Novel Solver for Dynamic Surfaces

Sumantro Ray **Hong Qin**

Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400
sray | qin@cs.sunysb.edu

Abstract

Physics-based modeling integrates dynamics and geometry. The standard methods to solve the Lagrangian equations use a direct approach in the spatial domain. Though extremely powerful, it requires time consuming discrete-time integration. In this paper, we propose to use an indirect approach using the Transformation Theory. In particular, we use z-transform from the digital signal processing theory, and formulate a general, novel, unified solver that is applicable for various models and behavior. The convergence and accuracy of the solver are guaranteed if the temporal sampling period is less than the critical sampling period, which is a function of the physical properties of the model. Our solver can seamlessly handle curves, surfaces and solids, and supports a wide range of dynamic behavior. The solver does not depend on the topology of the model, and hence supports non-manifold and arbitrary topology. Our numerical techniques are simple, easy to use, stable, and efficient. We develop an algorithm and a prototype software simulating various models and behavior. Our solver preserves physical properties such as energy, linear momentum, and angular momentum. This approach will serve as a foundation for many applications in many fields.

Key words: Physics-based deformable modeling, Numerical techniques, Heterogeneous models, Conceptual design techniques, z-Transforms

1 Introduction

Geometric modeling concerns with the computation and representation of various shapes of models. Physics-based modeling allows geometric models to be governed by differential equations. This approach offers unsurpassed advantages- it is natural to control, intuitive to manipulate, and the end user does not need mathematical sophistication. However, it requires numerical simulation, which is time consuming, and is occasionally unstable. Also, the topology (geometry) of the model is not clearly separated from its physical attributes.

Our approach uses a mass-spring model to develop a general and novel formulation. The physical properties are associated with each vertex of the mesh, and the vertices interact through (internal and external) forces and torques. This duality of the solver allows it to be independent of the topology of the model. This approach can be extended to a finite element based model, where the internal stretching and bending forces are replaced with normal and shear stresses. Our unified solver supports a hybrid model that seamlessly integrates the particle, elastic, and near-rigid models, and supports curves, surfaces and solids. The core solver is independent of the topology of the model, and can thus be used to model non-manifold surfaces, and models with arbitrary topology. We develop a hybrid technique to generate continuous surfaces from the mesh topology. The model depends on Newton's second law of motion to ensure that the total energy and (linear and angular) momentum of the system is conserved throughout the simulation. This paper presents a unique way of computing the physical state of the system by transforming the state equations to a transform domain *z* from the time domain *t* using the z-transform. The stability of the solver thus depends only on the *Nyquist Rate of Sampling*, which can be optimally computed from the physical properties of the model. We perform an inverse-z transform to get back to time domain after solving the equations in z-domain. The state of the system at time $t + \delta t$ can be expressed as a linear function of the state at time *t*. The coefficients of this function are called transformation matrices.

Semi-rigid motion is obtained by coupling the elastic solver with the rigid body equations. The moment of inertia of the surface is assumed to be due to a thin shell with a vanishing thickness δr. The moment of inertia varies dynamically as the model is manipulated. This technique allows the solver to simulate *near*-rigid surfaces. This technique can be easily extended to solids.

The system allows the user to interactively edit physical properties such as mass, damping, stiffness and length. The system fast-updates the transformation matrices and allows the user to interactively play with the model.

The rest of the paper is organized as follows. Section **2** explores the background. Section **3** presents the contribution. Section **4** deals with the formulation of the solver. Section **5** explains the internal/external forces and torques. Section **6** outlines the system and the prototype software. Section **7** presents the results. Section **8** concludes the paper.

2 Background

Various work has previously been done to generate dynamic surfaces using physics-based modeling. Terzopoulos[17] demonstrates simple interactive sculpting using viscoelastic and plastic methods. Celniker[4] uses finite-element optimization of energy functionals. Bloor et al[3] use similar optimizations through numerical methods. Dachille et al[5] use a finite difference method to solve the Lagrangian equation. Halstead et al[8] implement smooth interpolation with Catmull-Clark surfaces using a thin-plate energy functional. Raviv and Elber[15] perform three dimensional free-form sculpting using scalar trivariate functions. Qin and Terzopoulos[14] develop a framework for D-NURBS for physics-based design.

This paper attempts to merge Lagrangian mechanics in a transform domain framework. This allows the physics and the geometry to get decoupled, and the invariance of energy and momentum is the only link between the two. This has been a difficulty in previous works, where a strong coupling between physics and geometry meant that the solver is highly dependent upon the geometry of the model. Our approach is isomorphic to the eigenvalue analysis. We refer to [16], [7], [13] and [12] for the eigenvalue decomposition algorithms and principles. Kreyszig[9] discusses transformation theory and complex analysis, that has been used for the formulation of the solver. Interested readers might refer to Defatta et al [6] for the digital signal processing principles used in this paper.

