Deforming Geometric Volumes

Kinematics, Dynamics, Constraints, and Collisions

Jerry Tessendorf
Cinesite Digital Studios

February 12, 2002

Contents

1 Applications

These notes are intended as an outline of some geometric techniques that may be applicable to modeling and dynamics problems that include volumetric objects. Of particular interest are objects consisting of an enclosed volume, with the volume undergoing deformations while subjected to external forces, collisions, and boundary constraints that combine to reshape the volume dynamically. That description covers a very broad range of applications, and the techniques discussed here should hopefully be applicable to that breadth. In fact, we can expect that this deformable geometry approach can be applied to problems of modeling and simulating the motion of muscles, tissues, cloth, certain kinds of fluid (i.e. viscous fluids such as lava, slime, mud, goo, and under some conditions, water), softbodied organic objects, and possibly others not yet thought of.

There is a lot of effort being spent on simulating the dynamics of cloth, soft bodies, muscles, and lots of other phenomena where the object of interest is flexible and has an extended structure. In computer graphics, the universally adopted approach is a constrained particle dynamics[4, 9]. In this approach, the object is replaced by a cloud of particles which dynamically evolve using standard particle dynamics computations. In order to preserve the original object characteristics during the dynamics, constraining forces are added to the external forces that the particles are subjected to[5]. These constraining forces make the particles "aware" of the other nearby particles and encourage them to maintain reasonable locality with each other. The mathematical techniques used to add the constraint forces are traditional procedures formalized long ago in classical mechanics[6], and are known to be robust and produce well-defined results. However, numerical implementation of these methods with many particles in an object induces many technical difficulties:

1. The effect of the constraint forces is to couple the particles via a large matrix description. For N particles, the matrix is $N \times N$, and manipulating this matrix (computing its elements, taking inverses) can be difficult, time consuming, and sensitive to numerical precision.

2. The coupling requires modeling the constraint force in explicit detail. While some people view this as modeling the detailed fiber properties in a cloth, that is not the appropriate interpretation. The object in question may be some imagined material with properties totally alien to a cloth description. It seems like overkill to *require* a microscopic description of the material in order to compute the macroscopic behavior. A microscopic description would be appropriate as a way to specialize the macroscopic behavior for more subtle characteristics.

3. For incompressible objects, volume conservation is hard to implement in terms of constraining forces. The numerical implementation frequently must simply apply a pass in which the particle positions are adjusted ad hoc in order to keep the volume and/or surface area of the material constant.

4. The induced dynamics with constraints is hard to accurately compute numerically. The equations of motion tend to be stiff and sensitive to instabilities. There has been effort to suppress instabilities via implicit methods[10], with success.

Suppressing the instabilities via implicit methods has the added benefit of reducing the computational time by allowing larger time steps, and can also reduce the number of particles needed to complete a simulation to a desired accuracy.

Despite these technical difficulties, the theoretical machinery behind the particle-based approach is relatively easy to formulate and understand. This comes in handy when it is necessary to modify the algorithms in order to improve performance or handle some special case.

In constrast, the deformable geometry approach presented below has a very formidable theoretical machinery underlying it. To understand its workings, the reader needs a background in modern differential geometry, Lie Groups, and a little bit of differential equation solving. This background is not frequently found in computer graphics, so there is a bit of a barrier to understanding the techniques. To help in this, there is a recommended reading list for background material[1, 2, 3].

There are some significant benefits to the geometric approach however:

1. A geometric description relates much more directly to the type of object that the user is interested in than a particle description. There are no longer issues of how many particles to use, where to place them on the geometry, how to weight them, etc.

2. The method completely eliminates constraining forces needed to hold the geometry together. The geometric description assumes the role of keeping the object self-connected. This reduces the complexity of the dynamics problem.

3. No longer needing constraining forces, there is no longer a need to provide a microscoping description of the material that composes the object.

4. Volume conservation is easily and elegantly enforced within the geometric scheme. We derive below a general criterion for conserving volume, which can be universally applied at many stages of the computation. This criterion is independent of the object size or shape, the external forces and boundary conditions placed on it. For a broad class of dynamical problems, the criterion is transformed into direct constraints on the dynamics of the underlying geometry. In contrast to the particle dynamics approach, in which volume conservation is not a well-defined process, here it appears at a deep fundamental level.

5. The dynamics of the geometry does not introduce large matrices with difficult and time consuming manipulations.

6. The dynamics is stable, theoretically and numerically. This is partly because the lack of constraining forces simplifies the dynamics and eliminates large matrices. But it is also because the geometric description naturally “diagonalizes” the coupled motion of the extended object, reducing the equations of motion down to just those for the true degrees of freedom that must be computed.

So, while the mathematical machinery needed to understand deformable geometry is relatively involved, the numerical implementation of this approach is, in fact, considerably simpler than that for the particle-based approach, and more directly relates to the practical problem and solution sought by the user.

The approach we take in the following sections begins with an introduction to the geometric description of the volume. The concepts of tangent space, metric tensor, Connection, and holonomies are introduced. This introduction includes a description of how to solve some types of differential equations involving covariant derivatives, a handy tool later on. In section 3 we use these geometric tools to construct two alternative descriptions of how geometry is deformed, along with procedures for computing either one of these descriptions from the other. Along the way, we find the criterion for volume conservation comes neatly out of these representations.

With a solid method of describing deformations geometrically, we turn to the dynamical problem and extract from Newton's equation dynamical equations for the underlying geometric behavior - specifically for the Connection. Here we see another strength of this approach, because the dynamical equations are related just to the actually degrees of freedom that are available, and are shown explicitly to be stable.

The last section is devoted to a discussion of collisions between our deformable object and other objects. The other objects may be deformable or not. In keeping with the geometric philosophy, we introduce a remarkable quantity – the *Gauss Linking Number* – as a collision detector between extended objects.

2 Geometric Description of Volumes

A volume is modeled as a collection of points in space which have a linkage two each other. One way of establishing a linkage is to set up a nearest neighbor network, so that each point has a list of adjacent points in the volume. The network should be thought of as fixed for all time, even when the volume is deformed. Under deformation, the concept of the nearest neighbor list becomes a little muddled, because the collection of nearest points in a deformed geometry may be substantially different from the collection in the original "undeformed" state. Nevertheless we want to maintain that linkage as an underlying pillar keeping the description of the deformed geometry simple and compact.

So we imagine that the volume spans a (discrete or continuous) collection of points labelled $\mathbf{x}(\vec{u})$, with "material coordinates" $\vec{u} = (u_1, u_2, u_3)$. The material coordinates are analogous to the u, v coordinates of a NURBS surface: no matter how much the NURBS surface is stretched and distorted, the u, v coordinates act like a spot on the surface that rides along with the distortion. The collection of knots build a grid pattern across the surface which rides on the surface but is not broken by the deformations. This grid pattern serves also to connect points on the surface and maintain the surface topology. From this grid pattern some geometrically useful quantities can be calculated, in particular, the tangent plane derivatives, embedded metric tensor, and others.

Material coordinates of the volume serve the same purpose as the u, v coordinates of a nurbs surface. Points in the volume may move in space relative to one another, while their material coordinates ride along with the deformations and maintain their connectedness relationship to each other. However, we do not require that the material coordinates form a grid of a particular topology. It is enough that they allow us to compute quantities called the tangent space basis vectors and the Christoffel connection.

This section presents a model of volumetric geometry, sets down some definitions

and conventions, and sets the stage for further discussion of deformation of the geometry. Some suggested reading material on this topic might include references [1], [2], and [3].

Tangent Space

Just as a nurbs surface has a tangent plane at each point of the surface, so too a volume has a three dimensional tangent space at each point of the volume. The tangent space is described by three vectors

$$\mathbf{x}_a(\vec{u}) \equiv \frac{\partial \mathbf{x}(\vec{u})}{\partial u_a} \quad (a = 1, 2, 3) \; . \tag{1}$$

These three tangent vectors form a basis: any vector in the tangent space can be decomposed into a sum of these three with suitable coefficients. This is an extremely important property for constructing the entire geometric deformation program in this paper. However, the $\mathbf{x}_a$ are not orthogonal to each other, and they are not normalized to unity. Because of this, we must introduce the metric tensor,

$$g_{ab}(\vec{u}) = \mathbf{x}_a(\vec{u}) \cdot \mathbf{x}_b(\vec{u}) \tag{2}$$

which will also be very important in this paper. The definition of the metric tensor, which is a 3×3 matrix, also brings up two additional definitions of use later: the inverse metric tensor

$$g^{ab} \equiv \left(g^{-1}\right)_{ab} \tag{3}$$

such that

$$\sum_{c=1}^{3} g^{ac}(\vec{u}) \; g_{cb}(\vec{u}) = \delta^a_b \tag{4}$$

(with δ^a_b just the usual old-fashion Kronecker delta). Using the inverse metric, dual space vectors are built as

$$\mathbf{x}^a(\vec{u}) \equiv \sum_b \; g^{ab}(\vec{u}) \; \mathbf{x}_b(\vec{u}) \; . \tag{5}$$

Of what use is a dual vector? Well, applying these definitions, we see that the dual vectors "orthonormalize" the basis of the tangent space, i.e.

$$\mathbf{x}^a(\vec{u}) \; \cdot \; \mathbf{x}_b(\vec{u}) = \delta^a_b \; . \tag{6}$$

Christoffel Connection

The last geometric quantity of value in this scheme is the Connection, and in particular the Christoffel Connection $\Gamma^a_{bc}(\vec{u})$, which is related to the Riemannian curvature of the volume, and given by the expression

$$\frac{\partial^2 \mathbf{x}(\vec{u})}{\partial u_a \; \partial u_b} \equiv \mathbf{x}_{ab}(\vec{u}) \equiv \sum_c \; \Gamma^c_{ab}(\vec{u}) \; \mathbf{x}_c(\vec{u}) \tag{7}$$

This expression simply says that any vector at the material coordinate $\vec{u}$ can be spanned by the basis vectors $\mathbf{x}_a$ in the tangent space, with suitable labels for the coefficients. Making use of dual vectors, we can compute the connection directly as

$$\Gamma^c_{ba}(\vec{u}) = \mathbf{x}^c(\vec{u}) \ \cdot \ \mathbf{x}_{ba}(\vec{u}) \ . \tag{8}$$

Covariant Differentiation

The connection plays a critical role in the geometric description of volumes. For example, suppose we have a vector $\mathbf{f}(\vec{u})$ in the tangent space at the point $\vec{u}$. In complete generality, we can expand it as

$$\mathbf{f}(\vec{u}) = \sum_b \ f^b(\vec{u}) \ \mathbf{x}_b(\vec{u}) \tag{9}$$

Suppose now we wish to see how $\mathbf{f}$ changes as we move in the volume. The material derivative is

$$\begin{aligned} \frac{\partial \mathbf{f}(\vec{u})}{\partial u_a} &= \sum_b \left\{ \frac{\partial f^b(\vec{u})}{\partial u_a} \ \mathbf{x}_b(\vec{u}) + f^b(\vec{u}) \ \mathbf{x}_{ab}(\vec{u}) \right\} \\ &= \sum_b \left\{ \frac{\partial f^b(\vec{u})}{\partial u_a} \ + \sum_c f^c(\vec{u}) \ \Gamma^b_{ca}(\vec{u}) \right\} \ \mathbf{x}_b \\ &= \sum_b \ f^b_{;a}(\vec{u}) \ \mathbf{x}_b(\vec{u}) \end{aligned} \tag{10}$$

The notation

$$f^b_{;a}(\vec{u}) \equiv \frac{\partial f^b(\vec{u})}{\partial u_a} + \sum_c f^c(\vec{u}) \ \Gamma^b_{ca}(\vec{u}) \tag{11}$$

is called the *covariant derivative* of the coefficients f^b.

Solving Simple Covariant Differential Equations I: Solution Along a Path

We will encounter covariant derivatives in the form of differential equations. This section presents the concepts needed to solve some simple covariant differential equations. The concepts involve solving the equations along pre-defined paths through the material coordinates.

