Discrete bending forces and their Jacobians

Rasmus Tamstorf[a,*], Eitan Grinspun[b],

[a] *Walt Disney Animation Studios, Burbank, CA 91521, USA*
[b] *Columbia University, New York, NY, USA*

Abstract

Computation of bending forces on triangle meshes is required for numerous simulation and geometry processing applications. A common quantity in many bending models is the hinge angle between two adjacent triangles. This angle is straightforward to compute, and its gradient with respect to vertex positions (required for the forces) is easily found in the literature. However, its Hessian, which is required for efficient numerics (e.g., implicit time stepping, Newton-based energy minimization) is not documented in the literature. Readily available computations of the Hessian, such as those produced by symbolic algebra systems, or by autodifferentiation codes, are expensive to compute. We present compact, easily reproducible, closed form expressions for the Hessian. Compared to the automatic differentiation, we measure up to 7× speedup for the evaluation of the bending forces and their gradients.

Keywords: Discrete shells, Hinge angle Hessian, Bending force Jacobians

1. Introduction

Important problems in computer simulation, animation, and geometry processing, involve the formulation of an energy in terms of the *hinge angle* between pairs of adjacent mesh triangles (see Figure 1). Examples include the

- wrinkling energy of a worn garment [3, 8],
- elastic energy of a Kirchhoff-Love thin-shell [12],
- deformation energy for example-driven deformations [9]
- Willmore energy used in mesh smoothing [17], and
- dissipative potential of viscous liquid sheets [4].

Efficient numerical treatments of the associated variational problems (e.g., via Newton's method) or partial differential equations (e.g., via implicit time stepping [3]) necessitate a formulation not only of the energy and its gradient, but also of the *Hessian* of the energy with respect to mesh position.

In our own experience, and over years of interacting with researchers and practitioners working on myriad applications, we have found that these Hessians are exceedingly tedious to derive by hand, with compact formulations sometimes consuming weeks of manual derivation. This process is error prone, often leading to analytic expressions that disagree with numerical validation. The process can be suboptimal, missing opportunities for gathering like terms, thus leading to longer source code and more expensive computation. These liabilities are detrimental to the adoption of efficient numerical methods for hinge-based energies, as evidenced in the literature:

- Bridson et al. [8] avoided Hessians by treating bending forces explicitly; similarly Fröhlich and Botsch [9] avoided Hessians by using Gauss-Newton's method;
- Baraff and Witkin [3] introduced approximating assumptions (e.g, inextensible cloth, undergoing only small deformations, with flat rest shape) treating normals and edge lengths as constants;
- Bergou et al. [6], Wardetzky et al. [17] derived a simplified Hessian formula for the special case energy $\sin^2(\theta/2)$, using a technique that does not accommodate the general case;

*Corresponding author.
Email addresses: rasmus.tamstorf@disney.com (Rasmus Tamstorf), eitan@cs.columbia.edu (Eitan Grinspun)

- Grinspun et al. [12] computed the Hessian using automatic differentiation, which dominated the computational cost of the method.

Contributions. In light of these observations, this paper seeks to facilitate adoption, code legibility, and performance efficiency of hinge-based bending energies.

- We present a compact and efficient formulation of the Hessian for the general case of a hinge-based bending energy.
- By taking advantage of several symmetries in the expressions (some less obvious than others), we observe that many terms can be reused when assembling the Hessian for an entire mesh, further reducing the cost of computation.
- We present the results of experiments documenting up to 7× speedup of the formulation compared to autodifferentiation and up to 4× speedup compared to an existing (but unpublished) symbolic derivation.

2. Bending energy

Notation. Figure 1 presents the labels and indices for a single *hinge stencil*, consisting of four vertices $\mathbf{x}_i$, five edges $\mathbf{e}_i$ and $\tilde{\mathbf{e}}_i$, two normals $\mathbf{n}$ and $\tilde{\mathbf{n}}$, bend angle θ, interior angles α_i and $\tilde{\alpha}_i$, and heights h_i and $\tilde{h}_i$. Typically, the index i takes on values 0, 1 and 2. Arithmetic on all indices is performed modulo 3. Observe that edges (and all related quantities) are generally labeled the same as the opposing vertices. The tilde decoration is used to distinguish corresponding quantities on the upper and lower triangles T and $\tilde{T}$, respectively. Throughout we use bold letters for vectors, and triangle and edge normals are all assumed to be normalized.

Energy. For a given triangle mesh, consider an arbitrary energy given by a summation over all the interior edges (indexed by i), or "hinges," of a triangle mesh,

$$E(\mathbf{x}) = \sum_i \psi_i(\theta_i) \,, \tag{1}$$

where the "bend angle" θ is the angle between the normals of the two triangles incident to the hinge, and $\psi : \mathbb{R} \to \mathbb{R}$ is an application-specific transformation of the bend angle. Drawing from the literature, examples for $\psi_i(\theta_i)$ include

$a_i(\theta_i - b_i)^2$	*Discrete shells [12]*
$a_i(\sin(\theta_i/2)^2$	*Discrete Willmore energy [17]*
$a_i(\cos(\theta_i/2) - b_i\theta_i)$	*Simulation of clothing [8]*

where a_i and b_i are application-specific scalar coefficients, which typically depend on the local geometry of the mesh and, in physical simulations, the material constitutive properties.

Reference [8] presented a force, not an energy; above we have integrated the (conservative) force to obtain the corresponding energy. By focusing on the conservative setting, we can roughly halve the computation time, since the conservative force Jacobian is the negated energy Hessian, which is symmetric by definition.