A number of people have worked on rigid-body models. We refer to [11] for the details on moment of inertia calculations using Green's functions. David Baraff[1] talks about dynamic simulation of rigid bodies through analytical simulation. Baraff[2] demonstrates fast contact force computation for non-penetrating rigid bodies. We refer to the classic book Vector and Tensor Analysis by Harry Lass[10] for the derivations of the volume integration functions and kinematic motion equations.

3 Contribution

This paper attempts to separate the geometric and the physical properties of the model. The geometric properties govern the topology of the model. The physical properties are used to precompute a set of *transformation matrices* that compute the state of the model at time $t + \delta t$ given the state at time t. The state of the model is uniquely defined by the position and the velocity state vectors. The precomputed transformation matrices allow us to have an accurate, faster solver.

The following describe the different contributions of this paper.

Solver: We present a novel, fast, unified solver that can operate on a multitude of models. In particular, the solver can simulate curves, surfaces and solids of arbitrary topology and varying rigidity. The solver depends upon the critical sampling rate of the model which is precomputed from its physical properties. The solver is provably accurate, as the global properties such as the total energy and (linear and angular) momentum do not diverge for sampling periods less than the critical sampling period. We compute the critical sampling time from the physical properties of the model.

Forces: The solver handles both radial (stretching) and angular (bending) forces. The radial forces give rise to internal stress, and the angular forces result in internal torque. The internal stress depends upon the linear distance of a vertex from its neighboring vertices. The internal torque depends upon the angular distance of a vertex from its neighboring vertices.

Semi-rigid Behavior: The solver handles both purely elastic and purely rigid bodies and all shades in between. This approach enables the solver to be fast in case the internal stiffnesses are high, as it can increase the average rigidity of the model and reduce the internal stiffnesses. An infinite average stiffness implies a purely rigid model. This technique allows the solver to reduce the sampling rate when the average stiffness is high. This assumes that the external forces on the model are relatively small (compared to the internal forces), which is a fair assumption in most cases.

Dynamic Editing: The user can dynamically edit the physical properties of the model. The user can paint mass, (radial and angular) stiffness, damping, rigidity and rest-length and rest-angle to the model. The transformation matrices are dynamically updated to generate the desired effect. The energy of the model is *not* conserved in some cases, as the user interaction changes its total energy. The matrices are updated in O(k^2) time, where k is the average valence of a vertex. For small k (as it normally is), this is not very expensive.

Smoothing: The final smooth model is generated using a modified Doo-Sabin technique. The subdivision weights depend upon the physical properties of the surface. This ensures the invariance of energy, (linear and angular) momentum and total mass after subdivision. The normal to the surface is computed through a best-plane fit of the neighboring vertices in the limit case.

4 Solver

We begin with the Lagrangian equation, that defines the motion of a point, or a set of points in three dimensions. Given a model which has been arbitrarily sampled, the motion of the set of sampled points can be simulated using Lagrangian Dynamics (within a margin of error) to determine the behavior of the model under external and internal forces. The input consists of a set of sampled vertices and faces with associated physical properties.

4.1 Formulation

The Lagrangian equation of motion is expressed as:

$$\mu \mathbf{r}'' + \rho \mathbf{r}' + \sum_j \kappa_j [(\mathbf{r} - \mathbf{r}_j) + \lambda_j \mathbf{U}_j] = \mathbf{f}_{tot}$$

$$\mathbf{v} = \mathbf{r}' = d\mathbf{r}/dt$$

$$\mathbf{U}_j = (\mathbf{r} - \mathbf{r}_j)/|(\mathbf{r} - \mathbf{r}_j)| \tag{1}$$

for any vertex **r**, where $\mathbf{r}_j$ is a neighboring vertex. The mass, damping and the directional stiffness for the vertex are given by μ, ρ, and κ_j respectively. λ_j is the rest-length between **r** and $\mathbf{r}_j$. The right side is the sum of the external and torsional forces at vertex **r**. **U** is the directional unit vector. The first two terms represent the kinetic and the damping components of the total force. The third component represents the internal (radial spring) force at vertex **r**.

