Simulation of nonlinear Kirchhoff-Love thin shells using subdivision finite elements

David Clyde[*]
University of California, Los Angeles

Joseph Teran[†]
University of California, Los Angeles

Rasmus Tamstorf
Walt Disney Animation Studios

ABSTRACT

This document presents the details necessary for simulation of thin shells with finite strains based on the Kirchhoff-Love assumptions. With an eye towards cloth simulation, we combine this with a nonlinear orthotropic constitutive model. We also leverage a conforming spatial discretization using Catmull-Clark subdivision surfaces to ensure convergence under refinement. The dynamics is handled in a fully implicit fashion to allow for large timesteps and solution of quasi-static problems.

KEYWORDS

subdivision finite elements, isogeometric analysis, thin shell simulation, cloth simulation

ACM Reference format:
David Clyde, Joseph Teran, and Rasmus Tamstorf. 2017. Simulation of nonlinear Kirchhoff-Love thin shells using subdivision finite elements. In *Proceedings of SCA '17, Los Angeles, CA, USA, July 28-30, 2017,* 7 pages.
https://doi.org/10.1145/3099564.3099577

1 RELATED WORK

The method presented here is an example of "isogeometric analysis" (IGA), where subdivision surfaces are used to represent both the geometry (of cloth) and the basis functions for finite element analysis. This concept has been credited to [Hughes et al. 2005], but subdivision finite elements were originally introduced by [Cirak et al. 2000], where they were applied to linear elastic Kirchhoff-Love thin shells. The method was later extended to finite strain analysis in [Cirak and Ortiz 2001], where they considered a Neo-Hookean material but used explicit integration for the temporal evolution. In graphics, [Thomaszewski et al. 2006] combined the subdivision finite elements with a co-rotational linear elasticity model and introduced implicit time integration using the backward Euler scheme. More recently, [Long et al. 2012] considered shear deformable shells, but limited their discussion to quasi-static problems. The work by [Vetter et al. 2013] investigates growth problems using the St. Venant-Kirchhoff material model along with an explicit Newmark predictor-corrector integration scheme. All of this prior work is based on Loop subdivision on triangle meshes.

As an alternative to Loop subdivision, [Wawrzinek et al. 2011] considers the Catmull-Clark subdivision scheme with linear elasticity for a variety of applications that do not involve dynamics.

2 KIRCHHOFF-LOVE KINEMATICS

The Kirchhoff-Love shell theory models thin shell kinematics in terms of a 2D midsurface by assuming that straight lines initially perpendicular to the midsurface remain straight lines perpedicular to the midsurface and retain their length during deformation.

In this work, we assume a constant thickness τ per cloth object. We represent the deformed midsurface by a map $\boldsymbol{x}$ from a 2D parameter space ω to worldspace Ω. The corresponding map describing the volumetric cloth object is denoted by $\boldsymbol{r} : \omega^\tau \rightarrow \Omega^\tau$, where ω^τ is the 3D parameter space, and Ω^τ is the region of $\mathbb{R}^3$ occupied by the shell. The curvilinear coordinates for a point in the shell are denoted by $\boldsymbol{\xi} = (\xi^1, \xi^2, \xi^3)$, where superscripts indicate contravariant indices (not to be confused with exponentiation). Given these conventions, the Kirchhoff-Love assumptions directly imply

$$\boldsymbol{r}(\xi^1, \xi^2, \xi^3) = \boldsymbol{x}(\xi^1, \xi^2) + \xi^3 \boldsymbol{a}_3(\xi^1, \xi^2)\ , \quad -\tfrac{\tau}{2} \le \xi^3 \le \tfrac{\tau}{2},$$

where $\boldsymbol{a}_3(\xi^1, \xi^2)$ is the unit normal to the deformed midsurface. In the following, we use Greek letters for indices in $\{1, 2\}$, lowercase Latin letters for indices in $\{1, 2, 3\}$, and uppercase Latin letters for indices that range from 1 to n, where $n > 2$. Furthermore we use the comma notation to denote partial derivatives with respect to ξ^α such that $\boldsymbol{x}_{,\alpha} = \partial \boldsymbol{x} / \partial \xi^\alpha$. For convenience, let $\mathbf{a}_\alpha = \boldsymbol{x}_{,\alpha}$ denote the covariant basis vectors of the midsurface or, equivalently, the columns of the midsurface mapping Jacobian. We can then write the surface normal as

$$\boldsymbol{a}_3 = \frac{\mathbf{a}_1 \times \mathbf{a}_2}{\|\mathbf{a}_1 \times \mathbf{a}_2\|}.$$

Also, let $\mathbf{g}_i$ denote the covariant curvilinear basis vectors $\frac{\partial \boldsymbol{r}}{\partial \xi^i}$, and use $\mathbf{g}^i$ to denote the corresponding contravariant basis vectors; thus, $\mathbf{g}^i \cdot \mathbf{g}_j = \delta_i^j$ and

$$\begin{pmatrix} \mathbf{g}_1 & \mathbf{g}_2 & \mathbf{g}_3 \end{pmatrix} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\xi}} = \begin{pmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \mathbf{0} \end{pmatrix} + \begin{pmatrix} \xi^3 \boldsymbol{a}_{3,1} & \xi^3 \boldsymbol{a}_{3,2} & \boldsymbol{a}_3 \end{pmatrix}$$

Finally, define $\mathrm{G} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\xi}}^T \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\xi}}$ as the covariant metric tensor with entries $g_{ij} = \mathbf{g}_i \cdot \mathbf{g}_j$. Throughout this document, we use overbar notation for quantities related to the undeformed configuration.

[*]Also with Walt Disney Animation Studios.
[†]Also with Walt Disney Animation Studios.

Thus, for example, $\bar{\boldsymbol{x}} : \omega \to \overline{\Omega}$ denotes the undeformed midsurface map; analogous definitions are made for $\bar{\boldsymbol{r}}$, $\bar{\boldsymbol{a}}_3$, and so on.