Bending forces and Hessians. We differentiate the energy (1) with respect to vertex positions $\mathbf{x}$ to obtain the bending forces and energy Hessian

$$\mathbf{f}(\mathbf{x}) = -\sum_i \nabla\psi_i \quad \text{and} \quad H(\mathbf{x}) = \sum_i \operatorname{Hess}(\psi_i) \,.$$

For one particular hinge i, dropping implied subscript from ψ_i and θ_i, the chain rule gives

$$\nabla\psi = \psi'\nabla\theta \,, \tag{2}$$

$$\operatorname{Hess}(\psi) = \psi' \operatorname{Hess}(\theta) + \psi''\nabla\theta^T\nabla\theta \,, \tag{3}$$

using the prime to differente a univariate function with respect to its scalar argument, e.g., $\psi' = \mathrm{d}\,\psi/\,\mathrm{d}\,\theta$.

Observe that the Hessian of the energy is a weighted sum of $\operatorname{Hess}(\theta)$ and the outer product $\nabla\theta^T\nabla\theta$, thus the same weighting function ψ' appears in both $\nabla\psi$ and $\operatorname{Hess}(\psi)$.

3. Hinge-angle gradient and Hessian

The expression for $\nabla\theta$ has been previously documented in the literature in several forms equivalent to

$$\nabla_{\mathbf{x}_1}\theta = \frac{\cos\alpha_2}{h_1}\mathbf{n}^T + \frac{\cos\tilde{\alpha}_2}{\tilde{h}_1}\tilde{\mathbf{n}}^T \qquad \nabla_{\mathbf{x}_0}\theta = -\frac{1}{h_0}\mathbf{n}^T$$

$$\nabla_{\mathbf{x}_2}\theta = \frac{\cos\alpha_1}{h_2}\mathbf{n}^T + \frac{\cos\tilde{\alpha}_1}{\tilde{h}_2}\tilde{\mathbf{n}}^T \qquad \nabla_{\mathbf{x}_3}\theta = -\frac{1}{\tilde{h}_0}\tilde{\mathbf{n}}^T \,. \tag{4}$$

By contrast, the expressions for the hinge angle Hessian are not (to our knowledge) recorded in the literature. Like others, we found the derivation to be extended and error-prone, and have therefore archived a complete derivation in an accompanying technical report [15]. The final expressions for $H^\theta \equiv \operatorname{Hess}(\theta)$ are

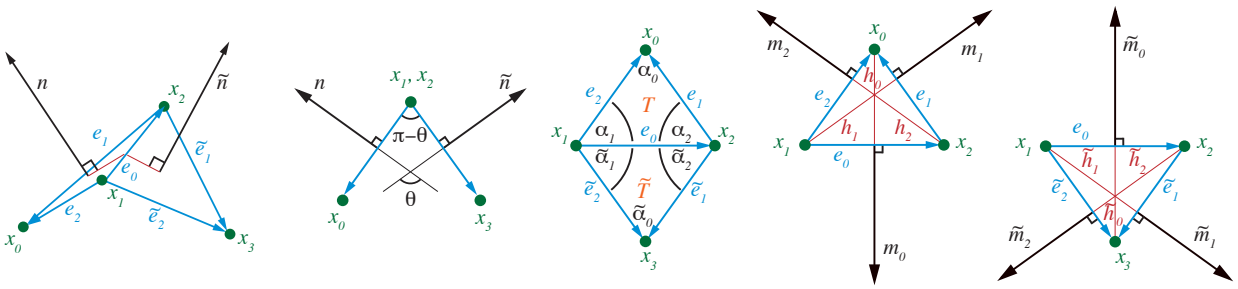

Figure 1: *Vertices, edges, normals and angles around the edge shared by two triangles. The two rightmost schematics show the in-plane edge normals and the associated altitudes from one edge to the opposing vertex. All of these are straightforward to compute given the edge vectors.*

conveniently expressed in terms of the building blocks

$$
\begin{aligned}
S(\mathbf{A}) &= \mathbf{A} + \mathbf{A}^T\,, \\
\omega_{ij} &= 1/(h_i h_j)\,, & \tilde{\omega}_{ij} &= 1/(\tilde{h}_i \tilde{h}_j)\,, \\
\mathbf{M}_i &= \mathbf{n}\mathbf{m}_i^T\,, & \tilde{\mathbf{M}}_i &= \tilde{\mathbf{n}}\tilde{\mathbf{m}}_i^T\,, \\
\mathbf{N}_i &= \mathbf{M}_i/\|\mathbf{e}_i\|^2\,, & \tilde{\mathbf{N}}_i &= \tilde{\mathbf{M}}_i/\|\tilde{\mathbf{e}}_i\|^2\,, \\
\mathbf{P}_{ij} &= \omega_{ij}\cos\alpha_i \mathbf{M}_j^T\,, & \tilde{\mathbf{P}}_{ij} &= \tilde{\omega}_{ij}\cos\tilde{\alpha}_i \tilde{\mathbf{M}}_j^T\,, \\
\mathbf{Q}_j &= \omega_{0j}\mathbf{M}_j\,, & \tilde{\mathbf{Q}}_j &= \omega_{0j}\tilde{\mathbf{M}}_j\,.
\end{aligned}
\tag{5}
$$