For a set of *N* vertices, the Lagrangian equation may be written in a matrix format, such that **R** represents the N$\times$3 dimensional positional state vector, and **V** represents the N$\times$3 dimensional velocity state vector. Therefore, we get

$$M\mathbf{R}'' + D\mathbf{R}' + K\mathbf{R} = \mathbf{F}_{tot}$$

$$\mathbf{V} = \mathbf{R}' = D\mathbf{R}/Dt \tag{2}$$

where the *D/Dt* operator is a matrix operator on the state vector **R**. M and D are N$\times$N diagonal mass and damping matrices. K is a N$\times$N symmetric sparse matrix such that -K_{ij} is the stiffness between the i^{th} and the j^{th} vertices. F_{tot} is the N$\times$3 force vector state. K_{ii} equals the sum of all neighboring stiffnesses at vertex $\mathbf{r}_i$. Therefore, elements of any row (or column) in K add up to zero. This is a direct consequence of Newton's third law of motion.

Let G = D$M^{-1}/2$ and Ω = $\sqrt{KM^{-1}}$, where G is the coefficient-of-damping matrix, and Ω is the angular frequency matrix. The eigen-value decomposition of $\sqrt{\Omega^2 - G^2}$ gives us the different modes of oscillation. The z-transform of equation 2 transforms it from the temporal domain to the digital z-domain. The various modes of oscillation represent the various poles/roots in the z-domain. Please refer to appendix A for further details on z-transform.
If the sampling time-period is *T*, then we can compute the N*th* state from the (N-1)*th* state, where ($\mathbf{R}^N$, $\mathbf{V}^N$) is the state of the system at sample number N. This result is obtained after an inverse z-transform on the z-domain formulation. Note that the trigonometric operations on the Ω matrix are approximated using the Taylor series expansions of the expressions.

$$\begin{pmatrix} \mathbf{R}_p^N \\ \mathbf{V}_p^N \end{pmatrix} = \begin{pmatrix} I & H' & H \\ \mathbf{0} & X' & X \end{pmatrix} \begin{pmatrix} \mathbf{R}_p^{N-1} \\ \mathbf{V}_p^{N-1} \\ \mathbf{F}_{tot}^{N-1} \end{pmatrix}$$

where $X = H' - GH, X' = H'' - GH'$ and

$$\begin{pmatrix} H \\ H' \\ H'' \end{pmatrix} = \begin{pmatrix} \Omega^{-2}(1 - cos\Omega T) \\ \Omega^{-1} sin\Omega T \\ cos\Omega T \end{pmatrix}$$

$$\simeq \begin{pmatrix} \frac{T^2}{2} & \frac{-T^4}{24} & \frac{T^6}{720} \\ T & \frac{-T^3}{6} & \frac{T^5}{120} \\ 1 & \frac{-T^2}{2} & \frac{T^4}{24} \end{pmatrix} \begin{pmatrix} 1 \\ \Omega^2 \\ \Omega^4 \end{pmatrix}$$

$$\mathbf{R}^N = M^{-1}\mathbf{R}_p^N$$
$$\mathbf{V}^N = M^{-1}\mathbf{V}_p^N \tag{3}$$

4.2 Sampling Rate

Equation 3 depends upon the fourth power of the angular frequency matrix. This implies that the dynamics of the i^{th} vertex at time *t+T* depends upon the (position and velocity) states of all the vertices that are a distance of two units (along the mesh graph) from it. Therefore, the convergence of the dynamic behavior is dependent upon the sampling rate. A lower sampling rate might result in a divergent set of equations. The goal is to find the critical sampling rate, such that the state vectors do not diverge as *t* goes to infinity.

If the eigen decomposition of Ω^2 is given by Ω^2 = SΛS^{-1}, where S is the column eigen-vector matrix and Λ diagonal are the eigen-values, then we must satisfy the following:

$$\lim_{k\to\infty}(H'' - GH')^k = \mathbf{0}$$

If γ is the minimum diagonal element of G, we have:

$$\begin{aligned} 0 < p + q\lambda_i + r\lambda_i^2 < 1 \\ p = (1 - \gamma T) \\ q = (1 - \frac{\gamma T}{3})\frac{T^2}{2} \\ r = (1 - \frac{\gamma T}{5})\frac{T^4}{24} \end{aligned} \quad (4)$$

for all eigen-values λ_i. Since $\lambda_i \geq 0$ (Ω^2 is positive-definite), and p, q, r > 0, it is sufficient if the above equation holds true for the largest eigen-value. Let the largest eigen-value be λ_0. From Gerschgorin Circle Theorem (explained in appendix B), λ_0 must be less than twice of the maximum diagonal entry of Ω^2. Let this upper bound for λ_0 be λ_+. If $\gamma \ll$ 1/T, we solve for the critical sampling time T_{max} from Equation 4:

$$\begin{aligned} \lambda_+ = 2max(K_{ii}\mu_i^{-1}) \\ T_{max} = \sqrt{\frac{12}{\lambda_+}} = \sqrt{\frac{6}{max(K_{ii}\mu_i^{-1})}} \end{aligned} \quad (5)$$

Including γ in the formulation increases T, since γ is a decaying factor which allows convergence at a higher sampling time. Therefore, T_{max} is an optimal value for sampling time, irrespective of damping. The factor '2' in the expression for λ_+ also appears in the *Nyquist Theorem*, where the sampling rate must be at least twice the maximum frequency of the system. T_{max} is inversely proportional to the minimum sampling rate (Equation 5).