2.1 Deformation gradient and strain tensor

The deformation function from the undeformed shell to the deformed shell is given by

$$\boldsymbol{\phi}(\bar{\boldsymbol{x}}) = \boldsymbol{r}(\bar{\boldsymbol{r}}^{-1}(\bar{\boldsymbol{x}}))$$

and the deformation gradient $\mathbf{F} : \overline{\Omega} \to \mathbb{R}^{3\times 3}$ is the Jacobian of $\boldsymbol{\phi}(\bar{\boldsymbol{x}})$:

$$\mathbf{F} = \frac{\partial \boldsymbol{\phi}}{\partial \bar{\boldsymbol{x}}} = \frac{\partial \boldsymbol{r}}{\partial \xi}\left(\frac{\partial \bar{\boldsymbol{r}}}{\partial \xi}\right)^{-1} = \mathbf{g}_i \otimes \bar{\mathbf{g}}^i. \tag{1}$$

Note that here, as well as in the following, there is an implied summation over repeated indices. Given the deformation gradient, the Green-Lagrange strain is

$$\begin{aligned}\mathbf{E} &= \tfrac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right)\\ &= \tfrac{1}{2}\left((\bar{\mathbf{g}}^i \otimes \mathbf{g}_i)(\mathbf{g}_j \otimes \bar{\mathbf{g}}^j) - \mathbf{I}\right)\\ &= \tfrac{1}{2}\left(g_{ij}\bar{\mathbf{g}}^i \otimes \bar{\mathbf{g}}^j - \mathbf{I}\right)\end{aligned} \tag{2}$$

The entries g_{ij} satisfy

$$\begin{aligned}g_{\alpha\beta} &= (\boldsymbol{a}_\alpha + \xi^3\boldsymbol{a}_{3,\alpha})^T(\boldsymbol{a}_\beta + \xi^3\boldsymbol{a}_{3,\beta})\\ g_{\alpha 3} &= 0\\ g_{33} &= 1,\end{aligned} \tag{3}$$

in which $g_{\alpha 3}$ has been simplified using $\mathbf{a}_\alpha \cdot \mathbf{a}_3 = \mathbf{a}_{3,\alpha} \cdot \mathbf{a}_3 = 0$. By using

$$\boldsymbol{a}_{3,\alpha} = \frac{1}{\|\mathbf{a}_1 \times \mathbf{a}_2\|}\left(\mathbf{I} - \boldsymbol{a}_3\boldsymbol{a}_3{}^T\right)(\boldsymbol{a}_{1,\alpha} \times \boldsymbol{a}_2 + \boldsymbol{a}_1 \times \boldsymbol{a}_{2,\alpha}), \tag{4}$$

we obtain $g_{\alpha\beta}$ from the inputs $\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_{1,1}, \boldsymbol{a}_{1,2}, \boldsymbol{a}_{2,2}, \xi^3$. The analogous formula for $\bar{\boldsymbol{a}}_{3,\alpha}$ yields $\bar{\mathbf{g}}_j$ and ultimately $\bar{\mathbf{g}}^j$ from $\bar{\boldsymbol{a}}_1, \bar{\boldsymbol{a}}_2, \bar{\boldsymbol{a}}_{1,1}, \bar{\boldsymbol{a}}_{1,2}, \bar{\boldsymbol{a}}_{2,2}, \xi^3$. We define the column vector $\mathbf{z} = \mathbf{z}(\xi)$ in $\mathbb{R}^{15}$ as the concatenation of $\mathbf{z}_1 = \boldsymbol{a}_1$, $\mathbf{z}_2 = \boldsymbol{a}_2$, $\mathbf{z}_3 = \boldsymbol{a}_{1,1}$, $\mathbf{z}_4 = \boldsymbol{a}_{1,2}$, and $\mathbf{z}_5 = \boldsymbol{a}_{2,2}$, i.e., $\mathbf{z} = (\mathbf{z}_1^T, \ldots, \mathbf{z}_5^T)^T$. In the following, it will be convenient to use the notation $\mathbf{E} = \mathbf{E}(\mathbf{z}, \xi^3)$, leaving implicit the dependence on the rest configuration.

3 ORTHOTROPIC CONSTITUTIVE MODEL

A hyperelastic material is *orthotropic* provided that there exists an orthonormal basis in the undeformed space such that the elastic potential is invariant under replacement of $\mathbf{E}$ with $\mathbf{Q}^T\mathbf{E}\mathbf{Q}$ for any element $\mathbf{Q}$ of the orthotropic symmetry group. Specifically, this means that it is invariant under reflection across any of the given basis vectors. In modelling woven cloth, the orthotropy basis is specified by the direction matrix $\mathbf{D} = [\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3]$, where $\boldsymbol{d}_1$, $\boldsymbol{d}_2$, and $\boldsymbol{d}_3$ correspond to the (normalized) material warp, weft, and normal directions, respectively.

Let $\mathbf{L}_j$ be the reflection matrix $\mathbf{I} - 2\boldsymbol{d}_j\boldsymbol{d}_j^T$; then, to demonstrate orthotropic symmetry, we must establish that $\mathbf{F}$ and $\mathbf{F}\mathbf{L}_j$ produce the same energy density for any $\mathbf{F}$ and for each j. If the energy density is denoted by ψ, then this means that that $\psi(\mathbf{E}) = \psi(\mathbf{L}_j\mathbf{E}\mathbf{L}_j)$.

To construct such an energy density, define the reduced strain $\tilde{\mathbf{E}}$ as $\tilde{\mathbf{E}} = \mathbf{D}^T\mathbf{E}\mathbf{D}$. By definition of the deformation gradient, $\mathbf{F}\boldsymbol{d}_\alpha$ lies in the deformed configuration tangent plane, while the Kirchhoff-Love assumptions imply $\mathbf{F}\boldsymbol{d}_3$ is a unit vector in the deformed configuration's normal direction. Thus, $\boldsymbol{d}_\alpha^T\mathbf{F}^T\mathbf{F}\boldsymbol{d}_3 = 0$ and $\boldsymbol{d}_3^T\mathbf{F}^T\mathbf{F}\boldsymbol{d}_3 = 1$, so $\tilde{\mathbf{E}}$ has the following block structure:

$$\tilde{\mathbf{E}} = \begin{pmatrix} \tilde{E}_{11} & \tilde{E}_{12} & 0\\ \tilde{E}_{12} & \tilde{E}_{22} & 0\\ 0 & 0 & 0\end{pmatrix}.$$