The 3×3 subblocks, H^θ_{ij}, of H^θ are

$$
\begin{aligned}
H^\theta_{00} &= -S(\mathbf{Q}_0) \\
H^\theta_{33} &= & -S(\tilde{\mathbf{Q}}_0) \\
\\
H^\theta_{11} &= S(\mathbf{P}_{11}) - \mathbf{N}_0 & +S(\tilde{\mathbf{P}}_{11}) - \tilde{\mathbf{N}}_0 \\
H^\theta_{22} &= S(\mathbf{P}_{22}) - \mathbf{N}_0 & +S(\tilde{\mathbf{P}}_{22}) - \tilde{\mathbf{N}}_0 \\
\\
H^\theta_{10} &= \mathbf{P}_{10} - \mathbf{Q}_1 \\
H^\theta_{20} &= \mathbf{P}_{20} - \mathbf{Q}_2 \\
H^\theta_{13} &= & \tilde{\mathbf{P}}_{10} - \tilde{\mathbf{Q}}_1 \\
H^\theta_{23} &= & \tilde{\mathbf{P}}_{20} - \tilde{\mathbf{Q}}_2 \\
\\
H^\theta_{12} &= \mathbf{P}_{12} + (\mathbf{P}_{21})^T + \mathbf{N}_0 & +\tilde{\mathbf{P}}_{12} + (\tilde{\mathbf{P}}_{21})^T + \tilde{\mathbf{N}}_0 \\
H^\theta_{03} &= 0\,.
\end{aligned}
\tag{6}
$$

contribution of upper triangle | contribution of lower triangle

The remaining blocks are obtained by symmetry of the Hessian, $H^\theta_{ij} = (H^\theta_{ji})^T$.

Exploiting symmetry. We have taken special care in laying out the expressions above, and in assigning the labels in Fig. 1. Observe that every contributing term depends on quantities from the hinge's upper triangle T, or lower triangle $\tilde{T}$, but not both. We write terms depending on T on the left column, and terms depending on $\tilde{T}$ on the right column. Comparing the two columns, we observe that *the two triangles contribute to the Hessian symmetrically.*

We will exploit this symmetry to derive a novel refactorization of the Hessian expressions, yielding a simpler, and more efficient, implementation.

4. Refactoring the bending energy Hessian

Assembling the Hessian for an entire mesh. Recall from (3) that the bending energy Hessian, $H(\mathbf{x})$, is the weighted sum of the hinge-angle Hessian, $\psi'\,\mathrm{Hess}(\theta)$, and the outer product of the hinge angle gradient with itself, $\psi''\nabla\theta^T\nabla\theta$. Therefore, it is natural to split the computation of the energy Hessian into two parts, iterating over triangles to compute $\sum \psi'_i\,\mathrm{Hess}(\theta_i)$, and iterating over interior edges to compute $\sum \psi''_i \nabla\theta_i^T\nabla\theta_i$; we examine these two parts in §4.1 and §4.2, respectively.

4.1. Exploiting two levels of symmetry in $\psi'\,\mathrm{Hess}(\theta)$

The half-hinge. As is evident from the two columns of (6), the two triangles of a hinge contribute to the hinge angle Hessian symmetrically. We exploit this symmetry by thinking of each hinge as a pair of *half-hinges* (see Fig. 2a).

The contributions of each half-hinge can each be computed using only the left column of (6), making the code compact, if care is taken to correct for the orientation of hinge edge $\mathbf{e}_0$.

To understand the needed correction, recall that the left and right columns of (6) are expressed with indices into the upper and lower triangles of Fig. 1, respectively. Hinge edge $\mathbf{e}_0$ flows *counter*clockwise versus clockwise

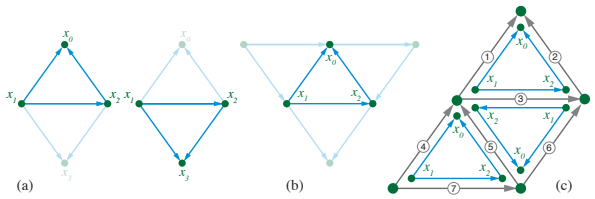

Figure 2: Refactoring the assembly: (a) *Every hinge is split into two* half-hinges. (b) *Each triangle associates with up to three half-hinges.* (c) *The Hessian is assembled by iterating a simple template over mesh triangles. In each iteration, the local indices of the template are mapped to corresponding global indices, and care is taken to account for mismatch in local/global edge orientation.*

along the upper and lower triangles, respectively. This disagreement in the assumed orientation of $\mathbf{e}_0$ is the only difference in the expressions derived for the left and right columns. In particular, if the lower half-hinge's contribution is computed using the expressions originally derived for the upper half-hinge, we must account for the reversal of $\mathbf{e}_0$.

To understand what is affected, we can rederive (6) with a revised Fig. 1 in which $\mathbf{e}_0$ is reversed. It turns out that the reversal affects only the computation of H_{12}, where $\mathbf{N}_0$ and $\tilde{\mathbf{N}}_0$ are now transposed. Since $\mathbf{N}_0 + \tilde{\mathbf{N}}_0$ is symmetric, transposing both does not alter the result, which is reassuring, since the Hessian should not depend on the (arbitrary choice of) orientation of $\mathbf{e}_0$ in the diagram. However, when we split the Hessian computation into a pair of half-hinges both reusing the left column of (6), this amounts to reversing the orientation in Fig. 1 only for the lower half-hinge: the pair of half-hinges are now computed with inconsistent versions of Fig. 1, an error we must correct: *we must transpose* $\mathbf{N}_0$ *for exactly one of the two half-hinge applications of the left column of* (6). This will be reflected below in our final computation.