4.3 Semi-rigid Motion

We define the rigidity coefficient of a vertex $\mathbf{r}_i$ as χ_i, such that $\chi_i \in [0, 1]$. A value of 0 indicates a pure local elasticity, and a value of 1 indicates a pure local rigidity.

The sampling rate of the solver is directly proportional to the square root of the maximum stiffness of the model in a purely elastic model. Therefore, the larger the stiffness, the larger is the sampling rate, and lower is the response time. However, notice that as we increase the local stiffness at any vertex $\mathbf{r}_i$, the local rigidity of the model increases, provided that the external forces at that vertex are not large. Also, notice that the internal forces are directly proportional to the local stiffness. Therefore,

- Let κ_0 be the maximum permissible stiffness. The critical sampling time depends on κ_0.
- Let σ_i = *min*(κ_0/κ_i, 1) for the vertex $\mathbf{r}_i$.
- Set rigidity coefficient χ_i = 1 - σ_i.

If the total force (external and internal) at $\mathbf{r}_i$ is $\mathbf{F}_i$, we partition this force into elastic and rigid-body forces. Let

- $\mathbf{F}_i^\kappa = \sigma_i\mathbf{F}_i$ be the elastic component.
- $\mathbf{F}_i^\chi = \chi_i\mathbf{F}_i$ be the rigid-body component.

The elastic component of the force is used in the solver equation. The rigid-body component is used for the rigid-body motion of the model using the rigid-body motion equations in [10].

5 Internal and External Forces

The total force on the model is a vector sum of the internal and the external forces. The internal forces can be radial stresses (due to the radial springs) or torsional stresses (due to the angular springs). The external forces can be user-interactions, or body forces such as gravity and virtual forces (to maintain constraints). The internal stresses are depicted in Figure 1.

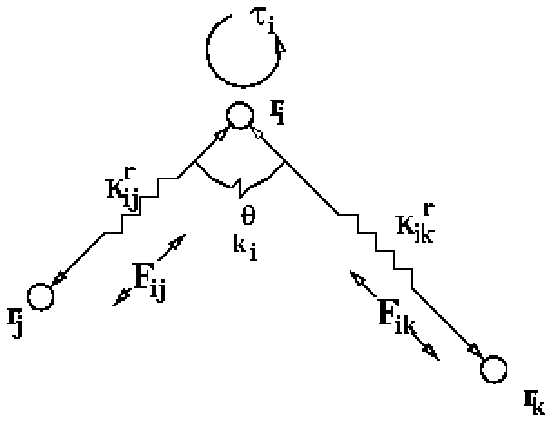

Figure 1: *The radial spring between two vertices results in a restoring (radial) force. The angular spring at the angle between two edges results in a restoring torque.* r *and θ stand for radial and angular respectively.*

5.1 Linear Stress

The radial stress at a vertex is due to the variation in the linear displacement from the rest-length along its connections to its neighbors. The radial stress on a spring of stiffness κ_r and length λ connecting $\mathbf{r}$ and its *j*th neighbor at sample N is defined as:

$$\mathbf{F}_r^N = \kappa_r((\mathbf{r}^{N-1} - \mathbf{r}_j^{N-1}) - \lambda_j\mathbf{U}_j^{N-1})$$