We begin the introduction of these concepts by picking a basic but very useful example problem. Suppose you are faced with solving the differential equation

$$f^b_{;a}(\vec{u}) = G^b_a(\vec{u}) \ , \tag{12}$$

where G^b_a is some prescribed driving term, and the Christoffel Connection $\Gamma^a_{bc}(\vec{u})$ is known.

The key to solving this differential equation is to solve it along a path through the volume. So we chose some convenient path of the form $\vec{u} = \vec{\gamma}(\lambda)$, with λ being the

path variable running from λ_0 and the start of the path and λ_1 at the end of the path. Evaluating equation 12 along that path, and noting that

$$\frac{df^b(\vec{\gamma}(\lambda))}{d\lambda} = \sum_a \frac{d\gamma_a(\lambda)}{d\lambda} \frac{\partial f^b(\vec{\gamma}(\lambda))}{\partial \gamma_a} \tag{13}$$

equation 12 becomes

$$\frac{df^b(\vec{\gamma}(\lambda))}{d\lambda} + \sum_c \left(\sum_a \Gamma^b_{ac}(\vec{\gamma}(\lambda)) \, \frac{d\gamma_a(\lambda)}{d\lambda} \right) \, f^c(\vec{\gamma}(\lambda)) = \sum_a \, G^b_a(\vec{\gamma}(\lambda)) \frac{d\gamma_a(\lambda)}{d\lambda} \tag{14}$$

To simplify the look of this, we can define the quantities:

$$A^b_c(\vec{\gamma}(\lambda)) \quad \equiv \quad \sum_a \, \Gamma^b_{ac}(\vec{\gamma}(\lambda)) \, \frac{d\gamma_a(\lambda)}{d\lambda} \tag{15}$$

$$G^b(\vec{\gamma}(\lambda)) \quad \equiv \quad \sum_a \, G^b_a(\vec{\gamma}(\lambda)) \frac{d\gamma_a(\lambda)}{d\lambda} \tag{16}$$

which reduces the nasty appearance a bit to

$$\frac{df^b(\vec{\gamma}(\lambda))}{d\lambda} + \sum_c \, A^b_c(\vec{\gamma}(\lambda)) \, f^c(\vec{\gamma}(\lambda)) = G^b(\vec{\gamma}(\lambda)) \tag{17}$$

This last equation has an exact solution in terms of the ordered-exponential integral of the matrix $\mathbf{A}$ with members $A^a_b = \mathbf{A}_{ab}$. This ordered exponential is called the *holonomy*, and has the form:

$$H^b_a(\lambda, \lambda_0) \equiv \left(\exp \left(- \int_{\lambda_0}^{\lambda} \, d\lambda' \, \mathbf{A}(\vec{\gamma}(\lambda')) \right) \right)_{+ \, ba} \tag{18}$$

Using the holonomy, the exact solution of equation 12 is

$$f^b(\vec{\gamma}(\lambda)) = \sum_a \, H^b_a(\lambda, \lambda_0) \, f^a(\vec{\gamma}(\lambda_0)) \, + \, \int_{\lambda_0}^{\lambda} d\lambda' \, \sum_a \, H^b_a(\lambda, \lambda') G^a(\vec{\gamma}(\lambda')) \tag{19}$$

The holonomy is an important object in many physical sciences that employ geometric techniques. As with those fields, we will find its use crucial again and again in the results we obtain below. Fundamentally, the holonomy is the tool by which the Connection is used to actually connect two points within the geometry, and transfer information between them. But while the Connection is a locally defined quantity, the holonomy is defined relative to a path specified through the volume. This is not a weakness, because any path may be used, and because it provides a rigorous mathematical framework on which to design a numerical implementation, which ultimately would employ paths of some kind through the volume in order to integrate quantities of interest.

This method of solving the covariant differential equation along a path is quite general and elegant, but some questions arise in its practical implementation:

- By what criteria do we choose paths? Should one path be used which connects all of the points from one starting point, or many paths eminating from multiple points?
- If there are many paths, how do we treat path intersections?
- How do we lay out paths to respect boundary conditions and/or constraints on the geometry?
- By what criteria do we answer these questions?

These questions do not imply that covariant integration along paths is incorrect, but the approach is simply so general that it does not deal with some practical questions of implementation.

One way around these questions is to find an alternate method of integrating the covariant differential equation over the whole volume without imposing the construction of paths. The method discussed below is just such an alternative. In a sense, it gets around the problem by systematically integrating along *every* possible path in short segments. It accomplishes this, and handles constrained points, by rephrasing the integration problem from a "push" of data from one point to another to a "gather" of data into a point from nearby points. The "gather" outlook on the solution also requires the addition of an iterative solver – somewhat analogous to a solver for Poisson's equation – whose convergence properties will need to be verified carefully. In this section we discuss each of these three methods in the context of our geometric problem. However, what is known about the convergence behavior of these methods comes from research on discrete Poisson equations, so we should expect some deviation from those results in this problem. Also, we simplify considerably the Congjugate Gradient method for the specifics of the problem at hand.

These "pull"-based iterative methods also have the advantage that it is trivially easy to impose boundary conditions or constraints at individual vertices. If a vertex is constrained by the application to be at a fixed position, that is accomplished by simply not updating its value in the iterative process. If the constraint is more of a collision-style one – e.g. the vertex may not be located beyond a wall or outside of a domain – then the vertex position is updated in the iterations, but between iterations its position is corrected if necessary to satisfy the constraint.

Jacobi Method

We begin with equation 19, which we cast into a solution for small steps in λ:

$$f^b(\vec{\gamma}(\lambda + \Delta\lambda)) = \sum_a H^b_a(\lambda + \Delta\lambda, \lambda)\, f^a(\vec{\gamma}(\lambda)) \;+\; \Delta\lambda\, G^b(\vec{\gamma}(\lambda)) \qquad (20)$$

This solution is a kind of "push" technique: given the value of f and G at the vertex at $\vec{\gamma}(\lambda)$, we push those values to the vertex at $\vec{\gamma}(\lambda + \Delta\lambda)$ and compute the value of f there.

Now let us reorganize this into a "pull" philosophy. Suppose we wish to compute the value of f at the vertex $\vec{\gamma}$, given knowledge of f at neighboring vertices. For a vertex at $\vec{\gamma} + \delta\vec{\gamma}$, the contribution is

$$\sum_a H_a^b(\vec{\gamma}, \vec{\gamma} + \delta\vec{\gamma}) \left[f^a(\vec{\gamma} + \delta\vec{\gamma}) - \delta G^b(\vec{\gamma}, \delta\vec{\gamma}) \right] \tag{21}$$

where the holonomy and driving term now look like:

$$H_a^b(\vec{\gamma}, \vec{\gamma} + \delta\vec{\gamma}) = (\exp(\delta \mathbf{A}(\vec{\gamma}, \delta\vec{\gamma})))_{ba} \tag{22}$$

$$\delta A_c^b(\vec{\gamma}, \delta\vec{\gamma}) = \sum_a \Gamma_{ac}^b(\vec{\gamma})\, \delta\gamma_a \tag{23}$$

$$\delta G^b(\vec{\gamma}, \delta\vec{\gamma}) = \sum_a G_a^b(\vec{\gamma}) \delta\gamma_a \tag{24}$$

This is the contribution of one of the neighbors. For the complete set of neighbors, the solution comes from summing over all of the neighbors

$$f^b(\vec{\gamma}) = \sum_{\delta\gamma} \left\{ \sum_a H_a^b(\vec{\gamma}, \vec{\gamma} + \delta\vec{\gamma}) \left[f^a(\vec{\gamma} + \delta\vec{\gamma}) - \delta G^b(\vec{\gamma}, \delta\vec{\gamma}) \right] \right\} \tag{25}$$

The computation of this sum does not by itself solve the covariant differential equation, because some of the values of f inside the sum are based on old data and do not reflect the impact of deformations. This situation is similar to variaus approaches for solving Poisson's equation, such as the Jacobi, SOR, and Conjugate Gradient methods. This full solution is obtained iteratively: (1) Evaluation equation 25 over the entire volume, then iterate and do it again using the results from that iteration.

Of the various methods of implementing iteration, the simplest is the Jacobi method. In this method, the $m - th$ iteration is computed from iteration $m - 1$ as

$$f_m^b(\vec{\gamma}) = \sum_{\delta\gamma} \left\{ \sum_a H_a^b(\vec{\gamma}, \vec{\gamma} + \delta\vec{\gamma}) \left[f_{m-1}^a(\vec{\gamma} + \delta\vec{\gamma}) - \delta G^b(\vec{\gamma}, \delta\vec{\gamma}) \right] \right\} \tag{26}$$

For our discuss below we want to refer to this particular expression, so we label the Jacobi solution by $[f_m^b(\vec{\gamma})]_J$. Two copies of f are maintained at each vertex: the previous one and the new one being computed. Although this is the simplest algorithm, it is also the slowest: For N vertices in the volume, the number of iterations that must be performed is proportional to N^2. This can be a very slow iteration process.

Successive Over Relaxation (SOR) Method

To begin introducing the method of Successive Over Relaxation (SOR), note that we could keep only one copy of f if we update it in place, so that in one iteration the values

from the neighbor vertices are either from the previous iteration, or from the present iteration if available. In that case we might write the iteration as

$$f_m^b(\vec{\gamma}) = \sum_{\delta\gamma} \left\{ \sum_a H_a^b(\vec{\gamma}, \vec{\gamma} + \delta\vec{\gamma}) \left[f^a_{\mathrm{latest}(m)}(\vec{\gamma} + \delta\vec{\gamma}) - \delta G^b(\vec{\gamma}, \delta\vec{\gamma}) \right] \right\} \tag{27}$$

where $\mathrm{latest}(m)$ means $m-1$ unless a value is available from the $m-th$ iteration. This approach does in fact improve convergence speed of the iterations. However, it can be improved more. Note that this version of the algorithm can be re-written as the Jacobi version in equation 26 plus a correction term

$$f_m^b(\vec{\gamma}) = [f_m^b(\vec{\gamma})]_J + \sum_{\delta\gamma} e_m^b(\vec{\gamma}, \delta\vec{\gamma}) \tag{28}$$

where the correction is

$$e_m^b(\vec{\gamma}, \delta\vec{\gamma}) = \sum_a H_a^b(\vec{\gamma}, \vec{\gamma} + \delta\vec{\gamma}) \left(f^a_{\mathrm{latest}(m)}(\vec{\gamma} + \delta\vec{\gamma}) - f^a_{m-1}(\vec{\gamma} + \delta\vec{\gamma}) \right) \tag{29}$$

The key to SOR is to modify how the correction term applies in equation 28. Note that if we apply a gain coefficient w to the correction term, i.e. as

$$w \sum_{\delta\gamma} e_m^b(\vec{\gamma}, \delta\vec{\gamma}) \tag{30}$$

Then the gain $w = 0$ reclaims the original Jacobi method while $w = 1$ gives the updates using $\mathrm{latest}(m)$. The SOR method is to increase the value of the gain higher, to a value between 1 and 2. In this range, depending on the initial distribution, vertex grid layout, and other factors, convergence can be faster than for $w \leq 1$, and in fact the number of iterations needed to achieve convergence becomes proportional to N, the number of points in the grid. This is much faster than just the Jacobi method alone. However, we have no particular criteria for how to choose the value of the gain w.