The three half-hinges of a triangle. A second level of symmetry is uncovered by observing that each triangle participates in up to three half-hinges (see Fig. 2b). Because these half-hinges involve the same triangle, their contributions to the bending energy Hessian all depend on the same set of local quantitites. It therefore becomes natural to compute the bending energy Hessian by examining one triangle at a time.

Local triangle energy Hessian. The complete matrix $\sum \psi_i' \operatorname{Hess}(\theta_i)$ is assembled in the usual style of finite-element stiffness matrix assembly, by visiting each triangle and computing a local Hessian $H^{\triangle}$.

Consider the contribution of one triangle. If the triangle lies in the mesh interior, it participates in three half-hinges, but if it is incident to a boundary, it may participate in fewer hinges. To account for the boundary cases without specialized formulae, we introduce the indicator function

$$\sigma_i = \begin{cases} 1 & \text{edge } i \text{ lies in interior}\,, \\ 0 & \text{edge } i \text{ lies on boundary}\,. \end{cases}$$

We instantiate the left column of (6) thrice, with labels permuted in correspondence to each of the three half-hinges. Per (3), we scale each half-hinge Hessian contribution by ψ_i', and sum the scaled contributions to arrive at the *local triangle Hessian*. The use of the indicator function, and the summation over three potentially participating half-hinges, exposes the second level of symmetry— a three-fold symmetry over the edges, vertices, and indeed all labels on the triangle. This allows for a surprisingly compact representation of the 3×3 subblocks of $H^{\triangle}$ as

$$H^{\triangle}_{ij} = \omega_{ij}\left(d_i \mathbf{M}_j^T + d_j \mathbf{M}_i\right) + \begin{cases} -\mathbf{R}_{i+1} - \mathbf{R}_{i+2} & i = j\,, \\ \mathbf{R}_{i+2}{}^{\dagger} & i \neq j\,, \end{cases}$$

where

$$\begin{aligned} c_i &= \sigma_i \psi_i'(\theta_i)\,, \\ d_i &= c_{i-1}\cos\alpha_{i+1} + c_{i+1}\cos\alpha_{i-1} - c_i\,, \\ \mathbf{R}_i &= c_i \mathbf{N}_i\,. \end{aligned} \tag{7}$$

These expressions are valid for $i \in \{0,1,2\}$, $j \in \{i, i+1\}$. Remaining subblocks are determined by $H^{\triangle}_{ij} = (H^{\triangle}_{ji})^T$.

The conditional transpose operator (†). The above expressions employ the conditional transpose operator denoted by a dagger: $\mathbf{R}_i^{\dagger}$ transposes $\mathbf{R}_i$ if and only if the orientation of mesh edge $\mathbf{e}_i$ is counterclockwise with respect to the triangle of interest. The choice of global orientation is immaterial, so long as it is held fixed throughout the assembly of the complete Hessian. Indeed, the precise definition of † is also immaterial, so long as for every interior edge, it transposes for exactly one of the two incident triangles.

Operation count. Thus, to assemble the local Hessian for one triangle, we first compute three cosine expressions needed for d_i, and three outer products $\mathbf{M}_i$ assembled in 12 linear combinations. This is more compact than a naïve computation of the Hessian for each hinge. The Hessian for a single full hinge requires six different outer products, 20 scaled versions of these, and 18 matrix additions. For a regular mesh there are twice as

many edges as faces, so the total (relative) cost for the entire mesh becomes 12 outer products, 40 scale operations and 36 additions vs. 3, 12 and 15 operations. Assuming that the computation is compute bound, we should therefore expect roughly a 3× speedup. However, the locality of the above computation also improves cache-coherency, in practice leading to an additional speedup.

4.2. Computing $\psi'' \nabla\theta^T \nabla\theta$

All but two of the subblocks in the outer product contain mixed terms, i.e., terms involving data from both of incident triangles. Therefore, the outer product does not decompose in the way of the hinge angle Hessian; it is more naturally computed *per edge*.

However, while it is conceptually most naturally computed per edge, in practice it is still advantageous to include it in the same loop as the hinge-angle Hessian. We do this by assigning each edge to one of its two incident triangles similar to how the conditional transpose operator only transposes for one of the two triangles. The advantage of using this approach is primarily that the calls to the assembly function for the global stiffness matrix can be consolidated. Assume that each call to the assembly function adds one subblock of the Hessian. If the two loops are kept separate then a total of $6F + 10E$ calls to the assembly function are needed (where F denotes the number of faces and E denotes the number of edges). In a regular mesh this is approximately equal to $13E$. By comparison, the combined loop only requires $6F + 4E \approx 7E$ calls to the assembly function (the naïve computation of the Hessian requires $10E$ calls to the assembly function).

5. Implementation of a thin shell code testbed

Our own motivation to derive the hinge energy Hessian stemmed from the implementation of an implicit time stepper for cloth simulation. To obtain a complete implementation for a cloth simulation there are several additional observations which are useful. We will present those in this section, and then use this framework for the performance comparisons in the next section.

Tan-based energy. Discrete Shells [12] employs the hinge bending energy

$$\underbrace{\psi_i(\theta_i)}_{\textit{Discrete Shells}} = k\frac{3\|\bar{\mathbf{e}}_i\|^2}{\bar{A}_i}(\theta_i - \bar{\theta}_i)^2 \,,$$

where k is a bending stiffness and $\bar{A}_i$ is the sum of the areas of the two triangles incident to the hinge. A bar indicates that the quantity refers to the undeformed configuration.