We then claim that $\tilde{E}_{\alpha\alpha} = \boldsymbol{d}_\alpha^T\mathbf{E}\boldsymbol{d}_\alpha$ and $\tilde{E}_{12}^2 = (\boldsymbol{d}_1^T\mathbf{E}\boldsymbol{d}_2)^2$ are invariant under the replacement $\mathbf{E} \to \mathbf{L}_j\mathbf{E}\mathbf{L}_j$. Indeed,

$$\begin{aligned}\boldsymbol{d}_\alpha^T\mathbf{L}_j\mathbf{E}\mathbf{L}_j\boldsymbol{d}_\alpha &= \begin{cases}(-\boldsymbol{d}_\alpha)^T\mathbf{E}(-\boldsymbol{d}_\alpha) & \text{if } \alpha = j\\ \boldsymbol{d}_\alpha^T\mathbf{E}\boldsymbol{d}_\alpha & \text{if } \alpha \neq j\end{cases}\\ &= \boldsymbol{d}_\alpha^T\mathbf{E}\boldsymbol{d}_\alpha\end{aligned}$$

and

$$\begin{aligned}\left(\boldsymbol{d}_1^T\mathbf{L}_j\mathbf{E}\mathbf{L}_j\boldsymbol{d}_2\right)^2 &= \begin{cases}\left(-\boldsymbol{d}_1^T\mathbf{E}\boldsymbol{d}_2\right)^2 & \text{if } 1 \le j \le 2\\ \left(\boldsymbol{d}_1^T\mathbf{E}\boldsymbol{d}_2\right)^2 & \text{if } j = 3\end{cases}\\ &= \left(\boldsymbol{d}_1^T\mathbf{E}\boldsymbol{d}_2\right)^2.\end{aligned}$$

From this it follows that any energy density, ψ, written as a function of $\tilde{E}_{\alpha\alpha}$ and $\tilde{E}_{12}^2$, is orthotropically invariant. In the following we consider the case where ψ is given by

$$\psi = \frac{a_{11}}{2}\eta_1(\tilde{E}_{11}^2) + a_{12}\eta_2(\tilde{E}_{11}\tilde{E}_{22}) + \frac{a_{22}}{2}\eta_3(\tilde{E}_{22}^2) + G_{12}\eta_4(\tilde{E}_{12}^2) \tag{5}$$

$$\eta_j(x) = \sum_{i=1}^{d_j}\frac{\mu_{ji}}{\alpha_{ji}}\left((x+1)^{\alpha_{ji}} - 1\right)$$

In this context, a_{11}, a_{22}, a_{12}, and G_{12} are all material parameters that should not be confused with $\boldsymbol{a}_{1,1}, \boldsymbol{a}_{1,2}, \boldsymbol{a}_{2,2}$ or the $(1,2)$ component of the $\mathbf{G}$ matrix. μ_{ji} and α_{ji} are also material parameters.

4 GOVERNING PDE

4.1 Weak form derivation

The PDE for evolution of the midsurface map $\boldsymbol{x}$ may be derived from Lagrangian mechanics. For $\boldsymbol{x} \in H^2(\omega \to \mathbb{R}^3)$, the Lagrangian is defined as the difference $L = T - V$ between kinetic energy T and potential energy V. The governing equations are then

$$\frac{\partial}{\partial t}\left(\frac{\partial L}{\partial \dot{\boldsymbol{x}}}\right) = \frac{\partial L}{\partial \boldsymbol{x}}$$

and the corresponding weak form is

$$\int_\omega \frac{\partial}{\partial t}\left(\frac{\partial L}{\partial \dot{\boldsymbol{x}}}\right)\boldsymbol{v}\,d\xi^1 d\xi^2 = \int_\omega \left(\frac{\partial L}{\partial \boldsymbol{x}}\right)\boldsymbol{v}\,d\xi^1 d\xi^2$$

for an arbitrary test function $\boldsymbol{v} \in H^2(\omega \to \mathbb{R}^3)$. Derivatives of L are interpreted in variational sense; for example,

$$\int_\omega \left(\frac{\partial L}{\partial \boldsymbol{x}}\right)\boldsymbol{v}\,d\xi^1 d\xi^2 = \left.\frac{\partial L(\boldsymbol{x} + \varepsilon\boldsymbol{v})}{\partial \varepsilon}\right|_{\varepsilon=0}.$$

Let $\bar{J}$ denote the Jacobian determinant $|\frac{\partial \bar{\boldsymbol{r}}}{\partial \boldsymbol{\xi}}|$ and $\hat{\rho}$ the density pull-back $\bar{\rho} \circ \bar{\boldsymbol{r}}$. Then the kinetic energy is

$$\begin{aligned} T &= \frac{1}{2}\int_{\bar{\Omega}^\tau} \bar{\rho}\|\dot{\boldsymbol{\phi}}\|^2 d\bar{\boldsymbol{x}} \\ &= \frac{1}{2}\int_{\omega^\tau} \hat{\rho}\|\dot{\boldsymbol{r}}\|^2 \bar{J} d\boldsymbol{\xi} \\ &= \frac{1}{2}\int_{\omega^\tau} \hat{\rho}\|\dot{\boldsymbol{x}} + \xi^3 \dot{\mathrm{a}}_3\|^2 \bar{J} d\boldsymbol{\xi}. \end{aligned}$$

The term $\xi^3 \dot{\mathrm{a}}_3$ introduces complex dependencies on both $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$, but the multiplier ξ^3 makes this term small in most contexts. For simplicity, we discard this term, resulting in the standard form for the finite element mass matrix in Equation (9), which is also used by e.g. [Cirak and Ortiz 2001; Kiendl et al. 2015]. Based on this simplification, the resulting variational derivatives then satisfy

$$\int_\omega \left(\frac{\partial T}{\partial \boldsymbol{x}}\right) \boldsymbol{v} d\xi^1 d\xi^2 = 0$$

$$\int_\omega \frac{\partial}{\partial t}\left(\frac{\partial T}{\partial \dot{\boldsymbol{x}}}\right) \boldsymbol{v} d\xi^1 d\xi^2 = \int_{\omega^\tau} \hat{\rho}\ddot{\boldsymbol{x}}^T \boldsymbol{v} \bar{J} d\boldsymbol{\xi}.$$