5.2 Torsional Stress

The torsional stress at vertex $\mathbf{r}_i$ is due to the variation in the angular displacement from the rest-angle along the angles formed with two neighboring vertices $\mathbf{r}_j$ and $\mathbf{r}_k$. The torque at $\mathbf{r}_i$ is proportional to the variation in the angle $\delta\theta{=}\theta^N - \theta_0$. If κ_θ is the angular stiffness, and θ_0 is the rest angle, we have the torque and the torsional stresses as:

$$\begin{aligned}
\tau_\theta^N &= \kappa_\theta(\theta^N - \theta_0) d_{ij} d_{ik} \mathbf{n} \\
\mathbf{F}_{ij}^N &= \kappa_\theta(\theta^N - \theta_0)\mathbf{n} * \mathbf{r}_{ij} \\
\mathbf{F}_{ik}^N &= \kappa_\theta(\theta^N - \theta_0)\mathbf{n} * \mathbf{r}_{ik} \\
\mathbf{F}_{ii}^N &= -(\mathbf{F}_{ij}^N + \mathbf{F}_{ik}^N) \\
\mathbf{n} &= \mathbf{r}_{ij} * \mathbf{r}_{ik}/(d_{ij} d_{ik})
\end{aligned}$$

where $\mathbf{n}$ is the normal vector, $\mathbf{r}_{ij}$ equals $\mathbf{r}_i - \mathbf{r}_j$, and d_{ij} is the length of this vector. Same holds for $\mathbf{r}_{ik}$ and d_{ik}. This ensures that the resulting forces on the vertices provide a restoring torque that balances the external torque. The total force on the system must be zero when a pure torque is applied. This requires the application of a restoring force on vertex $\mathbf{r}_i$. The operator '*' represents vector cross product in the above equations.

5.3 External Force/Torque

The system allows the user to apply forces and torques to the model. The external forces are added to the right side of equation 1. The external torques are converted to virtual forces on the neighboring vertices.

In case the local rigidity is not equal to zero, the total force and torque as obtained above are partitioned into the elastic and the rigid-body components. Refer to subsection 4.3 for details.

6 System Internals

We describe the flow diagram of the system in Figure 2. The system, on startup, initializes a number of structures. The transformation matrices H, H' and H'' and frequency matrices Ω^2 and Ω^4 (section 4.1) are precomputed. The time taken for the precomputation process is $O(Nk^2)$ where *N* is the number of vertices, and *k* is the average valence. The system operates two threads (Display and Solver) which are also initialized, and thread-specific data structures are set up.

The two threads are tightly coupled, each receiving some input from and feeding some output to the other on a per-cycle basis.

- Display: The Display is responsible for running the display and handling user inputs. The user interactions are updated into the model structures so that the Solver can incorporate them into the next time cycle. The Display expects updated state vector data from the Solver.

- Solver: The Solver evaluates the state vectors in a tight loop. It updates the display structures on completion of a cycle. The interactive forces by the user are evaluated on a per-cycle basis.

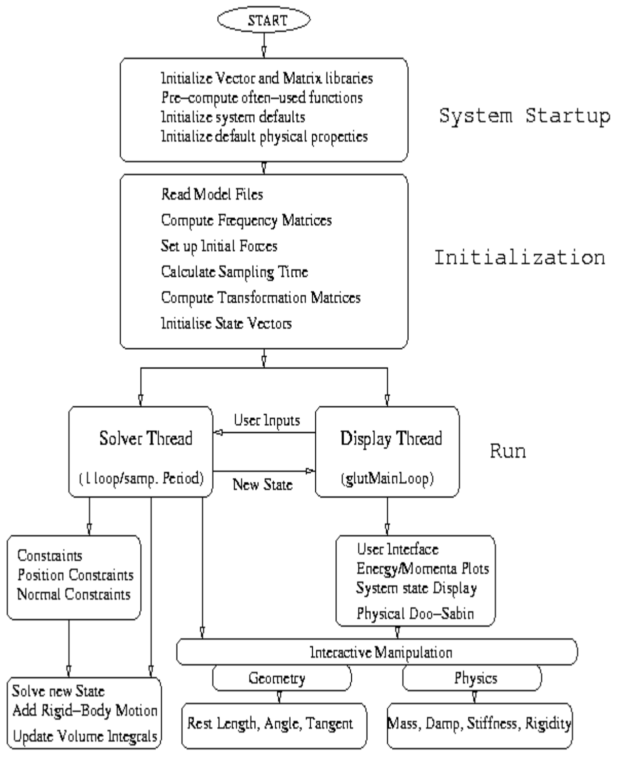

Figure 2: *Flow Diagram*

We explain the timing details for the Solver and the Display threads and their loads in the Results section. The system supports the following tools:

- Dynamic Editing: The user can interactively change the physical parameters of the model. The user can manipulate mass, damping, stiffness (radial and angular), rest-length, rest-angle and rigidity of the model. The system dynamically updates the pre-computed transformation matrices in constant time. This allows the system to maintain the integrity of the solver. However, editing physical properties may increase the intrinsic stress of the model, which is similar to the user adding energy to the environment. The editing process can be local as well as global.

- Constraints: The user can constrain the position and the normal vector at any vertex of the model. The solver generates virtual forces such that the appropriate result is obtained in the next time cycle.