Conjugate Gradient (CG) Method

The CG method adds logic to the SOR method that automatically chooses the "best" gain value for the problem. Suppose we have chosen a value for w by some criteria, and denote that "precursor" value as w_-. Using it, we generate the next iteration at all vertices, and denote the solution f^b_{m-}:

$$f^b_{m-}(\vec{\gamma}) = [f_m^b(\vec{\gamma})]_J + w_- \sum_{\delta\gamma} e_m^b(\vec{\gamma}, \delta\vec{\gamma}) \tag{31}$$

We can ask whether there is a better value for w that causes faster convergence. By faster convergence, we mean that the function

$$\Xi^2 \equiv \sum_\gamma \sum_b \left(f^b_{m-}(\vec{\gamma}) - f^b_{m-1}(\vec{\gamma}) \right)^2 \tag{32}$$

is a small as possible. Allowing w to vary from its initial guess w_-, minimizing Ξ^2 gives a new estimate for the gain:

$$w_+ = -\frac{\sum_\gamma \sum_b \left([f^b_m(\vec{\gamma})]_J - f^b_{m-1}(\vec{\gamma})\right) \sum_{\delta\gamma} e^b_m(\vec{\gamma}, \delta\vec{\gamma})}{\sum_\gamma \sum_b \left(\sum_{\delta\gamma} e^b_m(\vec{\gamma}, \delta\vec{\gamma})\right)^2} \tag{33}$$

With this new gain estimate, we can calculate a new value for each $f^b_m(\vec{\gamma})$ that is more accurate:

$$f^b_{m+}(\vec{\gamma}) = [f^b_m(\vec{\gamma})]_J + w_+ \sum_{\delta\gamma} e^b_m(\vec{\gamma}, \delta\vec{\gamma}) \tag{34}$$

Effectively, this procedure turns the gain factor w into an iterated quantity also.

Overall, the CG method is accomplished with the following set of steps.

1. Using the value w_{m-1} from the previous step, compute $f^b_{m-}(\vec{\gamma})$, the Jacobian estimate $[f^b_m(\vec{\gamma})]_J$, and the error correction $\sum_{\delta\gamma} e^b_m(\vec{\gamma}, \delta\vec{\gamma})$ as in equation 31 at each vertex.

2. Compute the new gain factor w_m from equation 33.

3. Compute the improved interated solution f^b_{m+} using the new gain factor in equation 34.

3 Geometric Reconstruction of Deformations

In this section we look at two separate ways of describing the deformation of a volumetic shape. Both techniques assume that the topology of the volume does not change, but both allow the shape to stretch, squash, and distort in almost unlimited ways. The topological restriction primarily means that chunks of material are not separated from or joined into the volume.

One of the representations describes the deformation in terms of a displacement vector at each material point. This is useful for evaluating the effects of several types of dynamics, as well as collisions and some constraints. But for motion involving true distortions of the volume, this representation is very difficult to manipulate, because of the constraints of the topology. The other representation is complementary, in that it computes directly the dynamical evolution of the Connection at all material points in the volume due to twisting, stretching, distorting forces. The combination of both into one comprehensive description provides great power and flexibility in computing the complete dynamical problem in the presence of collisions and other constraints.

The remainder of this section is devoted to quantifying these two representations, showing how they are related and can be made to work together, and showing how together they provide a convenient method of exactly constraining the geometry to conserve the volume of the material, regardless of the forces, boundary conditions, and constraints.

The Displacement Vector Representation of Deformation

This representation is a very straightforward method. It simply says that when a piece of geometry is deformed, the points it occupies is space are translated various amounts. If the "undeformed" position for the material point $\vec{u}$ is $\mathbf{x}(\vec{u})$, and the deformed position is $\mathbf{X}(\vec{u})$, the deformation is defined to be $\delta\mathbf{x}(\vec{u})$ by

$$\mathbf{X}(\vec{u}) = \mathbf{x}(\vec{u}) + \delta\mathbf{x}(\vec{u}) \tag{35}$$

This appears to be nothing more than a definition, which could make it difficult to use this to any practical end. But the second part of this representation comes from remembering that the tangent vectors span the 3d space, so be can write

$$\delta\mathbf{x}(\vec{u}) = \sum_a \epsilon^a(\vec{u})\ \mathbf{x}_a(\vec{u}) \tag{36}$$

The practical task now in this representation is manipulating the deformation coefficients $\epsilon^a(\vec{u})$.

At this stage we can evaluate some of the relationships between the deformed and undeformed states of the geometry. One important quantity is the deformed tangent space. Taking the derivative of $\mathbf{X}(\vec{u})$, and using equations 10 and 11, we obtain

$$\mathbf{X}_a(\vec{u}) = \sum_b \left(\delta^b_a\ +\ \epsilon^b_{;a}(\vec{u})\right)\ \mathbf{x}_b(\vec{u}) \tag{37}$$

The deformed metric tensor is then related to the undeformed metric tensor by a transformation:

$$\mathbf{X}_a(\vec{u}) \cdot \mathbf{X}_b(\vec{u}) = \overline{g}_{ab}(\vec{u}) = \sum_{cd} M^c_a(\vec{u})\ g_{cd}(\vec{u})\ M^d_b(\vec{u})\ , \tag{38}$$

with the transformation matrix M defined as

$$M^b_a(\vec{u}) \equiv \delta^b_a\ +\ \epsilon^b_{;a}(\vec{u})\ . \tag{39}$$

This expression will reappear later when we discuss volume conservation with the deformations. However, its appearance in this representation already portends the importance of the Connection in the deformation problem. In this case, it appears in the covariant derivative $\epsilon^b_{;a}(\vec{u})$. So the second representation below focuses more directly on the Connection as the basic descriptor of deformation.

Conserving Volume – The Lie Group SL(3)

Before proceeding to the second representation in terms of the Connection, it is worthwhile here to present an important consequence of the Displacement Vector representation. For many applications, it is very important to conserve the volume of the geometry under deformations. Further, it is important to conserve more than just the total volume; small portions of the volume should also conserve their volume in deformations. This is true for a wide range of materials such as tissue, water, cloth, and

many others. So volume conservation must be applied at the local scale of a geometry. The Displacement Vector representation provides a very natural way of characterizing the impact of a deformation on the volume at the local level.

The key to describing the volume is the metric tensor introduced earlier. Suppose a region of the material coordinates that forms a volume is labeled R. This region need not be the entire geometric object – it can be any "chunk" of that geometry. The volume of that region is

$$V = \int_R d^3u \, \sqrt{g(\vec{u})} \tag{40}$$

where g is the determinant of the metric tensor defined in equation 2. Upon deformation, the new volume follows from the new metric tensor:

$$\overline{V} = \int_R d^3u \, \sqrt{\overline{g}(\vec{u})} \tag{41}$$

The condition for conserving the volume under deformation is just the requirement that $V = \overline{V}$. Since the region R is any region in the volume, the only way to guarantee conservation is for the determinant of the metric to be unaffected by the deformation, i.e. $g(\vec{u}) = \overline{g}(\vec{u})$. This is a very general, strong, and restrictive statement about how deformations can function. But in the Displacement Vector representation, this restriction becomes a remarkably simple condition.

In the Displacement Vector representation, we can evaluate the determinant of the deformed metric in a very simple way. To see it, we introduce some matrix notation. Define the metric tensor matrix $\mathbf{g}$ with components $\mathbf{g}_{ab} = g_{ab}$. The matrix $\overline{\mathbf{g}}$ is similarly defined in terms of the deformed metric tensor. Also define the matrix $\mathbf{M}_{ab} = M^a_b$. Then the relationship in equation 38 between the metric and deformed metric has the form

$$\overline{\mathbf{g}} = \mathbf{M} \cdot \mathbf{g} \cdot \mathbf{M}^T \tag{42}$$

With this form, it is easy to see that the determinant of the deformed metric is easily evaluated as

$$\overline{g} = g \left(\det(\mathbf{M})^2 \right) \tag{43}$$

From this we have the elegant result that volume is conserved under deformation as long as $\det \mathbf{M} = \pm 1$. A value of -1 is not a good choice, since it causes the geometry to be inverted, and so will not appear to be a continous evolution of the structure. So, the condition we have for the conservation of volume under deformation is

$$\det \mathbf{M} = 1 \tag{44}$$

This result for guaranteed volume conservation is a very useful and remarkable result for several reasons:

1. The condition depends only on the deformation quantity $\epsilon^b_{;a}(\vec{u})$, and is independent of the actual values of the metric before and after deformation. It does have a dependence on the original geometry via the Connection term in the covariant derivative.

2. As we will see shortly, the volume conservation condition restricts most of the degrees of freedom in $\epsilon^b_{;a}(\vec{u})$. In general, this quantity has nine independent degrees of freedom, but volume conservation collapses that number down to only three.

The Connection Representation of Deformation

The second representation looks directly at the underlying geometric description of the volume. Here the Connection is the paramount means of controling/describing the action. So any deformation of the geometry will change the Connection in some way. So we define a deformation as that change of the Connection:

$$\overline{\Gamma}^a_{bc}(\vec{u}) = \Gamma^a_{bc}(\vec{u}) + \delta\Gamma^a_{bc}(\vec{u}) \tag{45}$$

Here $\overline{\Gamma}^a_{bc}(\vec{u})$ is the new Connection after the deformation $\delta\Gamma^a_{bc}(\vec{u})$ is applied to the old Connection $\Gamma^a_{bc}(\vec{u})$.

This representation has the very nice advantage that much of the dynamics of volumes is naturally expressed as dynamical equations for the Connection (see section 4), so it is natural to compute directly changes in $\Gamma^a_{bc}(\vec{u})$ due to applied forces.

There are however two potential drawbacks to using the Connection representation directly:

1. The Connection representation does not lend itself to intuitive explanation of what a deformation has done to the geometry.

2. It is very difficult to force volume conservation directly in terms of the Connection alone, especially compared to the very clean results of equation 44. In principle, there are 24 degrees of freedom in the Connection ($3 \times 3 \times 3 - 3$) because the Connection serves many uses in the geometry. This makes isolating the three degrees of freedom that must be frozen to insure volume conservation is a non-trivial task. (Although in our context of deformations we obtain exact requirements on the deformation of the Connection that insure volume conservation in section 3.)

Fortunately, we actually have the freedom to use both of these representations, and in particular we can mix their usage together. So for tasks involving dynamics, the Connection representation will frequently be the best choice, while in issues of constraints, boundary conditions, and volume conservation, the Displacement Vector representation will be preferred. But in order to work in a mixed representation environment, we need to understand the precise relationship between the two, especially how to start in one representation and end up in the other. This is the subject of the next two subsections.

Relationship Between the Representations: Connection from Displacement Vector

We have already almost generated the complete process of computing the deformed connection from the displacement vector. A few more steps will lead us there.

Since the Connection is defined in terms of the derivative of the tangent space vectors, we start with equation 37, written with the current notation:

$$\mathbf{X}_a(\vec{u}) = \sum_c \; M_a^c(\vec{u}) \; \mathbf{x}_c(\vec{u}) \; . \tag{46}$$

The second derivative with respect to a material coordinate is

$$\frac{\partial \mathbf{X}_a(\vec{u})}{\partial u_b} \equiv \mathbf{X}_{ab}(\vec{u}) = \sum_c \; M_{a;b}^c(\vec{u}) \; \mathbf{x}_c(\vec{u}) \tag{47}$$

with the covariant derivative of M having the explicit form

$$M_{a;b}^c(\vec{u}) = \frac{\partial M_a^c(\vec{u})}{\partial u_b} + \sum_d \; \Gamma_{bd}^c(\vec{u}) \; M_a^d(\vec{u}) \; . \tag{48}$$

Now, according to equation 8, the deformed Connection is given by

$$\overline{\Gamma}_{bc}^a(\vec{u}) = \mathbf{X}^a(\vec{u}) \cdot \mathbf{X}_{bc}(\vec{u}) \; . \tag{49}$$

The dual vector $\mathbf{X}^a$ comes from equation 46 as

$$\mathbf{X}^a(\vec{u}) = \sum_c \; Q_c^a(\vec{u}) \; \mathbf{x}^c(\vec{u}) \; , \tag{50}$$

with the Q_c^a forming the matrix inverse of M:

$$\mathbf{M} \cdot \mathbf{Q} = \mathbf{1} \; \leftrightarrow \; \sum_b \; M_a^b \, Q_b^c = \delta_a^c \tag{51}$$

Assembling all of this the deformed Connection is

$$\overline{\Gamma}_{ab}^c(\vec{u}) = \sum_d \; Q_d^c(\vec{u}) \; M_{a;b}^d(\vec{u}) \tag{52}$$

So in order to compute the Connection from the displacement vector, two derivatives (in covariant form) must be taken of ϵ^b. In the discussion later, we will see that we can always do this as long as the initial data is C^2, with good initial numerical values of these derivatives.