Given a general hyperelastic energy density $\psi = \psi(\mathbf{z})$ and a constant gravity field $\mathbf{g} \in \mathbb{R}^3$, the potential energy is

$$\begin{aligned} V &= \int_{\bar{\Omega}^\tau} \psi(\mathbf{z}(\bar{\boldsymbol{r}}^{-1}(\bar{\boldsymbol{x}}))) + \bar{\rho}(\bar{\boldsymbol{x}})\boldsymbol{g}^T \boldsymbol{\phi}(\bar{\boldsymbol{x}}) d\bar{\boldsymbol{x}} \\ &= \int_{\omega^\tau} \left(\psi(\mathbf{z}(\boldsymbol{\xi})) + \hat{\rho}(\boldsymbol{\xi})\mathbf{g}^T \boldsymbol{r}(\boldsymbol{\xi})\right) \bar{J} d\boldsymbol{\xi} \end{aligned}$$

with derivatives satisfying

$$\begin{aligned} \int_\omega \left(\frac{\partial V}{\partial \boldsymbol{x}}\right) \boldsymbol{v} d\xi^1 d\xi^2 = \int_{\omega^\tau} &\left(\frac{\partial \psi}{\partial \boldsymbol{x}_{,1}} \boldsymbol{v}_{,1} + \frac{\partial \psi}{\partial \boldsymbol{x}_{,2}} \boldsymbol{v}_{,2}\right. \\ &\left. + \frac{\partial \psi}{\partial \boldsymbol{x}_{,11}} \boldsymbol{v}_{,11} + \frac{\partial \psi}{\partial \boldsymbol{x}_{,12}} \boldsymbol{v}_{,12} + \frac{\partial \psi}{\partial \boldsymbol{x}_{,22}} \boldsymbol{v}_{,22} + \hat{\rho}\boldsymbol{g}^T \boldsymbol{v}\right) \bar{J} d\boldsymbol{\xi} \end{aligned}$$

and

$$\int_\omega \frac{\partial}{\partial t}\left(\frac{\partial V}{\partial \dot{\boldsymbol{x}}}\right) \boldsymbol{v} d\xi^1 d\xi^2 = 0$$

The weak form PDE is then

$$\begin{aligned} \int_{\omega^\tau} \hat{\rho}\ddot{\boldsymbol{x}}^T \boldsymbol{v} \bar{J} d\boldsymbol{\xi} = -\int_{\omega^\tau} &\left(\frac{\partial \psi}{\partial \boldsymbol{x}_{,1}} \boldsymbol{v}_{,1} + \frac{\partial \psi}{\partial \boldsymbol{x}_{,2}} \boldsymbol{v}_{,2}\right. \\ &\left. + \frac{\partial \psi}{\partial \boldsymbol{x}_{,11}} \boldsymbol{v}_{,11} + \frac{\partial \psi}{\partial \boldsymbol{x}_{,12}} \boldsymbol{v}_{,12} + \frac{\partial \psi}{\partial \boldsymbol{x}_{,22}} \boldsymbol{v}_{,22} + \hat{\rho}\boldsymbol{g}^T \boldsymbol{v}\right) \bar{J} d\boldsymbol{\xi} \end{aligned} \tag{6}$$

Note that the previously stated conditions $\boldsymbol{x}, \boldsymbol{v} \in H^2(\omega \to \mathbb{R}^3)$ are sufficient to make the weak form well-defined.

4.2 Strong form and boundary terms

Integrating the weak form by parts yields the corresponding strong form

$$\begin{aligned} \hat{\rho}\ddot{\boldsymbol{x}}\bar{J} = &-\hat{\rho}\mathbf{g}\bar{J} + \left(\frac{\partial \psi}{\partial \boldsymbol{x}_{,1}}\bar{J}\right)_{,1} + \left(\frac{\partial \psi}{\partial \boldsymbol{x}_{,2}}\bar{J}\right)_{,2} \\ &-\left(\frac{\partial \psi}{\partial \boldsymbol{x}_{,11}}\bar{J}\right)_{,11} - \left(\frac{\partial \psi}{\partial \boldsymbol{x}_{,12}}\bar{J}\right)_{,12} - \left(\frac{\partial \psi}{\partial \boldsymbol{x}_{,22}}\bar{J}\right)_{,22} \\ &+ \text{boundary terms} \end{aligned} \tag{7}$$

The integration by parts produces several boundary terms on the right-hand side of Equation (7), which may be expressed in terms of the outward normal $\boldsymbol{n} = (n_1, n_2, n_3)$ on $\partial\omega^\tau$:

$$\begin{aligned} \int_{\partial\omega^\tau} \Bigg\{ &n_1\left[\frac{\partial \psi}{\partial \boldsymbol{x}_{,1}}\bar{J} - \left(\frac{\partial \psi}{\partial \boldsymbol{x}_{,11}}\bar{J}\right)_{,1} - \frac{1}{2}\left(\frac{\partial \psi}{\partial \boldsymbol{x}_{,12}}\bar{J}\right)_{,2}\right]\boldsymbol{v} \\ &+n_2\left[\frac{\partial \psi}{\partial \boldsymbol{x}_{,2}}\bar{J} - \left(\frac{\partial \psi}{\partial \boldsymbol{x}_{,22}}\bar{J}\right)_{,2} - \frac{1}{2}\left(\frac{\partial \psi}{\partial \boldsymbol{x}_{,12}}\bar{J}\right)_{,1}\right]\boldsymbol{v} \\ &+\left[n_1\frac{\partial \psi}{\partial \boldsymbol{x}_{,11}}\bar{J} + \frac{1}{2}n_2\frac{\partial \psi}{\partial \boldsymbol{x}_{,12}}\bar{J}\right]\boldsymbol{v}_{,1} \\ &+\left[n_2\frac{\partial \psi}{\partial \boldsymbol{x}_{,22}}\bar{J} + \frac{1}{2}n_1\frac{\partial \psi}{\partial \boldsymbol{x}_{,12}}\bar{J}\right]\boldsymbol{v}_{,2}\Bigg\} d\boldsymbol{\xi} \end{aligned}$$

In practice, we set these Neumann boundary conditions equal to 0 and impose any desired position constraints by Dirichlet means instead. We implement three distinct types of boundary conditions. A clamped boundary has specified positions and normals for $\boldsymbol{x}$; a supported boundary has specified positions only; and a free boundary is not subject to any Dirichlet constraint.