7 Results

Table 1 presents a set of models used. The first column gives the number of vertices, edges and faces for the corresponding model. The second column gives the average values for mass and (radial) stiffness. The third, fourth and fifth columns present the timing data for the precomputation, display and solver threads in seconds. The sixth column gives the sampling time in seconds. The load on the system is inversely proportional to the sampling time. However, a higher sampling time also means a sluggish response time, and the two must be balanced for better interactivity. The results were obtained on a Pentium-II 450MHz PC. The solver time (for one iteration) exceeds (for the given mass and stiffness values) the sampling time for number of vertices larger than approximately 1200.

V,E,F	μ, κ	Init	Disp	Solv	STime
20, 30, 12	0.50,5.00	0.097	0.016	0.013	0.448
100, 200, 100	0.10,5.00	0.281	0.018	0.036	0.353
324, 612, 289	0.31,5.16	0.842	0.046	0.085	0.323
1600,3200,1600	0.01,9.50	9.723	0.134	0.393	0.291

Table 1: *Timing information for the models used for the paper: 1. Dodecahedron, 2. Torus-I, 3. Open surface, 4. Torus-II. γ is 0.00 for all models. All timings are in seconds. Torus-II is the twice Doo-Sabin subdivided Torus-I.*

Figure 3 presents the partial energy against T/T_{max} for model 1 (Table 1). The partial energy plot diverges for T/T_{max} greater than 1.080. This is slightly greater than the ideal value of 1.00, since the largest eigen-value for Ω^2 matrix lies slightly inside the largest Gerschgorin Circle. (appendix B). The partial energy is the average energy of the model (energy per vertex).

8 Conclusion

This paper presents a novel, unified solver that can simulate a multitude of models such that the global energy and momenta are conserved. We present a solver that does not require any input from the user except the geometric and physical properties that define a model. The user can dynamically modify the physical properties of the model to achieve the desired behavior. The system handles rigid, elastic and semi-rigid models. Future work will include integration of the solver architecture with subdivision and multi-resolution models, and transformation of the mass-spring based model into a finite-element based model.

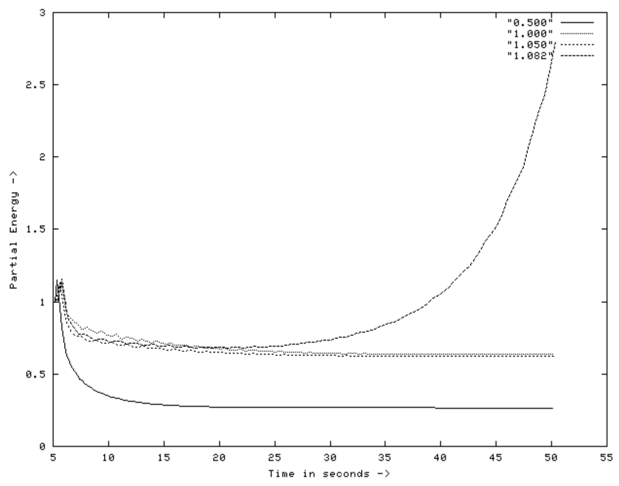

Figure 3: *Partial Energy - Time for T/T_{max} = 0.500, 1.000, 1.050 and 1.082. The plot diverges for values $\gtrsim$ 1.080.*

This will allow the system to use industry-standard techniques to achieve a robust, unified, and versatile solver that can be applied to a variety of models to achieve realistic animation and modeling.

9 Acknowledgements

This research was supported in part by NSF CAREER award CCR-9896123, the NSF grant DMI-9896170 and the NSF ITR grant IIS-0082035.

10 References

[1] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *SIGGRAPH '89, ACM Computer Graphics, July 1989*, 23(3):223–231, 1989.

[2] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 23–34. ACM SIGGRAPH, July 1994.

[3] M I G Bloor and M J Wilson. Using partial differential equations to generate free form surfaces. *Computer Aided Design, May 1990*, 22(4):202–212, 1990.

[4] George Celniker and Dave Gossard. Deformable curve and surface finite-elements for free-form shape design. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 257–266, July 1991.

[5] Frank Dachille IX, Hong Qin, Arie Kaufman, and Jihad El-Sana. Haptic sculpting of dynamic surfaces. In *Proceedings of the Conference on the 1999 Symposium on interactive 3D Graphics*, pages 103–110, April 26–28 1999.

[6] David DeFatta, Joseph Lucas, and William Hodgkiss. *Digitial Signal Processing - A system design approach*. John Wiley & Sons, first edition, 1988.