Relationship Between the Representations: Displacement Vector from Connection

Now we want to work in the opposite direction, assuming we have obtained the deformed value $\overline{\Gamma}_{bc}^a$ at all points $\vec{u}$ of the material, and we wish to derive the corresponding displacement vector. We begin this approach with the spot at which we left the previous one: equation 52, which we now write slightly differently:

$$M_{a;b}^c(\vec{u}) - \sum_d \; \overline{\Gamma}_{ab}^d(\vec{u}) \; M_d^c(\vec{u}) = 0 \tag{53}$$

This equation can be solved using the integration along paths approach used in section 2. We also bring in more matrix notation to make the solution somewhat clearer to understand. So, picking a path $\vec{u} = \vec{\gamma}(\lambda)$ through the volume, we multiply equation 53 by $\frac{d\vec{\gamma}(\lambda)}{d\lambda}$ to obtain

$$\frac{dM_a^c(\vec{\gamma}(\lambda))}{d\lambda} + \sum_b A_b^c(\vec{\gamma}(\lambda))\, M_a^b(\vec{\gamma}(\lambda)) - \sum_b \overline{A}_a^b(\vec{\gamma}(\lambda))\, M_b^c(\vec{\gamma}(\lambda)) = 0 \qquad (54)$$

and $\overline{A}$ is associated with the deformed Connection

$$\overline{A}_b^a = \sum_c \overline{\Gamma}_{bc}^a \, \frac{d\gamma_c(\lambda)}{d\lambda} \qquad (55)$$

Using the matrix notation previously introduced, this equation has the more compact and transparent form

$$\frac{d\mathbf{M}}{d\lambda} + \mathbf{A} \cdot \mathbf{M} - \mathbf{M} \cdot \overline{\mathbf{A}} \qquad (56)$$

which in turn has the general solution

$$\mathbf{M}(\lambda) = \mathbf{H}(\lambda, \lambda_0) \cdot \mathbf{M}(\lambda_0) \cdot \overline{\mathbf{H}}(\lambda_0, \lambda) \; . \qquad (57)$$

This solution uses the holonomy introduced earlier, with the matrix notation $H_b^a = \mathbf{H}_{ab}$.

From this expression for $\mathbf{M}$, the solution for the deformation components ϵ^a follows directly. Again introducing some abbreviated notation, the set of the ϵ^a is a vector $\vec{\epsilon}$ which satisfies the differential equation

$$\frac{d\vec{\epsilon}(\lambda)}{d\lambda} + \mathbf{A}(\lambda) \cdot \vec{\epsilon}(\lambda) = (\mathbf{M}(\lambda) - \mathbf{1}) \cdot \frac{d\vec{\gamma}(\lambda)}{d\lambda} \; . \qquad (58)$$

which has the general solution

$$\vec{\epsilon}(\lambda) = \mathbf{H}(\lambda, \lambda_0) \cdot \vec{\epsilon}(\lambda_0) + \int_{\lambda_0}^{\lambda} d\lambda' \, \mathbf{H}(\lambda, \lambda') \cdot (\mathbf{M}(\lambda') - \mathbf{1}) \cdot \frac{d\vec{\gamma}(\lambda')}{d\lambda'} \qquad (59)$$

Equations 57 and 59 are the complete solution to computing points on the deformed geometry, once the deformation $\delta\Gamma_{bc}^a$ is known. As discussed in the section on integration of covariant differential equations, an alternative approach to the solution for $\vec{\epsilon}$ is the use of the SOR or CG iterative methods. Numerically, these techniques are probably superior to layout a set of paths and integrating.

Volume Conservation Revisited

In the previous section a discussion of volume conserving deformation concluded that the requirement of constant volume – both locally and globally – over the entire volume is satisfied if and only if the quantity $\mathbf{M}$ is a member of the Lie Group $SL(3)$. In light of the solutions for $\mathbf{M}$ and $\vec{\epsilon}$ just obtained, it is worthwhile investigating the solution for methods to assure volume conservation.

The place to begin is to combine the volume conservation condition of equation 44 with the solution for $\mathbf{M}$ in equation 57, to obtain

$$\begin{aligned} 1 &= \det \mathbf{M}(\lambda) \\ &= \det \mathbf{M}(\lambda_0)\ \det \mathbf{H}(\lambda, \lambda_0)\ \det \overline{\mathbf{H}}(\lambda_0, \lambda) \\ &= \det \mathbf{M}(\lambda_0)\ \det \left(\mathbf{H}(\lambda, \lambda_0) \cdot \overline{\mathbf{H}}(\lambda_0, \lambda)\right) \end{aligned} \tag{60}$$

Since we want the volume conserved everywhere, $\det \mathbf{M}(\lambda_0) = 1$ as well, and the volume conservation condition reduced to the requirement that the matrix $\mathbf{H}(\lambda, \lambda_0) \cdot \overline{\mathbf{H}}(\lambda_0, \lambda)$ is also a member of $SL(3)$. However, because of the relationship $\overline{\Gamma}^a_{bc} = \Gamma^a_{bc} + \delta\Gamma^a_{bc}$, there is the relationship $\overline{\mathbf{A}} = \mathbf{A} + \delta\mathbf{A}$, and ultimately the result

$$\mathbf{H}(\lambda, \lambda_0) \cdot \overline{\mathbf{H}}(\lambda_0, \lambda) = \left(\exp\left(\int_{\lambda_0}^{\lambda} d\lambda'\ \delta\mathbf{A}(\lambda') \right) \right)_+ \tag{61}$$

which is independent of the original undeformed Connection. In addition, volume conservation now requires that $\exp(\int \delta\mathbf{A})$ is a member of $SL(3)$.

The sections on dynamics and constraints will continue this discussion of volume conservation, touching on aspects that are peculiar to those specific situations.

Relationship to Continuum Mechanics

There is an alternative analytical scheme for computing the affects of deformations in the form of Continuum Mechanics [7]. In continuum mechanics, the deformations are described in terms of the *elastic strain tensor* $\tau_{ab}(\vec{u})$, which is related to the covariant derivative of the deformation

$$\tau_{ab}(\vec{u}) \equiv \frac{1}{2} \left(\epsilon^a_{;b}(\vec{u}) + \epsilon^b_{;a}(\vec{u}) \right) \ . \tag{62}$$

Forces that cause deformation in continuum mechanics are modelled in terms of their behavior on the elastic strain tensor. These force models introduce coefficients such as the *bulk modulus*, *shear modulus*, and *modulus of rigidity*. These quantities need not arise in our description because volume is conserved in the formulation. Suppose, for example, that we did not impose volume conservation via equation 44, then the geometry would flop, strech, shrink, etc purely in response to external forces. Some degree of volume conservation can be restored in this situation by invoking elastic strain forces, involving pressure and bulk modulus. These do not rigorously guarantee volume conservation, but can be made to conserve volume over some averaging region and period of time. If at some point we wish to allow the volume to not be conserved, it will be useful to bring back these moduli in order to describe the forces that give rise to volume nonconservation.

One quantity defined in continuum mechanics that could prove useful is the pressure, defined as

$$p(\vec{u}) = -\frac{1}{3} \sum_a \epsilon^a_{;a}(\vec{u}) \tag{63}$$

This could prove useful in monitoring the magnitude of deformations.

4 Deformation Dynamics

This section looks at the dynamical evolution of a volumetric geometry that is subjected to external forces. We consider both conservative and frictional forces. Constraints and collisions are not treated here, but in section 5 collisions are introduced and handled.

Equations of Motion

The equations of motion we will handle here are relatively simple: Newton's equation

$$\ddot{\mathbf{x}}(t) = \mathbf{F}\left[\mathbf{x}(t)\right] \tag{64}$$

This equation applies to each point $\mathbf{x}$ of the volume, and in particular to the set of points we will use for our computations. Typical particle-based approaches use a collection of particles placed throughout the volume to describe the shape of the volume. External forces would tear the volume apart ordinarily, so the additional constraining forces must be added in order to preserve the underlying connectedness of the volume[5]. Here, the external force $\mathbf{F}$ does not include constraining forces between points in the volume, because we have connectedness already built in from the geometric description.

We have excluded here the case of damping forces – forces involving $\dot{\mathbf{x}}$ for several reasons:

1. Damping forces are frequently needed simply to handle numerical instabilities, which is a particularly important problem in particle based deformations techniques. Here the underlying geometric description reduces the stability issues considerably, as will be seen below.

2. Dampling is undesired because it is more efficiently accomplished at a later stage of the problem, rather than in the formulation of the approach. The result may be unphysical, but this is movie-making not science.

3. If you *really* need damping at this stage, we can start equation 64 with the force of the form $\mathbf{F}\left[\mathbf{x}(t), \dot{\mathbf{x}}(t)\right]$ and generate a line of calculations similar to the one below.

From Newton's equation, we will derive in the sections below a set of other equations that describe the motion of the geometry. As we will see, these equations devide naturally into describing three types of motion, which we label *body*, *flex*, and *twist* motions. Body motion is just the motion of the center of mass. The center of mass assumes a special role in this approach, partly because its location within the volume changes from moment to moment, but also because we will use it as the point which is defined to have no deformation. All deformations are then constructed relative to that position. With this separation of the motion into three types, the update to the deformation over time becomes

$$\begin{aligned} \mathbf{x}(\vec{u}, t+\delta t) &= \mathbf{x}(\vec{u}, t) + \delta\mathbf{x}(\vec{u}, t) \\ &= \mathbf{x}(\vec{u}, t) + \delta\mathbf{x}_{Body}(t) + \delta\mathbf{x}_{Flex}(\vec{u}, t) + \delta\mathbf{x}_{Twist}(\vec{u}, t) \end{aligned} \tag{65}$$

Body Dynamics

Body dynamics is the motion of the center of mass due to a force applied throughout the entire geometry. This does not include torques applied to the geometry. The position of the body point, $\mathbf{x}_{Body}$ is defined as the center of mass:

$$\mathbf{x}_{Body}(t) = \frac{1}{V}\int d^3u\, \mathbf{x}(\vec{u},t) \tag{66}$$

and undergoes motion due to the volume averaged force

$$\begin{aligned} \ddot{\mathbf{x}}_{Body}(t) &= \frac{1}{V}\int d^3u\, \mathbf{F}\left[\mathbf{x}(\vec{u},t)\right] \\ &= \mathbf{F}_{Body}\left[\mathbf{x}_{Body}(t)\right] \end{aligned} \tag{67}$$

This equation is just the usual dynamics of a single particle. Solve it by the usual techniques.

Flex Dynamics

Flex dynamics includes two types of motion: rigid body rotations and scaling (stretching). We can derive the force for this by Taylor series expansion of the external force around the Body dynamics force, keeping only the first, linear term. The general result looks like:

$$\ddot{\mathbf{x}}_{Flex}(\vec{u},t) = \mathbf{R}(t)\cdot\left(\mathbf{x}_{Flex}(\vec{u},t) - \mathbf{x}_{Body}(t)\right) \tag{68}$$

The matrix $\mathbf{R}$ describes the rotation and stretching that can happen due to the external forces, and is a function of time only. This is a linear dynamics problem, for which there are many numerical and analytical techniques that can be applied.

Twist Dynamics

Twist dynamics is all the other motion caused by the external force that is not handled by the Body and Flex dynamics terms. Since we are dealing with dynamics that change the shape of the geometry, and we are describing the shape of the geometry in terms of the Connection Γ^c_{ab}, this subsection is aimed at generating dynamical equations for the Connection, which is now a function of time as well as material coordinates.