5 SPATIAL DISCRETIZATION

We use the finite element method (FEM) to spatially discretize the governing weak form PDE in Equation (6). To provide the necessary H^2 smoothness, we use subdivision finite elements as originally introduced by [Cirak et al. 2000]. For purposes of this section, the actual choice of subdivision scheme does not matter as long as it guarantees sufficient smoothness. As with most finite element methods, we can express the kinematics as a linear combination of basis functions :

$$\boldsymbol{x}(\mathbf{q}; \xi^1, \xi^2) = \mathbf{q}_I^T N_I(\xi^1, \xi^2), \quad x_j(\mathbf{q}; \xi^1, \xi^2) = \mathrm{q}_{Ij} N_I(\xi^1, \xi^2), \tag{8}$$

where $\mathbf{q}_I = (\mathrm{q}_{I1}, \mathrm{q}_{I2}, \mathrm{q}_{I3}) \in \mathbb{R}^3$ is a row-vector. These $\mathbf{q}_I$ are often naturally interpreted geometrically. For example, with piecewise linear interpolation, these are the locations of the mesh grid nodes. However, with subdivision elements, they are control vertices; more generally, they can be thought of as generalized coordinates, which is why they are denoted by $\mathbf{q}$. Note that summation is implied on the repeated index I, where $I = 1, \ldots, n_v$ and n_v is the number of control vertices. The set $\{\mathbf{q}_{Ij}\}$ can be thought of as a $n_v \times 3$ matrix of control point locations. With a slight abuse of notation, we use $\mathbf{q}$ to denote the corresponding column vector of all the control vertices.

We now substitute the discretized $\boldsymbol{x}$ into the weak form in Equation (6) with $\boldsymbol{v} = N_K \boldsymbol{e}_j$ for arbitrary K and standard basis vectors $\boldsymbol{e}_j \in \mathbb{R}^3$. Note e.g. $\frac{\partial \psi}{\partial \boldsymbol{x}_{,1}} \boldsymbol{v}_{,1}$ becomes $\frac{\partial \psi}{\partial x_{j,1}} N_{K,1} = \frac{\partial \psi}{\partial x_{j,1}} \frac{\partial x_{j,1}}{\partial q_{Kj}}$. The resulting system of equations and exact mass matrix entries are

$$\begin{aligned} \tilde{M}_{KI}\ddot{\mathbf{q}}_I &= -\int_{\omega^\tau} \left(\frac{\partial \psi}{\partial \mathbf{q}_K} + \hat{\rho}\mathbf{g}^T N_K\right) \bar{J} d\boldsymbol{\xi} \\ \tilde{M}_{KI} &= \int_{\omega^\tau} \hat{\rho} N_I N_K \bar{J} d\boldsymbol{\xi}. \end{aligned} \tag{9}$$

6 QUADRATURE

Evaluating the above integrals numerically using a quadrature rule with evaluation sites $\boldsymbol{\xi}^{(r)}$ and corresponding weights $d\boldsymbol{\xi}^{(r)}$ gives

$$M_{KI}\ddot{\mathbf{q}}_I = \boldsymbol{f}_K(\mathbf{q}), \tag{10}$$

where

$$M_{KI} = \sum_r \hat{\rho}^{(r)} N_I^{(r)} N_K^{(r)} \bar{J}^{(r)} d\xi^{(r)}$$

$$\boldsymbol{f}_K(\mathbf{q}) = -\sum_r \left(\frac{\partial \psi}{\partial \mathbf{q}_K} \left(\mathbf{E}^{(r)} \right) + \hat{\rho}^{(r)} \mathbf{g}^T N_K^{(r)} \right) \bar{J}^{(r)} d\xi^{(r)}$$

In practice, we use a lumped mass matrix, which combines each row's entries onto the diagonal with the result

$$M_{II} = \sum_r \hat{\rho}^{(r)} N_I^{(r)} \bar{J}^{(r)} d\xi^{(r)}$$

and $M_{IK} = 0$ for $I \neq K$.

Regardless of which mass matrix is being used, we need to compute $\hat{\rho}^{(r)}$. For a uniform mass distribution, this is trivial. For a non-uniform mass distribution, the density will typically be specified at each control vertex, so we denote the corresponding set of values by $\hat{\rho}_L$, where $L = 1, \ldots, n_v$. To compute $\hat{\rho}^{(r)}$, we then interpolate our density function at the quadrature points:

$$\hat{\rho}^{(r)} = \hat{\rho}_L N_L^{(r)}.$$

For the lumped mass matrix this leads to

$$M_{II} = \sum_r \hat{\rho}_L N_L^{(r)} N_I^{(r)} \bar{J}^{(r)} d\xi^{(r)}. \tag{11}$$

Similar to [Cirak and Ortiz 2001], we use Simpson's rule for integration through the thickness and Gauss quadrature for integration across the surface. However, it is important to choose a sufficient number of Gaussian quadrature points. For Loop subdivision, a single point per triangle is sufficient, [Cirak et al. 2000]. Catmull-Clark basis functions on the other hand are of polynomial degree 3 in regular regions, so integrands involving just the basis functions have total degree 6, and thus require 4×4 quadrature points [Cools 1997]. For nonlinear integrands, more quadrature points may in general be necessary, but we have found 4×4 quadrature points to be sufficient in regular regions. The integration gets more complicated around extra-ordinary vertices. Most recently, this has been studied by [Jüttler et al. 2016] for Loop subdivision and [Wawrzinek and Polthier 2016] for Catmull-Clark schemes.