More generally, one could consider some discrete approximation of the locally-integrated mean curvature, $\varphi(\theta_i)$, and then express the hinge bending energy as

$$\underbrace{\psi_i(\theta_i)}_{\textit{Generalized}} = k\, a_i \left(\varphi(\theta_i) - \varphi(\bar{\theta}_i)\right)^2 \,, \tag{8}$$

where a_i is a scaling coefficient that allows to account for the local discrete hinge geometry. This is similar to the formulation by Gingold et al. [11].

We found $\varphi(\theta) = 2\tan(\frac{\theta}{2})$ to be useful in our application. This expression was used by Bobenko and Suris [7], Hoffmann [14], and Gingold et al. [11] to estimate the sum of the principal curvatures. The tan-based energy is convenient in practice because it leads to monotonically increasing forces as the bend angle increases. For this particular choice of $\varphi(\theta)$ the scaling coefficient remains $a_i = 3\|\bar{\mathbf{e}}_i\|^2/\bar{A}_i$.

Physical material does not interpenetrate, thus we require

$$\theta \in (-\pi, \pi) \,, \tag{9}$$

$$\bar{\theta} \in (-\pi, \pi) \,. \tag{10}$$

The tan-based energy enforces (9), since the tan-based restoring force becomes unbounded as $\theta \to \pm\pi$; the condition (10) is enforced at initialization.

We also considered whether to employ $\varphi(\theta - \bar{\theta})$ in (8) in place of $\varphi(\theta) - \varphi(\bar{\theta})$. We choose the latter because the nonpenetration condition (9) examines θ, not $\theta - \bar{\theta}$. Some earlier papers, perhaps less focused on enforcement of nonpenetration in the elastic model, *do* employ $\varphi(\theta - \bar{\theta})$ [10, 16]. For small (infinitesimal) bend-angles all of these methods are equivalent (the curvature approaches zero as the bend-angles approaches zero). However, for large (finite) bend-angles the distinction becomes important.

Bending stiffness. For actual simulations the bending stiffness, k, must be chosen. By considering the energy of a thin plate under small deflections[1, Sec. 6.6] we note that it can be identified with *half* the flexural rigidity, D. While we consider shells here rather than plates, the model must be consistent with the plate model for a flat rest state. Hence

$$k = \frac{D}{2} = \frac{Yh^3}{24(1-\nu^2)} \tag{11}$$

where Y denotes Young's modulus, ν is Poisson's ratio, and h is the thickness of the shell.

Trigonometric functions of signed hinge angle. In order to be able to determine which way a shell is bending, the bend-angle $\theta \in [-\pi, \pi)$ has to be treated as a signed quantity. We use the same convention as Bridson et al. [8], where θ has the same sign as $(\mathbf{n}_1 \times \mathbf{n}_2) \cdot \mathbf{e}_0 = \det[\mathbf{n}_1, \mathbf{n}_2, \mathbf{e}_0]$, i.e., positive when the two normals point away from each other.

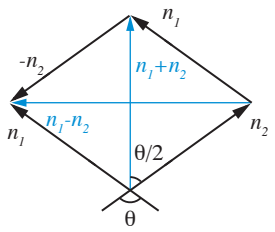

Figure 3: *Simple construction to compute the trigonometric functions for $\frac{\theta}{2}$ based on one of the right angled triangles. Note that each of the normal vectors have unit length.*

Referring to Fig. 3, the needed trigonometric functions for $\frac{\theta}{2}$ are

$$\sin\left(\tfrac{\theta}{2}\right) = \frac{\|\mathbf{n}_1 - \mathbf{n}_2\|}{2}, \qquad \cos\left(\tfrac{\theta}{2}\right) = \frac{\|\mathbf{n}_1 + \mathbf{n}_2\|}{2},$$

for $\theta \in [0; \pi]$. For values of θ outside the specified interval the usual identities for trigonometric functions can be used. In particular this gives

$$\tan\left(\tfrac{\theta}{2}\right) = \operatorname{sgn}(\det[\mathbf{n}_1, \mathbf{n}_2, \mathbf{e}_0]) \frac{\|\mathbf{n}_1 - \mathbf{n}_2\|}{\|\mathbf{n}_1 + \mathbf{n}_2\|} \tag{12}$$

for $\theta \in [-\pi; \pi]$. For $\theta = 0$, the determinant becomes zero and the sign function is undefined. However, in this case we have $\mathbf{n}_1 - \mathbf{n}_2 = 0$ so as long as any finite value is chosen for the sign function the result is correct. It should be noted that we never have to differentiate the right hand side of Equation (12). Instead we differentiate $\tan(\theta/2)$ using the chain rule. This avoids the trouble of the non-differentiability of the sign function at zero.

Efficient computation of viscous forces. The *discrete Rayleigh analogy* implementation of viscous damping introduces an *incremental dissipative potential*

$$E_v(\mathbf{x}) = \frac{k_d}{\Delta t} \sum_i a_i \left(\varphi(\theta_i) - \varphi(\hat{\theta}_i)\right)^2,$$

where $\hat{\theta}_i$ is the value of θ_i at the end of the previous time step and Δt is the size of the time step, [5]. In this model, viscosity is modeled as an elastic-type energy that binds the end-of-step position to its start-of-step counterpart. The parameter k_d is the damping coefficient which is conceptually similar to k in the elastic energy.