To begin the derivation of the dynamical equations for the Connection, we can take a derivative of Newton's equation 64:

$$\ddot{\mathbf{x}}_a(\vec{u},t) = \mathbf{x}_a\cdot\nabla_{\mathbf{x}}\,\mathbf{F}\left[\mathbf{x}(\vec{u},t)\right] \tag{69}$$

This equation details how the local coordinate system of the geometry changes in time. The equation will be very useful, however it is not the final dynamical equation we seek because the Connection is not involved. Once more, we take another derivative, to obtain:

$$\begin{aligned} \ddot{\mathbf{x}}_{ab}(\vec{u},t) &= \mathbf{x}_{ab}\cdot\nabla_{\mathbf{x}}\,\mathbf{F}\left[\mathbf{x}(\vec{u},t)\right] \\ &+ \left(\mathbf{x}_a\,\mathbf{x}_b\right) : \nabla_{\mathbf{x}}\nabla_{\mathbf{x}}\,\mathbf{F}\left[\mathbf{x}(\vec{u},t)\right] \end{aligned} \tag{70}$$

The dynamics of the Connection is embedded in this equation, because of the fact that the Connection is related to $\mathbf{x}_{ab}$. Further evaluation will simplify the dynamics from this also. Recalling the relationship in equation 8, we can expand the left hand side of equation 70 as

$$\ddot{\mathbf{x}}_{ab}(\vec{u},t) = \sum_c \left(\ddot{\Gamma}^c_{ab}(\vec{u},t)\mathbf{x}_c(\vec{u},t) + 2\dot{\Gamma}^c_{ab}(\vec{u},t)\dot{\mathbf{x}}_c(\vec{u},t) + \Gamma^c_{ab}(\vec{u},t)\ddot{\mathbf{x}}_c(\vec{u},t) \right) \quad (71)$$

Recalling equation 69, the last term in this sum becomes

$$\begin{aligned} \sum_c \Gamma^c_{ab}(\vec{u},t)\ddot{\mathbf{x}}_c(\vec{u},t) &= \sum_c \Gamma^c_{ab}(\vec{u},t)\ \mathbf{x}_c \cdot \nabla_{\mathbf{x}}\ \mathbf{F}\left[\mathbf{x}(\vec{u},t)\right] \\ &= \mathbf{x}_{ab} \cdot \nabla_{\mathbf{x}}\ \mathbf{F}\left[\mathbf{x}(\vec{u},t)\right] \end{aligned} \quad (72)$$

So, by combining equations 70, 71, and 72, we arrive at the dynamical equations

$$\sum_c \left(\ddot{\Gamma}^c_{ab}(\vec{u},t)\mathbf{x}_c(\vec{u},t) + 2\dot{\Gamma}^c_{ab}(\vec{u},t)\dot{\mathbf{x}}_c(\vec{u},t) \right) = (\mathbf{x}_a\ \mathbf{x}_b) : \nabla_{\mathbf{x}}\nabla_{\mathbf{x}}\ \mathbf{F}\left[\mathbf{x}(\vec{u},t)\right] \quad (73)$$

Taking the inner product of this equation with the dual vector $\mathbf{x}^d$, we arrive at the equations for the Connection dynamics

$$\ddot{\Gamma}^d_{ab}(\vec{u},t) + 2\sum_c \dot{\Gamma}^c_{ab}(\vec{u},t)\ \dot{\mathbf{x}}_c(\vec{u},t) \cdot \mathbf{x}^d(\vec{u},t) = \{(\mathbf{x}_a\ \mathbf{x}_b) : \nabla_{\mathbf{x}}\nabla_{\mathbf{x}}\ \mathbf{F}\left[\mathbf{x}(\vec{u},t)\right]\} \cdot \mathbf{x}^d(\vec{u},t) \quad (74)$$

We now see from this result why we chose at the beginning to separate out the Body and Flex dynamics from this Twist dynamics. The forces that give rise to Body and Flex motion do not appear in the Twist force, because of the two gradients on the force that appear in equation 74. Also, we see how the geometric description provides its own effective damping behavior in the second term on the left hand side. In fact, there is an analytical solution we can apply to this. Defining the matrix $\mathbf{W}$ by

$$\left(\mathbf{W}\right)_{ab}(\vec{u},t) = 2\dot{\mathbf{x}}_b(\vec{u},t) \cdot \mathbf{x}^a(\vec{u},t) \quad (75)$$

equation 74 can be integrated once to produce an equation for $\dot{\Gamma}$. Here we will do that for a short time δt, to give

$$\begin{aligned} \delta\Gamma^c_{ab}(\vec{u},t+\delta t) &= \delta t\ \dot{\Gamma}^c_{ab}(\vec{u},t+\delta t) \\ &= \sum_d \left(\exp\left\{-\delta t\ \mathbf{W}(\vec{u},t)\right\}\right)_{cd}\ \delta\Gamma^d_{ab}(\vec{u},t) \\ &+ (\delta t)^2\ \left\{(\mathbf{x}_a(\vec{u},t)\ \mathbf{x}_b(\vec{u},t)) : \nabla_{\mathbf{x}}\nabla_{\mathbf{x}}\ \mathbf{F}\left[\mathbf{x}(\vec{u},t)\right]\right\} \cdot \mathbf{x}^c(\vec{u},t) \end{aligned} \quad (76)$$

This last result is essentially the full solution to the dynamics problem for unconstrained motion in response to external forces.

Volume Conservation Issues

At this juncture we have two conditions on the Connection as it deforms in response to forces: the volume conservation constraint represented equivalently by equations **??**

and 44, and the dynamical equations of motion in equation 74. How do we merge the two so that the dynamics will always conserve volume?

Lets start with an iterative approach. Suppose $\delta\Gamma^d_{ab}(\vec{u},t)$ is antisymmetric (and therefore conserves volume). To what extent does equation 76 leave $\delta\Gamma^d_{ab}(\vec{u},t+\delta t)$ antisymmetric as well?

As far as the first term on the right hand side of equation 76, it is worthwhile noting that $\mathbf{W}$ is antisymmetric. This is because $\mathbf{x}_a \cdot \mathbf{x}^b = \delta^b_a$, so $\dot{\mathbf{x}}_a \cdot \mathbf{x}^b = -\mathbf{x}_a \cdot \dot{\mathbf{x}}^b$. So, in a recurring theme throughout these notes, $\exp(-\delta t\mathbf{W})$ is an $SO(3)$ matrix. However, the product of an $SO(3)$ matrix and an antisymmetric matrix in general is *not* antisymmetric.

The second term on the right hand side is antisymmetric or not depending on the nature of the external force used in the dynamics. So in general we can't count on antisymmetry.

Overall, it appears that volume conservation requires that we explicity antisymmetrize the results of the dynamics in equation 76. From this point on we will designate this antisymmetric part of $\delta\Gamma^c_{ab}$ as $(\delta\Gamma^c_{ab})^-$. There is an ambiguity of how we go about doing this however. We could generate $(\delta\Gamma^c_{ab}(\vec{u},t))^-$, feed that into equation 76, and antisymmetrize $(\delta\Gamma^c_{ab}(\vec{u},t+\delta t))^-$, or use the full outcome for $\delta\Gamma^c_{ab}(\vec{u},t)$ in equation 76 and antisymmetrize the result from that. The results of the two approaches are different. At the moment, there does not appear to be a consistent guideline for choosing one over the other. But the issue is more complex, because volume conservation must take into account the affects Body and Flex motions. Body motions will not violate volume conservation, but Flex motion in general will. In the next section, applying collision constraints will also raise issues for volume conservation. While it might seem of little value to antisymmetrize the Twist motion at this stage when there are more steps needed to conserve volume, it is a good idea to apply antisymmetrization here regardless of what other manipulations to the volume do. If not, volume-changing Twist motions could interact in a nonlinear way with other effects to produce unreasonable behaviors, even though the overall motion conserves volume. So for now, the choice we will make is to antisymmetrize the twist motion, as well as taking additional steps later to conserve volume after other forces/collisions/constraints have been applied.

Stability

Before launching into a discussion of numerical implementation steps, we can first examine the stability properties of the dynamics solution contained in equation 76. We assume that there are no boundaries or collision considerations here. The simplest way to look at stability is to imagine that some external force has driven the geometry into some appropriate motion, and then at time t_0 the force is instantaneously turned off. The volume is then in a "weightless" environment undergoing some residual motion left over from the now gone force. In the case of Body and Flex motion, the geometry is left translating at a constant velocity and rigidly rotating at a constant angular velocity.

For the Twist motion, the equations of motion are now

$$\delta\Gamma^c_{ab}(\vec{u}, t+\delta t) = \sum_d \left(\exp\left\{-\delta t\ \mathbf{W}(\vec{u}, t)\right\}\right)_{cd}\ \delta\Gamma^d_{ab}(\vec{u}, t) \tag{77}$$

for $t > t_0$. Since $\mathbf{W}$ is antisymmetric, $\exp(-\delta t\ \mathbf{W})$ is just a rotation of the change in the connection. In particular, its magnitude is conserved:

$$\sum_c \left(\delta\Gamma^c_{ab}(\vec{u}, t+\delta t)\right)^2 = \sum_c \left(\delta\Gamma^c_{ab}(\vec{u}, t)\right)^2 \tag{78}$$

So the twist motion is just a continual cycling through what could be a complex series of shape changes, for example like the very organic blobby motion of a glob of fluid suspended in zero gravity. But the solution is intrinsically stable over time. A useful test of any numerical implementation would be to produce this behavior over a long sequence of time steps.

Computational Scheme

Here we discuss some aspects of a computational scheme to implement the deformation dynamics discussed in this section. We focus mostly on the chain of steps needed in the computation, and not at all on the architecture of the software. We assume there are no collisions with other objects. Collisions will be discussed in the next section (5).

As we have found in this section, the dynamics is handled reasonably well by an explicit finite difference scheme, generating a new state for the geometric system at time $t + \delta t$ from the state at time t. Our goals here are to

1. Identify the elements of the geometric state,
2. Step through the computations required to update the state.

Geometric State

The elements of the geometric state include the following items at each material coordinate:

1. The value $\vec{u}$ of the material coordinate.
2. The position $\mathbf{x}(\vec{u})$ of the the point in space.
3. The tangent space basis $\{\mathbf{x}_a(\vec{u}),\ a = 1, 2, 3\}$
4. The metric tensor $g_{ab}(\vec{u})$ and its inverse $g^{ab}(\vec{u})$.
5. The dual basis $\{\mathbf{x}^a(\vec{u}),\ a = 1, 2, 3\}$, ($\mathbf{x}^a = \sum_b g^{ab}\mathbf{x}_b$).
6. The Connection $\Gamma^a_{bc}(\vec{u})$.
7. The Connection change $\delta\Gamma^a_{bc}(\vec{u})$ from the last time step (after antisymmetrization).

8. The basis derivative $\dot{\mathbf{x}}_a(\vec{u})$ from the previous time step.

This is the full list of items needed to handle the dynamics of the volume. The list has some repetitive information. For example, item 5 can be computed from items 3 and 4. However, we include it explicitly to highlight the fact that it will be needed.

Computation Steps

The set of steps listed here ignore the issue of collisions with over objects. Collisions are treated in the next section, at which point this list will be altered appropriately.

