{r}_j) = 0 \tag{A.5}$$

Using the identity of the cross product, the following substitution can be made:

$$\hat{r}_i \times \hat{r}_j = s_{ij} \hat{n} \tag{A.6}$$

where

$$s_{ij} = \sin\theta |\hat{r}_i||\hat{r}_j| \tag{A.7}$$

to give

$$\sum_{ij} m_i \alpha_i \beta_j (s_{ij} \hat{n}) = 0 \tag{A.8}$$

Where $\hat{n}$ is the normal to the element surface. Note that the s_{ij}'s are entries of a matrix S where diagonal elements, s_{ii} are equal to 0 and $s_{ij} = -s_{ji}$ revealing a skew symmetric matrix. This means that if we let $a_i = m_i \alpha_i$ then A.8 simplifies to

$$a^T S \beta = 0 \tag{A.9}$$

Because S is skew symmetric we can show

$$(a^T S a)^T = (Sa)^T a = a^T S^T a = -a^T S a \tag{A.10}$$

So we make the choice of α to be

$$\alpha_i = \frac{\beta_i}{m_i} \tag{A.11}$$

which results in

$$\sum_i m_i \alpha_i = \sum_i \beta_i = 0 \tag{A.12}$$

as indicated by our original assumption. So now we have determined that the position and velocity updates, $\Delta\hat{x}_i$ and $\Delta\hat{v}_i$, must be proportional to $\frac{\beta_i}{m_i}$ in order to ensure preservation of angular and linear momenta. This means that our update formulas take on the following format:

$$\hat{x}_i^{new} = \hat{x}_i + \alpha \frac{\beta_i}{m_i} \hat{d} \tag{A.13}$$

$$\hat{v}_i^{new} = \hat{v}_i + \gamma \frac{\beta_i}{m_i} \hat{d} \tag{A.14}$$

A.2 Derivation of α and γ

A.2.1 Deriving α

The vector, $\hat{d}$, representing the direction of strain across the element has an equivalent representation, $\hat{q}$, in parameter space. The ratio of their lengths, $\frac{|\hat{d}|}{|\hat{q}|}$ is the scalar value of the strain in direction $\hat{d}$. We can look at setting the desired strain, ϵ_{des}, as a scaling of $\hat{d}$ such that

$$|\hat{d}^{new}| = \epsilon_{des} |\hat{q}| \tag{A.15}$$

Using the definition from A.1 and plugging into A.15 results in

$$\sum_i \beta_i \hat{r}_i^{new} \cdot \hat{d} = \epsilon_{des} |\hat{q}| \tag{A.16}$$

From the definition of the update formula (A.13) this becomes:

$$\sum_i \beta_i (\hat{r}_i + \alpha \frac{\beta_i}{m_i} \hat{d}) \cdot \hat{d} = \epsilon_{des} |\hat{q}| \tag{A.17}$$

Then since $\hat{d} \cdot \hat{d} = 1$ and given A.1 this simplifies to:

$$1 + \alpha \sum_i \frac{\beta_i^2}{m_i} = \epsilon_{des} |\hat{q}| \tag{A.18}$$

Finally, solving for α gives:

$$\alpha = \frac{\epsilon_{des} |\hat{q}| - 1}{\sum_i \frac{\beta_i^2}{m_i}} \tag{A.19}$$

A.2.2 Deriving γ

The derivation of γ proceeds very similarly to that of α. We begin with

$$\sum_i \beta_i \hat{v}_i^{new} \cdot \hat{d} = 0 \tag{A.20}$$

indicating we want to stop any velocities that would continue to stretch the element in the $\hat{d}$ direction. As with the positions in the derivation of α, we now replace the new $\hat{v}_i$'s with the update formula A.14

$$\sum_i \beta_i (\hat{v}_i + \gamma \frac{\beta_i}{m_i} \hat{d}) \cdot \hat{d} = 0 \tag{A.21}$$

Using $\hat{d} \cdot \hat{d} = 1$ this simplifies to:

$$\sum_i \beta_i \hat{v}_i \cdot \hat{d} + \gamma \sum_i \frac{\beta_i^2}{m_i} = 0 \tag{A.22}$$

Solving for γ then gives us the definition:

$$\gamma = -\frac{\sum_i \beta_i \hat{v}_i \cdot \hat{d}}{\sum_i \frac{\beta_i^2}{m_i}} \tag{A.23}$$

Appendix B

Connection to Cubic Splines

Given a 1 dimensional parametric curve parameterized by the scalar s we have a curve defined by the function $f(s)$. We can obtain the the curvature at a point by taking the second derivative of $f(s)$ such that:

$$K(s) = f''(s) \tag{B.1}$$

A spline curve is defined as a curve which passes through the constrained points or control points in the smoothest path possible i.e. by taking a path which minimizes curvature along the curve. Mathematically this translates into the minimization of the integral of curvature across the entire curve and in terms of $f(s)$, a minimization of $f(s)$ across the integral of $f''(s)$:

$$\begin{aligned} &\min \int K(s)^2 \\ &\min f \int (f''(s))^2 \end{aligned} \tag{B.2}$$

If we define this quantity as E we can approach the minimization of E from a variational perspective. Assume we have an arbitrary function g. We take a small step of distance ϵ in the direction of g.

$$\begin{aligned} E(\epsilon) &= \int[(f+\epsilon g)'']^2 \\ &= \int(f''+\epsilon g'')^2 \\ &= \int f''^2 + 2\epsilon f''g'' + \epsilon^2 g''^2 \end{aligned} \tag{B.3}$$

If f is a minimizer of E then $E'(0) = 0$ at the minimium point. Thus we differentiate B.3 and equate to 0:

$$E'(\epsilon) = \int 2f''g'' + 2\epsilon g''^2 \tag{B.4}$$

$$E'(0) = 2\int f''g'' \tag{B.5}$$

Now we use integration by parts to obtain:

$$\begin{aligned} E'(0) &= -2\int f'''g' \\ &= 2\int f^{(iv)}g \end{aligned} \tag{B.6}$$

Since g is an aribtrary function and $E'(0) = 0$, we know that $f^{(iv)}$ must equal zero for this to hold. Thus since the 4th derivative of f is zero then by definition f is a cubic function.