7 BOUNDARY CONDITIONS

To enforce a supported boundary condition (positions constrained), we add the requirement $\boldsymbol{x}(\xi^1, \xi^2) = \boldsymbol{b}_s(\xi^1, \xi^2)$ for some specified list of parameter space points (ξ^1, ξ^2) along the boundary segment in question. Each mesh element corresponds to a quadrilateral region in parameter space and, in practice, we apply boundary conditions (if any) to the points at the corners of each of these regions. The condition at each such points discretizes as $\boldsymbol{b}_s(\xi^1, \xi^2) = \mathbf{q}_I^T N_I(\xi^1, \xi^2)$, which is a linear constraint on $\mathbf{q}$. For a clamped boundary segment (positions and normals constrained), we start with the supported boundary condition $\boldsymbol{b}_s$, and then additionally enforce normality of a specified direction $\boldsymbol{b}_n(\xi^1, \xi^2)$ to the surface. Let $\boldsymbol{t}(\xi^1, \xi^2)$ denote the surface tangent vector perpendicular to the boundary edge; then $\boldsymbol{t}$ is some linear combination of $\mathbf{a}_1 = \mathbf{q}_I^T N_{I,1}$ and $\mathbf{a}_2 = \mathbf{q}_I^T N_{I,2}$, with coefficients depending on the boundary edge orientation in parameter space. Thus, the necessary condition $\boldsymbol{t}(\xi^1, \xi^2) \cdot \boldsymbol{b}_n(\xi^1, \xi^2) = 0$ amounts to another linear constraint on $\mathbf{q}$. For compatibility, the curve $\boldsymbol{b}_s$ must be perpendicular to the desired normal $\boldsymbol{b}_n$, which (approximately) ensures the surface tangent vector along the boundary edge will be normal to $\boldsymbol{b}_n$ without requiring any additional constraints.

We write the combination of all linear constraints for a given simulation via the requirement $\mathbf{Bq} = \boldsymbol{b}(t)$. Note that the matrix $\mathbf{B}$ of linear constraint coefficients may be precomputed since it depends only on the topology of the parameter space mesh. We incorporate this extra requirement into Equation (10) using a Lagrange multiplier vector $\boldsymbol{\lambda}$ as

$$\begin{aligned} \mathbf{M}\ddot{\mathbf{q}} &= \boldsymbol{f} + \mathbf{B}^T \boldsymbol{\lambda} \\ \mathbf{Bq} &= \boldsymbol{b} \end{aligned} \tag{12}$$

This approach is based on the ideas presented by [Green and Turkiyyah 2004].

8 TIME DISCRETIZATION

For simulation purposes, the ODE in Equation (12) is discretized temporally using backward Euler. The result is a nonlinear system to be solved at each timestep for $(\mathbf{q}^{n+1}, \boldsymbol{\lambda}^{n+1})$:

$$\frac{\mathbf{M}}{h^2}(\mathbf{q}^{n+1} - \mathbf{q}^n - h\dot{\mathbf{q}}^n) = \mathbf{f}\,(\mathbf{q}^{n+1}) + \mathbf{B}^T \boldsymbol{\lambda}^{n+1}$$
$$\mathbf{Bq}^{n+1} = \mathbf{b}^{n+1}.$$

Here, h denotes the time step size. To compute the new configuration, $\mathbf{q}^{n+1}$, we consider the incremental potential associated with the unconstrained time step, [Gast and Schroeder 2014; Kharevych et al. 2006; Radovitzky and Ortiz 1999]. This potential is minimized subject to the constraints from the boundary conditions by way of Newton's method; thus, given a current iterate $(\mathbf{q}^*, \boldsymbol{\lambda}^*)$, we set $(\mathbf{q}^{n+1}, \boldsymbol{\lambda}^{n+1}) = (\mathbf{q}^* + \Delta\mathbf{q}, \boldsymbol{\lambda}^* + \Delta\boldsymbol{\lambda})$ and linearize around $(\mathbf{q}^*, \boldsymbol{\lambda}^*)$ to obtain

$$\frac{\mathbf{M}}{h^2}\,(\mathbf{q}^* + \Delta\mathbf{q} - \mathbf{q}^n - h\dot{\mathbf{q}}^n) = \mathbf{f}\,(\mathbf{q}^*) + \frac{\partial \mathbf{f}}{\partial \mathbf{q}}(\mathbf{q}^*)\Delta\mathbf{q} + \mathbf{B}^T\boldsymbol{\lambda}^* + \mathbf{B}^T\Delta\boldsymbol{\lambda}$$
$$\mathbf{Bq}^* + \mathbf{B}\Delta\mathbf{q} = \mathbf{b}^{n+1}.$$

Denoting the energy Hessian by $\mathbf{K} = -\frac{\partial \mathbf{f}}{\partial \mathbf{q}}$ now allows computation of the Newton step $(\Delta\mathbf{q}, \Delta\boldsymbol{\lambda})$ via the linear system

$$\begin{pmatrix} \mathbf{K} + h^{-2}\mathbf{M} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{q} \\ -\Delta\boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} + \mathbf{B}^T\boldsymbol{\lambda}^* + h^{-2}\mathbf{M}\,(h\dot{\mathbf{q}}^n + \mathbf{q}^n - \mathbf{q}^*) \\ \mathbf{b}^{n+1} - \mathbf{Bq}^* \end{pmatrix}. \tag{13}$$

Our implementation augments the Newton step selection above with a line search procedure to satisfy the strong Wolfe conditions based on the incremental potential. We begin each timestep with the initial guess $\mathbf{q}^* = \mathbf{q}^n$.

9 DERIVATIVES OF ψ

The solution of the discretized system Equation (13) requires evaluation of $\psi = \psi(\mathbf{E}(\mathbf{z}(\mathbf{q})))$ together with its derivatives $\frac{\partial \psi}{\partial \mathbf{q}}$ and $\frac{\partial^2 \psi}{\partial \mathbf{q} \partial \mathbf{q}}$. The evaluation of the derivatives of ψ may be performed via the chain rule given routines for computing the functions $\psi(\mathbf{E})$, $\mathbf{E}(\mathbf{z})$, and $\mathbf{z}(\mathbf{q})$ together with their first and second derivatives. In particular,