1. Compute the deformations due to the Flex and Body dynamics.
   (a) Initialize the deformation and new basis:
   $$\mathbf{X} = \mathbf{x} + \delta\mathbf{x}_{Flex} + \delta\mathbf{x}_{Body} \quad (79)$$
   $$\mathbf{X}_a = \mathbf{x}_a + \frac{\partial}{\partial u_a}\delta\mathbf{x}_{Flex} \quad (80)$$
   (b) Test for volume nonconservation at each material coordinate by comparing the old and new values of the determinant of the metric.
   (c) Where volume has not been conserved:
      i. Compute $\mathbf{M}$ at that point.
      ii. Project $\mathbf{M}$ to just its $SL(3)$ component (more on that later).
      iii. Using neighboring material coordinates as a very short path, compute a new value for ϵ^a at this material coordinate.
      iv. Replace $\mathbf{X}$ with the new value, relabel $\mathbf{x}(\vec{u}) \rightarrow \mathbf{X}(\vec{u})$.
   (d) Recompute the remaining elements of the geometric state: $\mathbf{x}_a$, g_{ab}, Γ^c_{ab}, etc.
2. Compute deformations due to Twist dynamics
   (a) Update the Connection changes $\delta\Gamma^a_{bc}(\vec{u})$ at all material coordinates $\vec{u}$.
      i. Compute $\mathbf{W}(\vec{u})$ and exponentiate it for $\exp(\delta t\ \mathbf{W})$.
      ii. Compute the external force $(\delta t)^2\ \{(\mathbf{x}_a\ \mathbf{x}_b) : \nabla_{\mathbf{x}}\nabla_{\mathbf{x}}\ \mathbf{F}\} \cdot \mathbf{x}^c$
      iii. Update $\delta\Gamma^c_{ab}$ using equation 76, and antisymmetrize it.
   (b) Establish a collection of interlinked paths throughout the volume.
   (c) Choose a path that connects a reference point to the point of interest.
   (d) As the path is traversed, use Γ and $\overline{\Gamma} = \Gamma + \delta\Gamma$ to compute a running evaluation of $\mathbf{H}$, $\overline{\mathbf{H}}$, $\mathbf{M}$, and ϵ^a at each material coordinate along the path using equations 57 and 59.
   (e) Compute the new basis vectors and metric, using $\mathbf{M}$ at each material coordinate and equation 46.
   (f) Update the remaining members of the geometric state.

In step 1(c)ii, Flex dynamics sometimes produces a matrix $\mathbf{M}$ that is not an element of $SL(3)$. In this situation, we need to project the matrix to an $SL(3)$ member in order to conserve volume. We can accomplish that through the simple step of rescaling the elements of $\mathbf{M}$. Suppose $\det(\mathbf{M}) \neq 1$, then the rescaling

$$\mathbf{M} \rightarrow \mathbf{M}/\det(\mathbf{M})^{1/3} \tag{81}$$

gives the matrix a unit determinant as needed. However, for large deformations this scaling can greatly distort the solution. The alternative approach that is better behaved is to rescale the deformation components directly, i.e. introduce a scaling factor f and the transformation

$$\mathbf{M} \rightarrow \mathbf{1} + f(\mathbf{M} - \mathbf{1}) \tag{82}$$

and fixed the value of f by the determinant constraint. Although superficially the polynomial equation for f is cubic, one of the solutions is trivially $f = 0$. Excluding that one, the polynomial equations for the remain roots is quadratic. If we denote the two roots as $f_{\pm}$, the best one to choose is the one with the minimum absolute value.

5 Collisions

The last major problem to deal with is collisions between deformable geometric volumes and other volumes and/or boundaries. This problem and its solutions are divided into three steps:

1. Detecting a collision between two extended objects,
2. Correcting the positions of the boundary vertices in accordance with the collision conditions,
3. Recomputing the shape of the geometry, keeping volume conservation enforced.

The typical approach to collision detection for cloth dynamics is based on the underlying particle description used: test individual particles for collision events[8]. Our description in terms of geometry would be best served by a geometrically-oriented approach to collision detection and adjustment. There is in fact just such a technique, based on a quantity called the *Gauss Linking Number* (GLN). Since the GLN figures prominently in this technique, we begin by introducing it, discuss how it is useful in collision detection, and how to efficiently compute it. That is followed by a discussion on GLN networks, which are collections of closed curves on the surface of the volume which act as the measurement sensors for detecting and isolating a collision. We finish this section with a discussion of how to use the collision detection/isolation to correct respond to the collision and recompute the shape of the deformable geometry while still conserving volume.

Gauss Linking Number

We introduce here a very interesting and unique quantity called the Gauss Linking Number (GLN) [2]. This quantity is related to the topological characteristics of closed

loop curves in space. More specifically, it is equal to the number of times two curves link to each other. For two closed C^1 loops, characterized by the functions $\vec{\gamma}_i(t),\ i = 1, 2$ with $0 \leq t \leq 1$ and $\vec{\gamma}_i(0) = \vec{\gamma}_i(1)$ in space, the Gauss Linking Number can be computed from the integral

$$GLN(\vec{\gamma}_1, \vec{\gamma}_2) = \frac{1}{4\pi} \int_0^1 dt_1 \int_0^1 dt_2 \ \left(\frac{d\vec{\gamma}_1(t_1)}{dt_1} \times \frac{d\vec{\gamma}_1(t_2)}{dt_2} \right) \cdot \frac{(\vec{\gamma}_1(t_1) - \vec{\gamma}_2(t_2))}{|\vec{\gamma}_1(t_1) - \vec{\gamma}_2(t_2)|^3} \tag{83}$$

Incredibly, this quantity is completely independent of the shapes of the two curves, and is an integer, equal to the number of times the two curves link. So if the two curves $\vec{\gamma}_1$ and $\vec{\gamma}_2$ can be separated from each other by any continuous deformation of their shapes, then $GLN = 0$. If they are linked together once like two chain links, then $GLN = 1$, and so on for more complex linkages. So, despite the fact that GLN is evaluated from an integral expression, in principle it can act as a boolean test for whether two curves are linked.

The utility of the Gauss Linking number comes from picking one of the curves to lie on one surface geometry, while the other curve lies on some other surface geometry. If the two surfaces do not intersect within the area enclosed by the two curves, then $GLN = 0$. But if the two surfaces intersect to the extent that the curves cross each other, then computing GLN will detect that intersection. To be able to localize the intersection on the surface, a network of curves on each surface can be layed out, and localization happens by examining the collection of curves on each surface that link according to the value of GLN.

Numerical Evaluation of GLN

The utility of the Gauss Linking Number to collision detection depends on being able to evaluate the integrals in equation 83 rapidly. Here we make an attempt to reduce the complexity of the effort and achieve a fast evaluation.

The numerical evaluation of this quantity is suprisingly easy for piecewise linear curves. If we take the two curves to be composed of linear segments, so that

$$\vec{\gamma}_1(s) = \vec{\gamma}_1^k + \frac{s - s_k}{s_{k+1} - s_k}(\vec{\gamma}_1^{k+1} - \vec{\gamma}_1^k) \quad s_k \ \leq s \ \leq s_{k+1} \ \ k = 0, \ldots N \tag{84}$$

$$\vec{\gamma}_2(r) = \vec{\gamma}_2^k + \frac{r - r_k}{r_{k+1} - r_k}(\vec{\gamma}_2^{k+1} - \vec{\gamma}_2^k) \quad r_k \ \leq r \ \leq r_{k+1} \ \ k = 0, \ldots M \tag{85}$$

$$\tag{86}$$

and because these are closed loops, we have that $s_{N+1} = s_0$, $\vec{\gamma}_1^{N+1} = \vec{\gamma}_1^0$, $r_{M+1} = r_0$, and $\vec{\gamma}_2^{M+1} = \vec{\gamma}_2^0$. With this form, the Gauss Linking Number is

$$GLN(\vec{\gamma}_1, \vec{\gamma}_2) = \frac{1}{4\pi} \sum_{k=0}^{N} \sum_{\ell=0}^{M} \left\{ \left(\vec{\gamma}_1^{k+1} - \vec{\gamma}_1^k\right) \times \left(\vec{\gamma}_2^{\ell+1} - \vec{\gamma}_2^\ell\right) \right\} \cdot \left(\vec{\gamma}_1^k - \vec{\gamma}_2^\ell\right) \ G_{k\ell} \,, \tag{87}$$

with the coefficients

$$G_{k\ell} = \int_0^1 ds \int_0^1 dr \left| \vec{\gamma}_1^k - \vec{\gamma}_2^\ell + s(\vec{\gamma}_1^{k+1} - \vec{\gamma}_1^k) + r(\vec{\gamma}_2^{\ell+1} - \vec{\gamma}_2^\ell) \right|^{-3} \tag{88}$$

Once again, we are very fortunate that at least one of the integrals in this expression can be evaluated exactly [11]using the formula:

$$\begin{aligned}\int_0^1 dt \left(a+bt+ct^2\right)^{-3/2} &= \frac{2(2c+b)}{4ac-b^2}\frac{1}{\sqrt{a+b+c}} \\ &- \frac{2b}{4ac-b^2}\frac{1}{\sqrt{a}} \end{aligned} \tag{89}$$

The remaining integral can be evaluated by brute force without any serious problems. In fact, code which accomplishes its evaluation, can be written in a Maya plugin called GaussLinkingNumber as:

```
double
GaussLinkingNumber::gln_segmented_compute( MFnNurbsCurve& curve1 ,
                                           MFnNurbsCurve& curve2 )
{
//============================================================
//   Segment evaluation  method - divide into multiple straight
//   line segments and evaluate integral exactly on one curve
//   and numerically on the other.
//============================================================
double p1sampling = 50, p2sampling = 50;
double gln = 0;
double p1start = 0, p1end = 0;
double p2start = 0, p2end = 0;
curve1.getKnotDomain(p1start, p1end);
curve2.getKnotDomain(p2start, p2end);
double dp1 = tolerance * (p1end - p1start) / p1sampling;
double dp2 = tolerance * (p2end - p2start) / p2sampling;
MPoint v1minus, v1plus;
MPoint v2minus, v2plus;
MVector dv12mm, dv11pm, dv22pm;
curve1.getPointAtParam ( p1start, v1minus, MSpace::kWorld );
curve2.getPointAtParam ( p2start, v2minus, MSpace::kWorld );
for(double p1=p1start+dp1; p1 <= p1end; p1 += dp1)           //Loop over first
{                                                           //  curve segment
   curve1.getPointAtParam ( p1, v1plus, MSpace::kWorld );   //
   dv11pm = v1plus - v1minus;                               //
   for(double p2=p2start+dp2; p2 <= p2end; p2 += dp2)       //Loop over second
   {                                                        //  curve segment
     curve2.getPointAtParam ( p2, v2plus, MSpace::kWorld ); //
     dv22pm = v2plus - v2minus;                             //
     dv12mm = v1minus - v2minus;                            //
     double c = dv22pm * dv22pm;                            //
     double ds = 0.02 * tolerance;                          //
     double accum = 0;                                      //
     for(double s = 0; s <= 1; s += ds)                     //Evaluate integral
     {                                                      //  for G_{kl}
       MVector va = dv12mm + s * dv11pm;                    //
       double a = va * va;                                  //
       double b = -2.0 * va * dv22pm;                       //
       accum += ((2*c+b)/sqrt(a+b+c)-b/sqrt(a))*2/(4*a*c-b*b);//
     }                                                      //
     accum *= ds;                                           //accum = G_{kl}
     gln += dv12mm * ( dv11pm ^ dv22pm ) * accum;           //Increment GLN
     v2minus = v2plus;                                      //
```

```
   }
   v1minus = v1plus;
}
gln /= (4.0 * 3.14159265);
return gln;
}
```