[7] Stephen Friedberg, Arnold Insel, and Lawrence Spence. *Linear Algebra*. Prentice Hall, third edition, 1997.

[8] Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. *Computer Graphics*, 27(Annual Conference Series):35–44, 1993.

[9] Erwin Kreyszig. *Advanced Engineering Mathematics*. John Wiley & Sons, sixth edition, 1988.

[10] Harry Lass. *Vector and Tensor Analysis*. McGraw-Hill Book Company, 1950.

[11] Brian Mirtich. *Impulse-Based Dynamic Simulation of Rigid Body Systems*. PhD thesis, 1996.

[12] James M. Ortega. *Numerical Analysis, a second course*. Siam, first edition, 1972.

[13] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.

[14] Hong Qin and Demetri Terzopoulos. D-NURBS: A Physics-Based Framework for Geometric Design. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):85–96, March 1996.

[15] Alon Raviv and Gershon Elber. Three dimensional freeform sculpting via zero sets of scalar trivariate functions. In *Proceedings of the Fifth Symposium on Solid Modeling and Applications (SSMA-99)*, pages 246–257. ACM Press, June 9–11 1999.

[16] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Wellesley, 1986.

[17] Demetri Terzopoulos and Kurt Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, December 1988.

Appendices

A Z-Transforms

The z-transform of a (causal) digital signal $\mathbf{x} = (x_0, x_1, x_2, ..., x_n, ...)$ is defined as

$$X(z) = x_0 + x_1 z^{-1} + x_2 z^{-2} + ... + x_n z^{-n} + ...$$

The partial position (and velocity) state vectors R_p and V_p are multi-dimensional temporal signals. Therefore, we can apply z-transforms to these signals. The z-transforms of the derivatives of R_p and V_p are some constant matrix multiplied by the z-transform of R_p and V_p respectively.

Applying z-transform to equation 2, the right side of the equation equals the following, where F_k^N is the force at vertex *k* at time *N*.

$$\begin{pmatrix} \mathbf{F}_0^0 & \mathbf{F}_0^1 & \mathbf{F}_0^2 & \dots & \mathbf{F}_0^N & \dots \\ \mathbf{F}_1^0 & \mathbf{F}_1^1 & \mathbf{F}_1^2 & \dots & \mathbf{F}_1^N & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\ \mathbf{F}_k^0 & \mathbf{F}_k^1 & \mathbf{F}_k^2 & \dots & \mathbf{F}_k^N & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} 1 \\ z^{-1} \\ \vdots \\ z^{-N} \\ \vdots \end{pmatrix}$$

Similarly, the left hand side can be expanded, and written in a form similar to above. Equating both sides, we get a matrix equation. Solving the equation, we get a closed form solution for $R_p^N(z)$ and $V_p^N(z)$. The final equation 2 is obtained by performing an inverse z-transform. Interested readers may refer to [6] for details.

B Gerschgorin Circle Theorem

According to the Gerschgorin Circle Theorem, for any $A = (a_{ij}) \in L(C^n)$ the eigen-values of A must lie inside the disks defined by Equation 6.

$$R_i = z : |a_{ii} - z| \leq \sum_{j \neq i} |a_{ij}|, i = 1, ..., n \qquad (6)$$

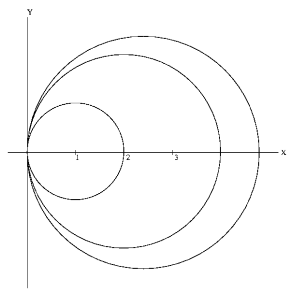

Figure 4: *Gerschgorin disks: The three disks represent the disks corresponding to three rows of Ω matrix (n=3). The eigen-values are bounded by the three disks.*

In our case, as shown in Figure 4, the circle *must* pass through the center. Since the eigen-values must lie inside the disks, the maximum value for the eigen-value λ_i is twice of the radius of the R_i. However, the radius is equal to the i*th* diagonal element of Ω^2, since $sum_{j \neq i} \Omega_{ij}^T = -\Omega_{ii}$. Therefore, we have $\lambda_i \leq 2\Omega_{ii}$.