The *combined elastic and viscous* energy for a time step is then given by

$$\underbrace{\psi_i(\theta)}_{\textit{viscoelastic}} = k a_i \left(\varphi(\theta_i) - \varphi(\bar{\theta}_i)\right)^2 + \frac{k_d}{\Delta t} a_i \left(\varphi(\theta_i) - \varphi(\hat{\theta}_i)\right)^2.$$

and the expressions for the force and Hessian follow as before. In particular, per (3), the expensive hinge angle gradient, $\nabla\theta$, and Hessian, $\mathrm{Hess}(\theta)$, are computed just once (not twice), and the independent elastic and viscous contributions appear only in the scalars ψ' and ψ''. Compared to a separate evaluation of elastic and viscous forces this provides a 2× speedup since none of the vector or matrix computations are duplicated.

6. Method in brief

For convenience we collect all the equations into pseudocode in Algorithm 1 and 2. The computation of k may need to be done in different places depending on whether the material parameters are spatially and/or temporally varying. Note that on current microprocessors it can be advantageous in Algorithm 2 to compute and store the inverse of the edge lengths and triangle altitudes. That way the subsequent number of divisions is reduced significantly.

Algorithm 1 Preprocessing

Input: Undeformed mesh
1: **for all** edges, $\mathbf{e}_i$ **do**
2: Compute $\varphi(\bar{\theta}_i)$ using Equation (12).
3: Compute $a_i = 3\|\bar{\mathbf{e}}_i\|^2 / \bar{A}_i$.
4: Compute k using Equation (11).
5: **end for**

7. Evaluation

To illustrate the practical benefits of the results presented in the preceding sections we compare the performance and accuracy of our implementation to two existing alternatives. In particular we compare against the following methods:

- *Automatic differentiation.* Originally, Grinspun et al. [12] proposed to evaluate the force Jacobians using automatic differentiation. This method is characterized by providing accurate results without having to perform a lot of manual algebra.

Algorithm 2 ComputeForcesAndGradients

Input: Deformed mesh

for all edges, $\mathbf{e}_i$ **do**
  Compute inverse edge lengths, $1/l_i = 1/\|\mathbf{e}_i\|$.
  Compute unit edge, $\hat{\mathbf{e}}_i = \mathbf{e}_i/l_i$.
  Compute $\varphi(\theta_i)$ (Eq. 12).
  Compute ψ' and ψ''.
end for
for all triangles, $\mathcal{T}$ **do**
  Compute area, A, of triangle.
  Compute cosines, $\cos\alpha_i = \hat{\mathbf{e}}_j \cdot \hat{\mathbf{e}}_k$.
  Compute inverse altitudes, $1/h_i = l_i/2A$.
  Compute edge normals, $\mathbf{m}_i = \hat{\mathbf{e}}_i \times \mathbf{n}$.
  Compute $d_i, \omega_{ij}, \mathbf{M}_i, \mathbf{N}_i, \mathbf{R}_i$ (Eqs. 5 and 7).
  Compute contributions to the Hessian, $H^{\triangle}_{ij}$.
  Assemble $\nabla\mathbf{f}_{ij} = H^{\triangle}_{ij}$.
  for all nonboundary CCW edges in $\mathcal{T}$ **do**
    Compute bend gradient, $\nabla\theta$ (Eq. 4).
    Compute bending forces (Eq. 2).
    Compute $\psi''\nabla\theta^T\nabla\theta$.
    Update $\nabla\mathbf{f} = \nabla\mathbf{f} + \psi''\nabla\theta^T\nabla\theta$.
  end for
end for

- *Symbolic derivation*. The results presented in a number of papers, [6, 17, 10, 4], are based on an unpublished symbolic derivation of the bending forces and their gradients, but this derivation does not leverage all the available symmetry.

For each of these methods our comparison is based on the original source code provided by the authors, but ported into our testbed for a fair comparison. The comparison focuses on the computation of $\nabla\theta$ and Hess(θ), so all other computations are kept the same among all the implementations.

7.1. Test cases

We consider a number of different cloth simulations for our evaluations. One set of these has a square piece of cloth falling onto and draping over a (static) sphere (see Figure 4). This simulation is run at 36 different cloth resolutions (ranging from 121 vertices up to 6561 vertices). The goal of this test is to evaluate the behavior of the methods in the presence of large bend angles. No explicit damping is included in this example.

Another set of simulations shown in Figure 5 have a horizontal "beam" of varying thickness where one end is fully constrained while the remainder of the beam is allowed to bend under gravity. The goal of these simulations is to evaluate the behavior for a range of different

Figure 4: *A piece of cloth draping over a sphere. This simulation is run at 36 different resolutions.*

Figure 5: *Five different beams of varying thickness. The beams will be referred to by number with the one in front being number 1 and the one farthest away being number 5.*

materials from soft cloth to relatively stiff thin shells. These simulations have been damped to reach an equilibrium relatively quickly. In practice 5 different materials are considered and each one is run at three different resolutions (366, 787 and 1372 vertices).

All benchmarks have been run on an Intel Core i7-2640 at 2.80 GHz. Since this work is not focused on parallelization, everything is run in a single thread for easier comparisons. However, both the assembly of the Hessian and the solution of the resulting linear systems can obviously be parallelized.

7.2. Numerical accuracy

All of the methods considered for comparison should compute identical results in exact arithmetic. In floating point arithmetic slight differences are to be expected due

to rouding operations of intermediate results. To confirm the correctness of the new method and to evaluate its accuracy compared to the existing methods we have computed the normwise relative error of the Hessian at each time step of all the cloth-sphere simulations.