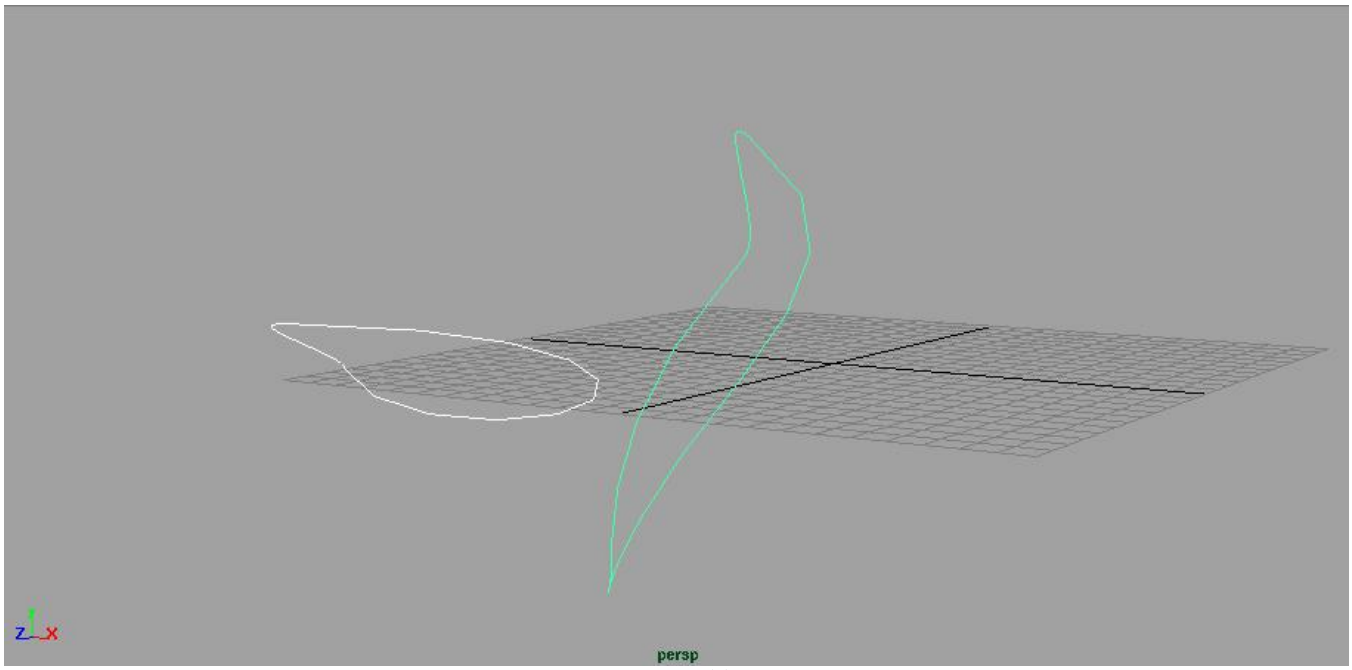
The result is sensitive to the sample spacing in the integral, as illustrated by the example in figures 1 through 4. Figure 1 shows the two curves used in the computations, at their initial and final positions in an animation. During 100 frames, one of the curves remains fixed while the second moves closer, intersecting the other at frame 68. Neither curve rotates or changes shape during the animation. The minimum distance, or distance of closest approach, between the two curves is plotted as a function of frame number in figure 2. After intersection at frame 68, the two curves remain linked even as the closest approach distance increases.

The parameter `tolerance` is used to control the sampling quality of both the curves and the numerical integral evaluation. With a default value of 1, larger values of `tolerance` correspond to coarser sampling, and small values are finer sampling.

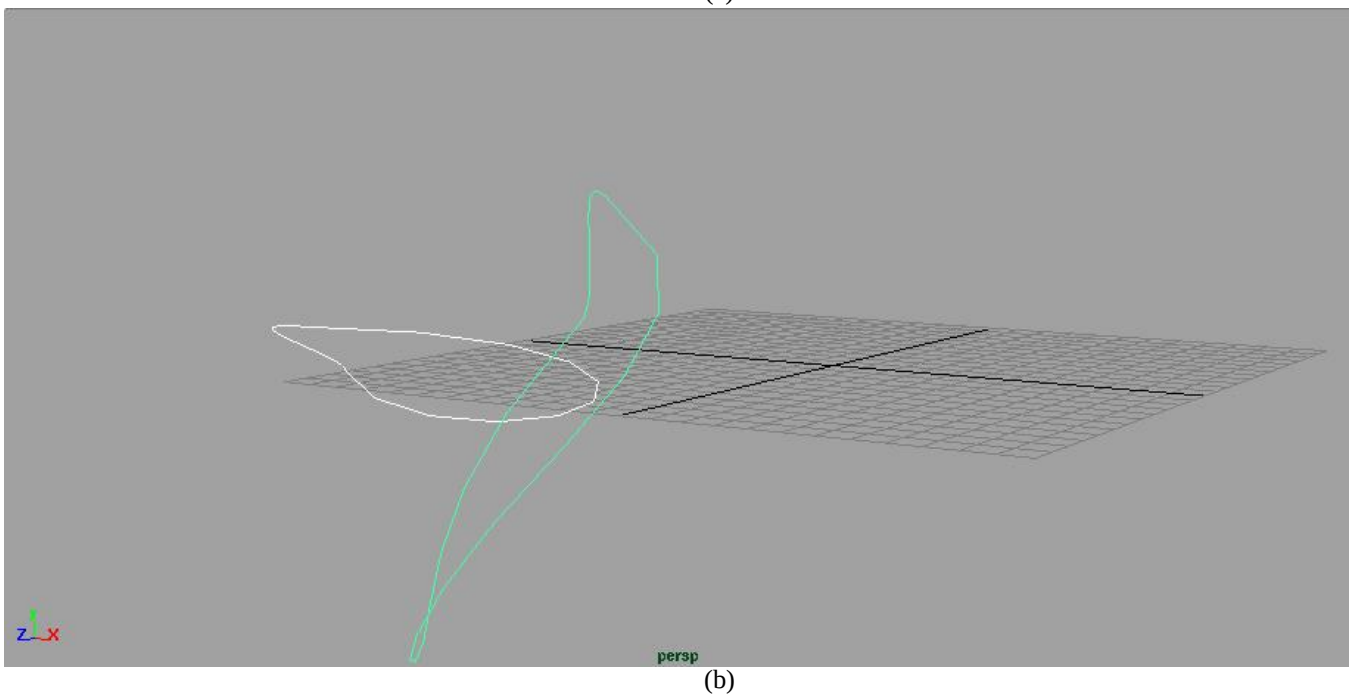
For this example, when the sampling is low (e.g. `tolerance` ~ 10), the value of GLN does not immediately snap from 0 to 1 when intersection occurs (see figure 3). There is a precursor rise in its value its value as the two curves come closer to each other, and a lag in reaching the full value after the curves interlink. Well after interlinking, there is still considerable fluctuation in the value of GLN. Qualitatively, this happens because low sampling in the integral is not able to resolve the shape of the curves adequately. This could be useful in the collision detection problem as a way to adaptively evaluate GLN depending on how great the risk of collision is. For example, if a coarse evaluation of GLN yields a very low value for GLN, then a collision is unlikely and we move on. If the coarse evaluation yields a value significantly above 0, then it may be necessary for a slower, more careful evaluation to make the call for or against collision. This approach lets us use fast approximate evaluations, while still being able to do more thorough evaluations if the answer was not sufficiently compelling. For smaller values of the `tolerance` (≤ 1), the computed GLN behaves very much like the step function expected of the exact situation. These results for GLN are also plotted in figure 4 as a function of the separation distance of the two curves.

GLN Collision Networks

The GLN has value as a collision detector because it has a discrete integer-valued function regardless of the shape of the two curves in the integral: it responds only to the condition of whether the two curves are linked. When we try to go further, and isolate the location of the collision, this topological-invariant property of GLN proves to be a double-edged sword. It is always possible to take a curve on the surface of a volume, and redraw it across the surface in almost unlimited ways without changing the value of the GLN. So when the GLN is not zero, we know that there is a collision between two objects, but we have very little information about where on the surface of either object the collision occurs. This subsection outlines a process for building networks of surface curves, for which the evaluation of GLN for each curve localizes



(a)



(b)

Figure 1: Two curves used in the example Gauss Linking Number calculations. (a) Positions at frame 1, well separated from each other; (b) Positions at frame 100, fully linked.

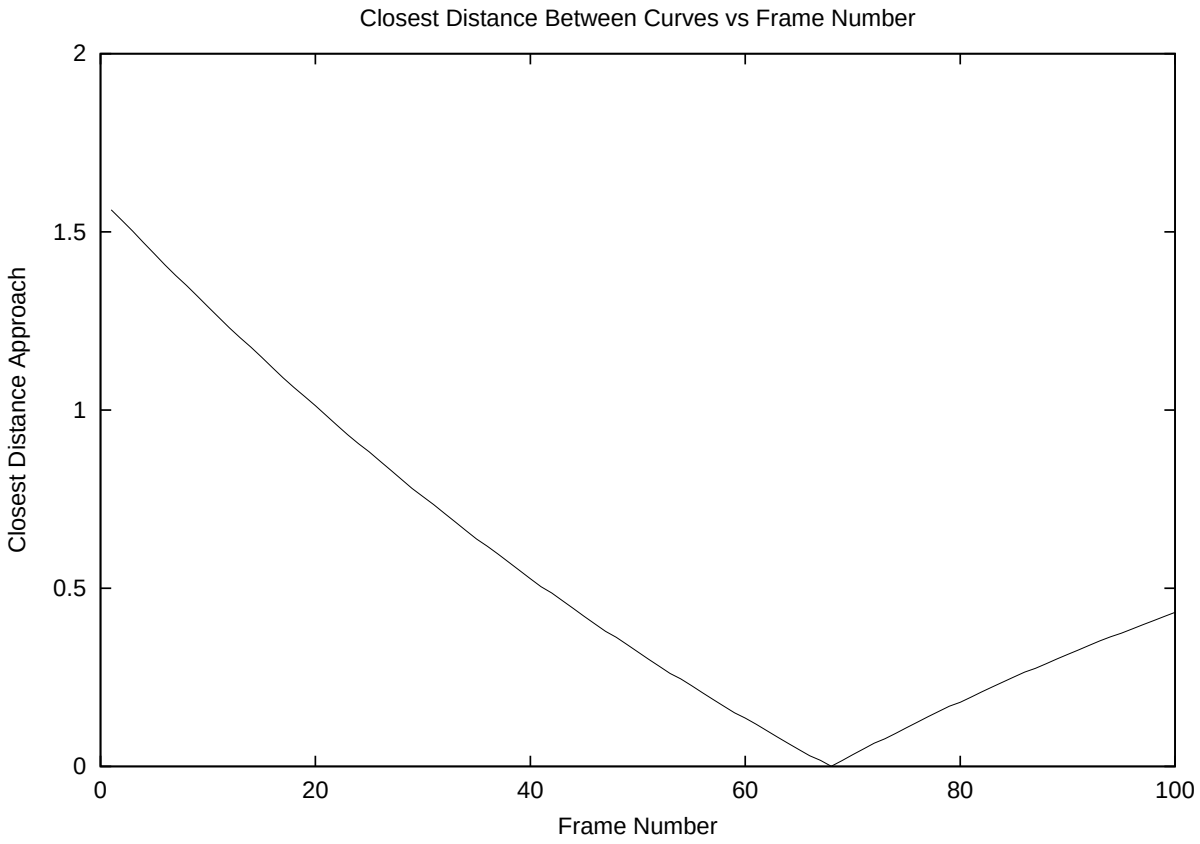


Figure 2: Distance of Closest Approach as a function of the frame number.