$$\begin{aligned}\psi &= \psi(\mathbf{E}(\mathbf{z}(\mathbf{q};\boldsymbol{\xi})))\\ \frac{\partial\psi}{\partial\mathbf{q}} &= \frac{\partial\psi}{\partial\mathbf{E}}\frac{\partial\mathbf{E}}{\partial\mathbf{z}}\frac{\partial\mathbf{z}}{\partial\mathbf{q}}\\ \frac{\partial^2\psi}{\partial\mathbf{q}\partial\mathbf{q}} &= \frac{\partial\mathbf{z}}{\partial\mathbf{q}}^T\frac{\partial\mathbf{E}}{\partial\mathbf{z}}^T\frac{\partial^2\psi}{\partial\mathbf{E}\partial\mathbf{E}}\frac{\partial\mathbf{E}}{\partial\mathbf{z}}\frac{\partial\mathbf{z}}{\partial\mathbf{q}} + \frac{\partial\psi}{\partial\mathbf{E}}\left(\frac{\partial\mathbf{z}}{\partial\mathbf{q}}^T\frac{\partial^2\mathbf{E}}{\partial\mathbf{z}\partial\mathbf{z}}\frac{\partial\mathbf{z}}{\partial\mathbf{q}}\right)\end{aligned}\tag{14}$$

where we have used the fact that $\frac{\partial^2\mathbf{z}}{\partial\mathbf{q}\partial\mathbf{q}} = 0$. To see this, recall that the vector $\mathbf{z}(\mathbf{q};\boldsymbol{\xi})$ is the concatenation of vectors $\boldsymbol{x}_{,\alpha}$ and $\boldsymbol{x}_{,\alpha\beta}$. The result then follows by applying the discretization in Equation (8) to $\boldsymbol{x}_{,\alpha}$ and $\boldsymbol{x}_{,\alpha\beta}$:

$$\begin{aligned}\boldsymbol{x}_{,\alpha} &= \mathbf{q}_I^T N_{I,\alpha} & \boldsymbol{x}_{,\alpha\beta} &= \mathbf{q}_I^T N_{I,\alpha\beta}\\ \frac{\partial\boldsymbol{x}_{,\alpha}}{\partial\mathrm{q}_{Ii}} &= N_{I,\alpha} & \frac{\partial\boldsymbol{x}_{,\alpha\beta}}{\partial\mathrm{q}_{Ii}} &= N_{I,\alpha\beta}\\ \frac{\partial^2\boldsymbol{x}_{,\alpha}}{\partial\mathrm{q}_{Ii}\partial\mathrm{q}_{Jj}} &= 0 & \frac{\partial^2\boldsymbol{x}_{,\alpha\beta}}{\partial\mathrm{q}_{Ii}\partial\mathrm{q}_{Jj}} &= 0.\end{aligned}\tag{15}$$

From this, we also get the values for $\frac{\partial\mathbf{z}}{\partial\mathbf{q}}$.

The strain $\mathbf{E}(\mathbf{z})$ and its $\mathbf{z}$ derivatives are computed from the formulas in §2.1. First, the vectors $\bar{\mathbf{g}}_j$ may be prestored for each quadrature point at the start of the solve and reused thereafter. Recall that $\mathbf{G} = \frac{\partial \boldsymbol{r}}{\partial\boldsymbol{\xi}}^T\frac{\partial \boldsymbol{r}}{\partial\boldsymbol{\xi}}$. Then Equation (2) means that during the solve, evaluation of $\mathbf{E}$ (and its $\mathbf{z}$ derivatives) just requires evaluation of $\mathbf{G}$ (and its $\mathbf{z}$ derivatives). According to Equation (3), we can write

$$\mathbf{G}(\xi^1,\xi^2,\xi^3) = \mathbf{A}(\xi^1,\xi^2) + \xi^3\mathbf{B}(\xi^1,\xi^2) + (\xi^3)^2\mathbf{C}(\xi^1,\xi^2),\tag{16}$$

where

$$\begin{aligned}\mathbf{A}_{\alpha\beta} &= \mathbf{a}_\alpha\cdot\mathbf{a}_\beta & \mathbf{A}_{\alpha 3} &= 0 & \mathbf{A}_{33} &= 1\\ \mathbf{B}_{\alpha\beta} &= \mathbf{a}_\alpha\cdot\mathbf{a}_{3,\beta} + \mathbf{a}_\beta\cdot\mathbf{a}_{3,\alpha} & \mathbf{B}_{\alpha 3} &= 0 & \mathbf{B}_{33} &= 0\\ \mathbf{C}_{\alpha\beta} &= \mathbf{a}_{3,\alpha}\cdot\mathbf{a}_{3,\beta} & \mathbf{C}_{\alpha 3} &= 0 & \mathbf{C}_{33} &= 0\end{aligned}\tag{17}$$

For a given evaluation point ξ^1,ξ^2 on the midsurface, we must evaluate $\mathbf{G}$ and its $\mathbf{z}$ derivatives at several values of ξ^3 (depending on the quadrature rule). We minimize repeated computation by first building $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and their $\mathbf{z}$ derivatives. All required evaluations of $\mathbf{G}$ are then obtained using Equation (16). To compute $\mathbf{z}$ derivatives of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, we first use the chain rule based on Equation (4) to store the often-reused expressions $\frac{\partial\mathbf{a}_{3,\alpha}}{\partial\mathbf{z}}$ and $\frac{\partial^2\mathbf{a}_{3,\alpha}}{\partial\mathbf{z}\partial\mathbf{z}}$. Then $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are differentiated by the product rule on Equation (17). Throughout the above process, all Hessians with respect to $\mathbf{z}$ are stored as sparse symmetric matrices, which significantly improves runtime because several differentiated quantities have extremely simple dependencies on $\mathbf{z}$. The differentiation of more complex terms using the product and chain rules then amounts to a series of sparse matrix-matrix products. Note that $\mathbf{E}$ exhibits a complex (nonquadratic) dependence on $\mathbf{z}$ through the $\mathbf{a}_{3,\alpha}$ terms, and thus on the generalized coordinates $\mathbf{q}$. This is in contrast to more standard volumetric FEM and is the source of greater implementation complexity for Kirchhoff-Love FEM approaches.