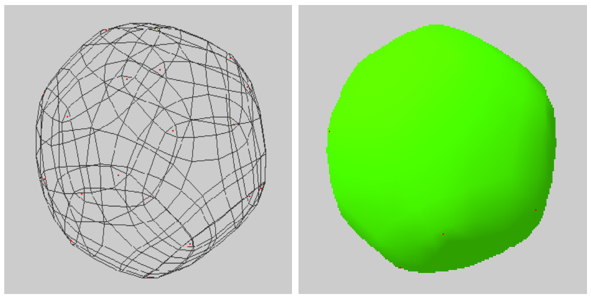

Figure 5: *The dodecahedron: The original mesh, and the interpolating Doo-Sabin surface. The original vertices are interpolated by the surface. The control vertices are in red. The subdivision weights are functions of the physical properties of the surface.*

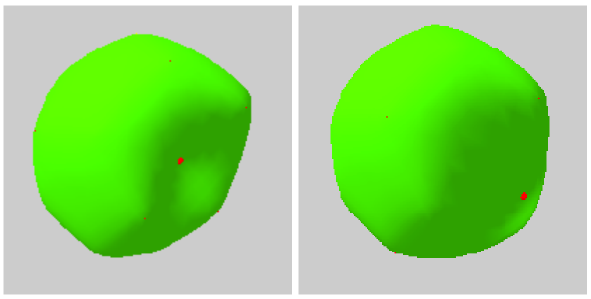

Figure 6: *The dodecahedron under external stress ($\boldsymbol{F}_{ext} = 1.153N$). The force is applied at the red point. The second image shows the model recoiling after the force is removed. The time difference between the two images is three seconds.*

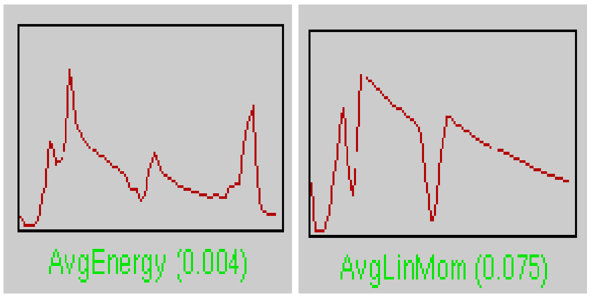

Figure 7: *The energy and the momentum plots for the dodecahedron when the forces are applied. The damping coefficient is 0.04. Application of multiple forces result in multiple spikes in energy. This is a over-damped system. The energy is expressed in $N.m$, and the momentum is expressed in $kg.m.s^{-1}$.*

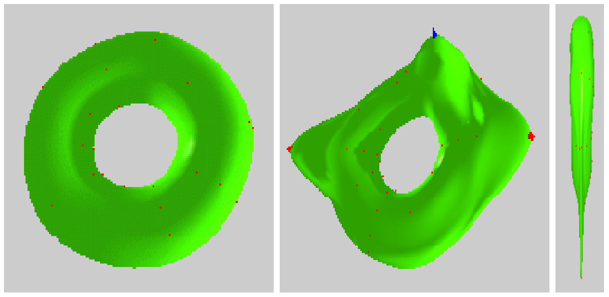

Figure 8: *Editing: The local rest-length (red points) and local rest-angle (blue point) for the torus are modified. The total internal stress is non-zero, since the restoring forces for the rest-length and rest-angle oppose each other. The third image presents the view from the side.*

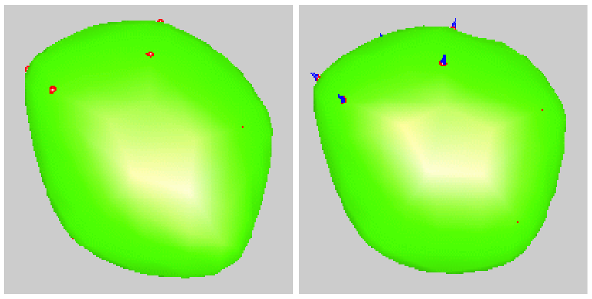

Figure 9: *Constraints: The dodecahedron is under position (red) constraints in Image I and under position and normal (blue) constraints in Image II. The same force has been applied in both cases. Note that the normals in the two cases are different.*

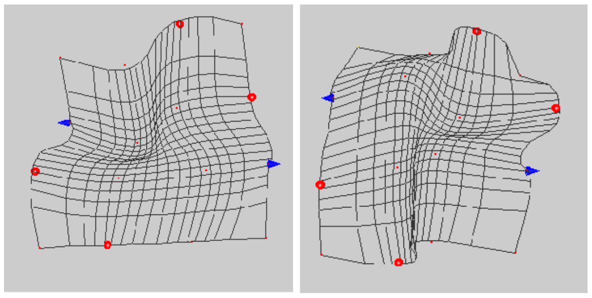

Figure 10: *Torsional forces on a planar mesh under multiple constraints. The torque ($\tau_{ext} = 1.3J/rad$) is applied near the central region of the mesh. The first image shows the mesh for an anti-clockwise torque. The second image is due to a clockwise torque.*