Let H_{new} denote the Hessian of the bending energy using the method presented in this paper, and let H_{old} denote the Hessian based on one of the existing methods. The normwise relative error is then given by :

$$\eta = \frac{\|H_{\text{new}} - H_{\text{old}}\|}{\|H_{\text{new}}\|}$$

For our computations we have used the Frobenius norm in the above expression. The distribution of the resulting errors is shown in Figure 6. As can be seen from this figure, the difference between the various implementations is typically on the order of $1 - 2$ ulp (unit in last place). The maximum error observed was 4.9 ulps, so the results are identical except for small differences which may affect up to the last $\lceil \log_2(4.9) \rceil = 3$ bits. These numbers are all based on double precision where one ulp is 2^{-53}. This is all consistent with our expectation that the results should be identical except for differences due to rounding. One thing worth noting is that the implementation based on automatic differentiation consistently has a higher error than the symbolic derivation. However, the difference is small enough that it is unlikely to be of practical significance.

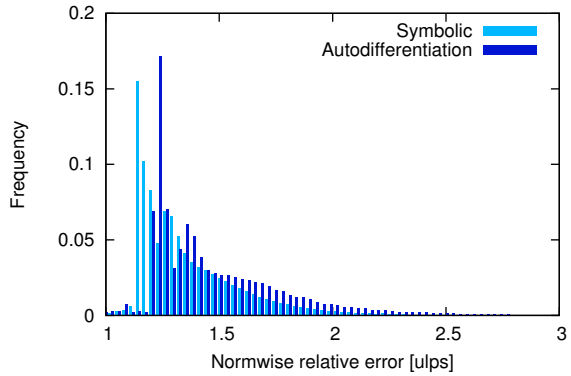

Figure 6: *The distribution of normwise relative errors when comparing the proposed method to an existing symbolic derivation and an implementation using automatic differentiation.*

7.3. Performance

Fundamentally, the flow of execution for the method proposed here is not dependent on the mesh configuration. We therefore expect that any performance gains are consistent across all the examples considered. Furthermore, we expect the cost to be proportional to the number of hinges. To verify this we first consider the runtime for the cloth-sphere example as a function of the number of vertices. The results are shown in Figure 7. Each of the three implementations considered show near perfect linear scaling. However, other parts of the cloth simulation do not scale linearly with the number of vertices. Most notably is the linear solver which in this case is PARDISO from Intel's MKL v. 11.0. With an estimated complexity of $O(n^{1.3})$ it is doing fairly well from a complexity point of view, but ultimately this is still bound to dominate any gain in the evaluation of the bending Hessian.

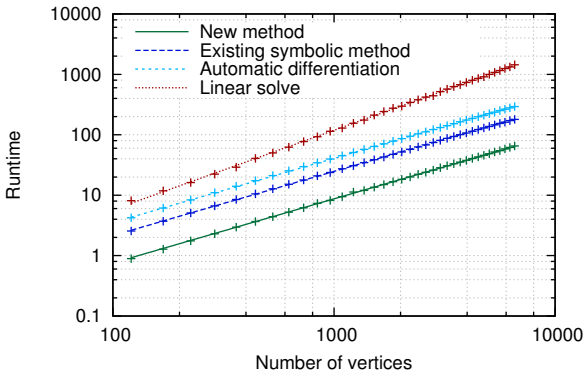

Figure 7: *The cost of evaluating bending forces and force gradients as well as assembling the associated stiffness matrix. Each of the three methods considered exhibit close to $O(n)$ complexity in the number of vertices. For comparison the cost of the linear solve is also shown. The estimated complexity for this phase is $O(n^{1.3})$.*

In addition to the linear solve there are other costs, like collision detection and response which depend on the particular simulation. Still, the fraction of the total time spent on bending force evaluation and stiffness matrix assembly can be significant as shown in Table 1.

Method	Cloth-sphere	Beams
Presented method	4–6%	9–12%
Existing symbolic	9–15%	17–23%
Autodifferentiation	14–22%	26–33%

Table 1: The fraction of the total simulation time which is spent on bending force evaluation and the associated stiffness matrix assembly. These numbers are minimums and maximums over all the examples in each group.

To get an accurate estimate of the speedup of the Hessian evaluation by itself we model the runtime cost as a

Experiment	k	$\ln a_N$	$\ln a_S$	$\ln a_A$	Speedup over symbolic	Speedup over autodiff
Cloth-sphere	1.062	−5.175	−4.129	−3.632	2.85	4.68
Beam 1	1.079	−2.629	−1.829	−1.287	2.22	3.83
Beam 2	1.075	−3.305	−2.505	−1.963	2.23	3.83
Beam 3	1.079	−4.711	−3.909	−3.370	2.23	3.82
Beam 4	1.081	−6.467	−5.674	−5.134	2.21	3.79
Beam 5	1.079	−7.056	−6.261	−5.715	2.21	3.82

Table 2: The results of the parameter estimation process for $\ln f(x) = k \ln x + \ln a$, where x is the number of vertices, and $f(x)$ is the runtime. The subscript "N" is used to denote the new method, "S" is used for the existing symbolic method, and "A" is used for the autodifferentiation method.

function of the number of vertices by $f(x) = ax^k$. We then estimate the two parameters, a and k by linear regression of the associated function $g(\ln x) \equiv \ln f(x) = k \ln x + \ln a$. Based on an initial data analysis we have observed that for each example the estimated exponent is the same for all the methods to within the statistical error of the estimate. Hence we have re-estimated $\ln a$ under the assumption that the value for k is the same for each method. Given the estimated values, the speedup is given as the ratio between the values of a for the different methods. The results are shown in Table 2.

It should be noted that counter to the expectation, there is a slight (but statistically significant) nonlinearity. This nonlinearity appears to be due to the sparse matrix data structure used for the assembly of the stiffness matrix.