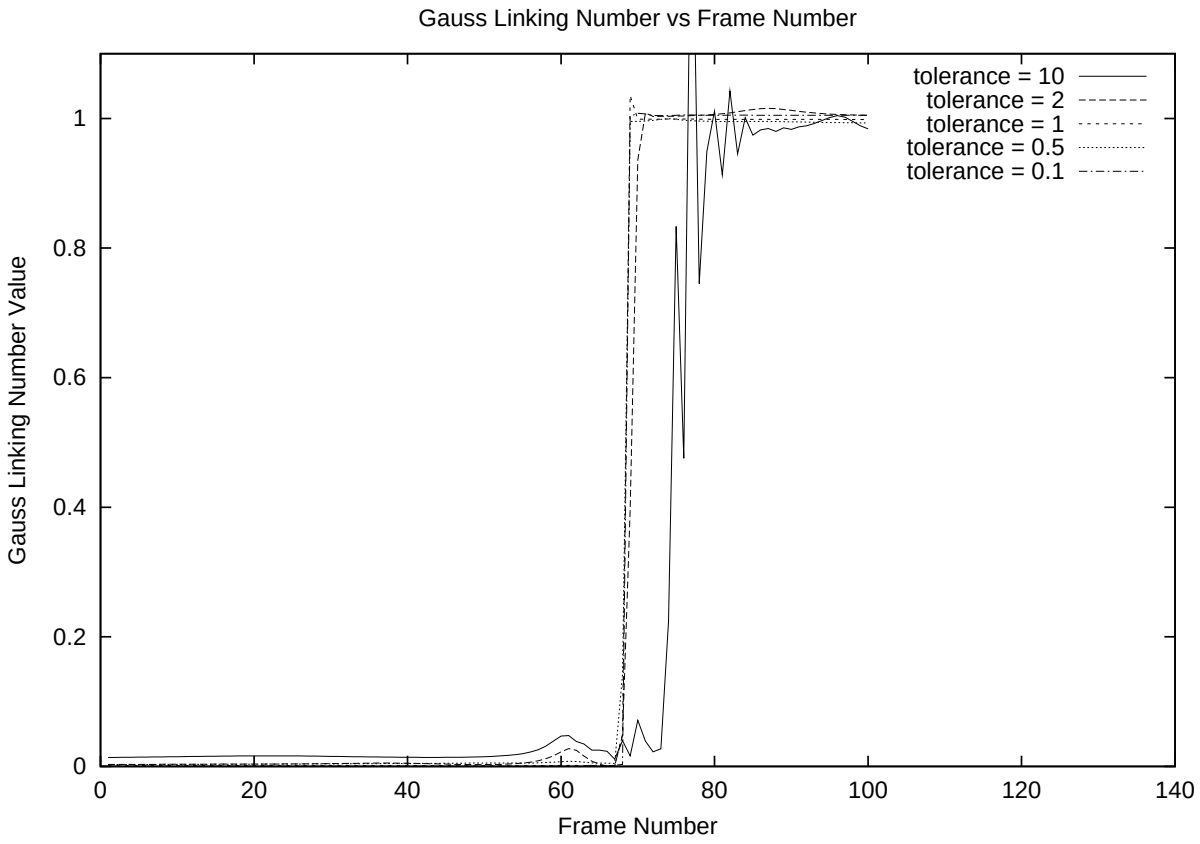


Figure 3: Computed Gauss Linking Number as a function of the frame number.

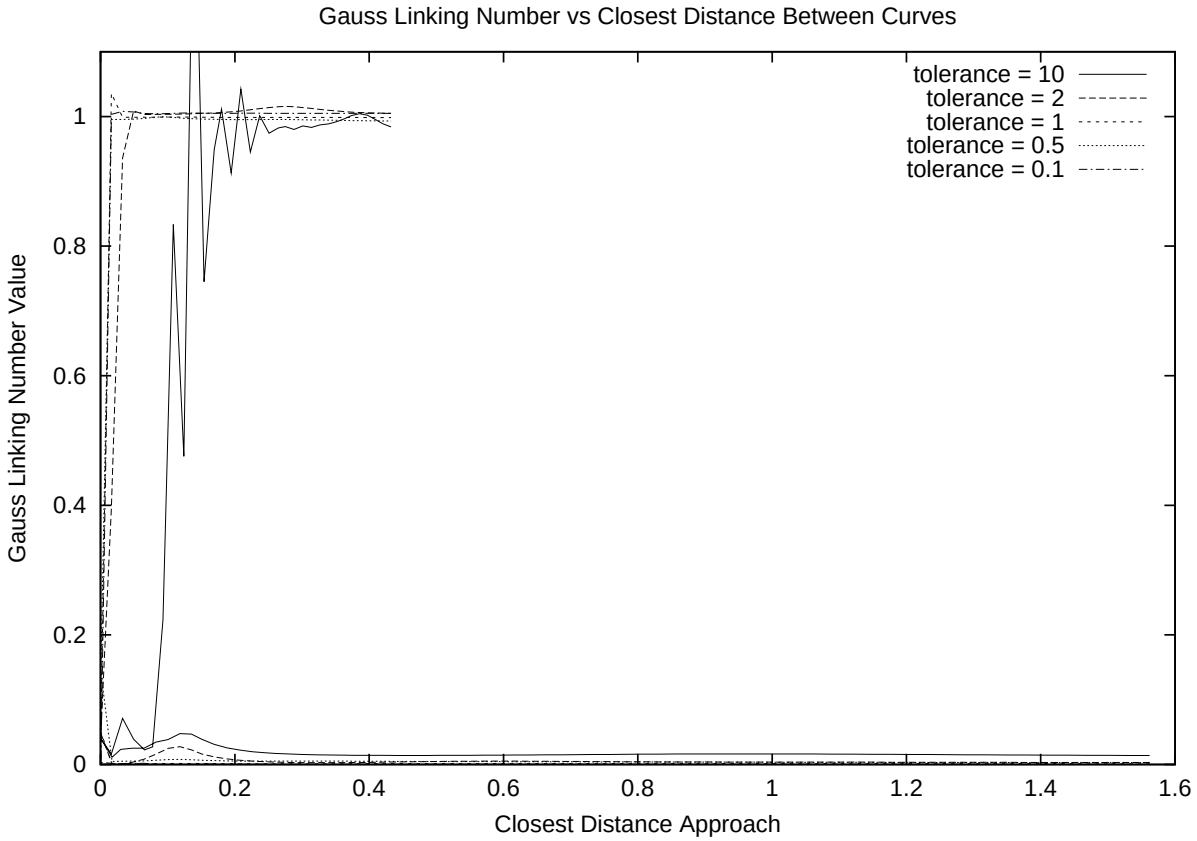


Figure 4: Computed Gauss Linking Number as a function of the distance of closest approach.

the collision on the surface. We begin by defining a *GLN network* as a collection of closed curves on the surface of a volume. The curves typically will overlap each other and redundantly cover areas of interest of the surface of the volume that may be subject to collision. The closed curves are labeled $\vec{\gamma}_i,\ i = 1, \ldots N$. The interior of each curve, denoted $Interior(\vec{\gamma}_i)$, is the set of points on the surface of the volume that are inside the curve and are vertices of the curve. We suppose that a similar network of closed curves has been built for the object with which our volume is possibly colliding, and denote those curves by $\vec{\gamma}_B$. The goal of the collision networks is to find the set of points R on the surface of the volume that are presently intersecting with the collision object. Initializing this set $R = \oslash$, we iterate over all curves on both surfaces, updating R for each curve pair $(\vec{\gamma}_B, \vec{\gamma}_i)$ using the following tests and procedures:

$$GLN(\vec{\gamma}_B, \vec{\gamma}_i) \geq 1 \quad \longrightarrow \quad R := R \cup Interior(\vec{\gamma}_i) \tag{90}$$

$$GLN(\vec{\gamma}_B, \vec{\gamma}_i) = 0 \quad \longrightarrow \quad R := R - R \cap Interior(\vec{\gamma}_i) \tag{91}$$

The points contained in the set R after this process have collided with the collision geometry, and must be repositioned to no longer collide. Repositioning depends on the type of surface interaction that is dictated by the types of the materials the two objects are conceptually composed of. If one or both of the objects are sticky, then repositioning should place the points exactly on the surface of the collision object. Perfectly elastic collisions would position the boundary points according to normal reflection formulae. The logic behind repositioning is the same as for collision of particles with objects[9].

Clearly, the set of points found in R depends are the layout of the collision networks. It is possible for the network to be very coarse and either miss a collision, or include more points in the collision set R than actually are involved. And if there are gaps in the coverage of the surface by the network, points actually involved with a collision may be overlooked. But the approach is very general: any number of closed curves can be used, distributed on the surface of the volume in any way desired, including multiple overlapping regions. So in principle the resolution of the set of collision points R can be very good if the strategy for laying out the collision network fits well with the problem. Success or failure depends entirely on this strategy of picking the network member curves.

Deformation from Collisions

- Use the newly positioned collision points as the reference points for integration of the holonomies $\mathbf{H}$ and the matrix $\mathbf{M}$.
- Recompute Connection change $\delta\Gamma$ and antisymmetrize it.

References

[1] Richard S. Millman and George D. Parker, *Elements of Differential Geometry*, Prentice-Hall Inc., 1977.

[2] Rudolfa Gambini and Jorge Pullin, *Loops, Knots, Gauge Theories and Quantum Gravity*, Cambridge Monographs on Mathematical Physics, Cambridge University Press, 2000.

[3] Sean Carroll, "Lecture Notes on General Relativity," gr-qc/9712019, 1997.

[4] Andrew Witkin, Michael Gleicher, and William Welch, "Interactive Dynamics".

[5] Murilo Coutinho, *Dynamic Simulations of Multibody Systems*, Springer, 2001.

[6] Herbert Goldstein, *Classical Mechanics*, Addison-Wesley, 1950.

[7] Alexander L. Fetter and John Dirk Walecka, Theoretical Mechanics of Particles and Continua, McGraw-Hill, 1980.

[8] Donald H. House, Richard W. DeVaul, and David E. Breen, "Towards Simulating Cloth Dynamics Using Interacting Particles," 1996.

[9] Andrew Witkin, "Physically Based Modeling: Principles and Practice / *Constrained Dynamics*" 1997.

[10] David Baraff and Andrew Witkin, "Large Steps in Cloth Simulation," Computer Graphics Proceedings, Siggraph, 1998.

[11] *CRC Standard Mathematical Tables 25th Edition*, William H. Beyer, Ph.D., ed. Integral #239.

Revision History

```
$Id: paper.tex,v 1.26 2002/01/08 00:55:44 jerry Exp jerry $

$Log: paper.tex,v $
Revision 1.26  2002/01/08 00:55:44  jerry
grammer correction

Revision 1.25  2001/11/09 19:56:43  jerry
typo correction

Revision 1.24  2001/11/09 19:00:25  jerry
Added an entire section on the Jacobi, SOR and CG methods to
solve covariant differential equations.

Revision 1.23  2001/09/18 16:25:57  jerry
small changes and cleaning

Revision 1.22  2001/09/05 18:17:44  jerry
more detail on collisions

Revision 1.21  2001/08/29 16:43:40  jerry
update before long weekend break

Revision 1.20  2001/08/28 17:25:10  jerry
more on GLN and collisions

Revision 1.19  2001/08/27 19:02:10  jerry
more analysis of GLN

Revision 1.18  2001/08/21 13:12:41  jerry
relationship to continuum mechanics; computational steps

Revision 1.17  2001/08/10 18:01:24  jerry
Yet more typos

Revision 1.16  2001/08/10 15:57:35  jerry
typo fix

Revision 1.15  2001/08/09 14:44:34  jerry
typos

Revision 1.14  2001/08/09 14:42:56  jerry
change of heart over volume conservation

Revision 1.13  2001/08/09 14:37:52  jerry
all but collisions done

Revision 1.12  2001/08/08 11:33:37  jerry
finished dynamics discussion - except numerical implementation

Revision 1.11  2001/08/07 16:35:02  jerry
lots of dynamics material, more collision material

Revision 1.10  2001/08/03 17:36:19  jerry
details about gauss linking number

Revision 1.9  2001/08/01 17:47:06  jerry
began work on dynamics section

Revision 1.8  2001/08/01 15:52:48  jerry
improved references list

Revision 1.7  2001/07/31 12:50:22  jerry
nearly complete section on geometry deformation

Revision 1.6  2001/07/27 19:06:03  jerry
incremental addition to section 3

Revision 1.5  2001/07/27 17:31:50  jerry
put revision history into body of document

Revision 1.4  2001/07/27 17:27:53  jerry
Lots more material, including most of Geometric Reconstruction

Revision 1.3  2001/07/23 19:04:32  jerry
more material on covariant derivatives and solutions

Revision 1.2  2001/07/13 17:50:15  jerry
initial tracking
```