What is measured and modeled above is the speedup for the combined evaluation of forces and force Jacobians plus the assembly of the stiffness matrix. The cost of the assembly operation is difficult to measure on its own in our codebase, but by using Intel's VTune we have estimated the assembly cost to be roughly 49% of the runtime for the new method, 25% for the existing symbolic method and also 25% for the automatic differentiation method. Factoring this in, the speedup for the computation of the bending forces and their Jacobians on their own is 1.5× the numbers shown in Table 2. This gives a speedup between 3× and 4× relative to the existing symbolic method, which is remarkably close to the rough estimate provided at the end of section 4.1. Compared to the automatic differentiation method the speedup can exceed 7×.

In addition to the comparisons above we have made a less extensive comparison to the approximate method employed by Baraff and Witkin [3]. This comparison suggests that computing the exact derivatives using our method is no more expensive than computing the approximate derivates using their method.

8. Limitations

Hinge based energies have a number of limitations which should be kept in mind in any implementation. These are not specific to the derivations in this paper, but rather inherent to the use of the hinge angle.

Degeneracies. The geometry depicted in Fig. 1 could degenerate (to first order) in two ways: *edge collapses*, where the hinge edge degenerates, and *altitude collapses*, where one of the vertices opposite the hinge edge becomes colinear with the hinge edge.

Either degeneracy causes at least one hinge altitude to vanish, leading to division by zero in the bend energy gradient and Hessian. In the case of an edge collapse, it is possible to show that all the relevant terms have well-defined limit values as the edge length decreases, so the division by zero can be handled by a numerically-stable special case. By contrast, we do not know of a remedy for the altitude collapse. In practice, the membrane (stretch) energy of a shell should prevent any element from degenerating, but for constitutive models like Saint-Venant Kirchhoff which allow inversion, a collapse remains possible. However, neither type of degeneracy has occured in any of our experiments.

Indefiniteness of the hinge angle Hessian. In many situations it is desirable for a matrix to be positive definite. In particular this is a requirement for being able to use the conjugate gradient method or a Cholesky based direct method to solve the corresponding set of linear equations. However, for general mesh positions, the Hessian of the hinge angle is indefinite.

The fact that it is not positive definite is not at all surprising. Strict convexity implies *uniqueness* of all solutions, which precludes bifurcation phenomena such as buckling [13, 2].

9. References

[1] B. Audoly, Y. Pomeau, Elasticity and Geometry: From hair curls to the nonlinear response of shells, Oxford University Press, 2010.

[2] J.M. Ball, Convexity conditions and existence theorems in nonlinear elasticity, Archive for Rational Mechanics and Analysis 63 (1976) 337–403.

[3] D. Baraff, A. Witkin, Large steps in cloth simulation, in: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, SIGGRAPH '98, ACM, New York, NY, USA, 1998, pp. 43–54.

[4] C. Batty, A. Uribe, B. Audoly, E. Grinspun, Discrete viscous sheets, ACM Trans. Graph. 31 (2012) 1131–1137.

[5] M. Bergou, B. Audoly, E. Vouga, M. Wardetzky, E. Grinspun, Discrete viscous threads, ACM Trans. Graph. 29 (2010) 116:1–116:10.

[6] M. Bergou, M. Wardetzky, D. Harmon, D. Zorin, E. Grinspun, A Quadratic Bending Model for Inextensible Surfaces, in: SGP '06: Proceedings of the fourth Eurographics Symposium on Geometry Processing, Eurographics Association, 2006, pp. 227–230.

[7] A.I. Bobenko, Y.B. Suris, Discrete Time Lagrangian Mechanics on Lie Groups, with an Application to the Lagrange Top, Communications in Mathematical Physics 204 (1999) 147–188.

[8] R. Bridson, S. Marino, R. Fedkiw, Simulation of clothing with folds and wrinkles, in: SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, 2003, pp. 28–36.

[9] S. Fröhlich, M. Botsch, Example-driven deformations based on discrete shells, Computer Graphics Forum 30 (2011) 2246–2257.

[10] A. Garg, E. Grinspun, M. Wardetzky, D. Zorin, Cubic shells, in: Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '07, Eurographics Association, 2007, pp. 91–98.

[11] Y. Gingold, A. Secord, J.Y. Han, E. Grinspun, D. Zorin, A Discrete Model for Inelastic Deformation of Thin Shells, Technical Report, Courant Institute of Mathematical Sciences, New York University, 2004.

[12] E. Grinspun, A.N. Hirani, M. Desbrun, P. Schröder, Discrete shells, in: SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, 2003, pp. 62–67.

[13] R. Hill, On uniqueness and stability in the theory of finite elastic strain, Journal of the Mechanics and Physics of Solids 5 (1957) 229–241.

[14] T. Hoffmann, Discrete Curves and Surfaces, Ph.D. thesis, Fachbereich 3 Mathematik der Technischen Universität Berlin, 2000.

[15] R. Tamstorf, Derivation of discrete bending forces and their gradients, Technical Report, Walt Disney Animation Studios, 2013.

[16] M. Wardetzky, M. Bergou, A. Garg, D. Harmon, D. Zorin, E. Grinspun, Discrete Differential Geometry: An Applied Introduction, in: SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 Courses, ACM, New York, NY, USA, 2008.

[17] M. Wardetzky, M. Bergou, D. Harmon, D. Zorin, E. Grinspun, Discrete Quadratic Curvature Energies, Computer Aided Geometric Design 24 (2007) 499–518.