The derivatives $\frac{\partial\psi}{\partial\mathbf{E}}$ and $\frac{\partial^2\psi}{\partial\mathbf{E}\partial\mathbf{E}}$ are constructed by the chain rule:

$$\begin{aligned}\frac{\partial\psi}{\partial E_{ab}} &= D_{qa}\frac{\partial\psi}{\partial\tilde{E}_{qr}}D_{rb}\\ \frac{\partial^2\psi}{\partial E_{ab}\partial E_{cd}} &= D_{qa}D_{sc}\frac{\partial^2\psi}{\partial\tilde{E}_{qr}\partial\tilde{E}_{st}}D_{rb}D_{td}.\end{aligned}$$

The derivatives $\frac{\partial\psi}{\partial\tilde{\mathbf{E}}}$ and $\frac{\partial^2\psi}{\partial\tilde{\mathbf{E}}\partial\tilde{\mathbf{E}}}$ depend in turn on derivatives of η_j:

$$\begin{aligned}\eta_j'(x) &= \sum_{i=1}^{d_j}\mu_{ji}(x+1)^{\alpha_{ji}-1}\\ \eta_j''(x) &= \sum_{i=1}^{d_j}\mu_{ji}(\alpha_{ji}-1)(x+1)^{\alpha_{ji}-2}.\end{aligned}$$

10 EVALUATION OF SUBDIVISION SURFACES

To complete the evaluation of the derivatives in the previous section, we need to evaluate the basis functions, N_I, and their derivatives at the quadrature points. As previously mentioned, we use the Catmull-Clark subdivision scheme to define the basis functions, but our method is not inherently tied to this scheme. Any other subdivision scheme with sufficient smoothness could in theory be substituted. In particular, this will work fine with a triangle-based method like the Loop subdivision. However, different subdivision schemes may require different numbers of quadrature points.

In practice, we use the OpenSubdiv library from Pixar, which is what presents a restriction. The current release (version 3.2.0 at the time of this writing) only has sufficient support for Catmull-Clark subdivision surfaces. In particular, OpenSubdiv has not implemented computation of limit surface locations and tangents for any other scheme, which we need for evaluation of the basis functions N_I and their derivatives.

10.1 Partial derivatives

Given a mesh description and an input point (ξ^1,ξ^2) in parameter space, OpenSubdiv is able to efficiently evaluate the basis functions $N_I(\xi^1,\xi^2)$ and their first derivatives $N_{I,\alpha}(\xi^1,\xi^2)$ for all control control points I. As of version 3.2.0, it can also evaluate the second derivatives $N_{I,\alpha\beta}(\xi^1,\xi^2)$. This functionality was added specifically for this project.

To evaluate the basis functions and their first derivative, OpenSubdiv first isolates irregular vertices by adaptively subdividing the control mesh, up to some user-specified maximum level. If the parameter-space input point now lies in a regular patch, OpenSubdiv evaluates using standard bicubic B-spline basis functions for that patch. If the parameter space input point is still in an irregular patch, OpenSubdiv performs an approximate evaluation using a Gregory patch.

Since our energy computations at each timestep always requires evaluation at the same quadrature points we can avoid Gregory patches altogether. To see this, note that for 4-point Gauss quadrature on $[0,1]$, the smallest evaluation point is

$$\frac{1}{2}\left(1-\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}\right)\approx 0.0694 > 0.0625 = 2^{-4}.$$

By symmetry, it follows that for our 4×4 point quadrature rule on $[0,1]\times[0,1]$, all evaluation points are at least distance ≈ 0.0694 away from the boundary. Additionally, note that if we set OpenSubdiv to use s_a adaptive subdivisions, then any remaining irregular patches must lie in the $2^{-s_a} \times 2^{-s_a}$ corner regions of the original square. Thus, if we choose an adaptive subdivion level $s_a \geq 4$, we find that our quadrature points never fall in an irregular patch, which lets us eliminate the Gregory patch approximations as a possible source of error. By the same reasoning, any choice $s_a > 4$ has no effect on the accuracy of the weights. Thus, the current code uses $s_a = 4$ as the number of adaptive subdivision levels. If we wanted to use a different quadrature rule besides the current 4×4 Gauss points per patch, we would still be able to avoid Gregory patch evaluation. However, the simple analysis above would have to be repeated to find the best number of adaptive subdivisions for the new scheme.

11 IMPLEMENTATION

For the actual implementation we separate the computations into a precomputation phase, and steps that have to be repeated within the main simulation loop.

11.1 Precomputations

The mass matrix is assumed constant in our simulations and can be computed directly in the precomputation phase using Equation (11).

For each Newton iteration and/or time step during the simulation we need to evaluate Equation (17) along with the associated $\mathbf{z}$ derivatives. This requires evaluation of $\mathbf{z}$ at all quadrature points, which can be done using Equation (15). We note, however, that the derivatives of the basis functions in these expressions will always be evaluated at the same locations. We precompute these weights using OpenSubdiv.

Finally, obtaining $\mathbf{E}$ from $\mathbf{G}$ requires the basis vectors $\bar{\mathbf{g}}^j$ and our numerical integration requires $\bar{J}$. These depend only on the rest configuration, so we precompute and store them for all quadrature points at the start of the simulation.

11.2 Simulation loop

Inside the main simulation loop we compute the hyperelastic energy Hessian contribution from each quadrature point $\xi^{(r)}$ and then sum these to build the full energy Hessian. To reduce synchronization in our parallel implementation, one thread sums the contributions from all quadrature points within a specified element. The per-element stiffness matrix contributions are built concurrently, and are then added into the final Hessian one at a time. Combining the contributions per element works well because with subdivision surfaces, all $\xi^{(r)}$ within a given element necessarily have the same stencil (that is, will share the same set of nonzero N_I).

The algorithm is summarized Algorithm 1, where the bulk of the work happens in line 5. As described in §9, we can compute the strain $\mathbf{E}(\mathbf{z})$ and its $\mathbf{z}$ derivatives if we know the values, z-gradients, and z-Hessians for $\mathbf{a}_\alpha \cdot \mathbf{a}_\beta$, $\mathbf{a}_\alpha \cdot \mathbf{a}_{3,\beta}$, and $\mathbf{a}_{3,\alpha} \cdot \mathbf{a}_{3,\beta}$.

The terms $\mathbf{a}_\alpha \cdot \mathbf{a}_\beta$ are trivial to compute in terms of $\mathbf{z}$. To clarify the dependence of the remaining terms on $\mathbf{z}$, we introduce the

Algorithm 1 Parallelization Structure for Energy Computation

```
1: Initialize overall system energy/gradient/Hessian as 0.
2: for all quad faces of the control cage do {in parallel}
3:     Initialize face energy/gradient/Hessian contributions as 0.
4:     for all quadrature points in the chosen face do {in serial}
5:         Compute energy/gradient/Hessian at this quadrature
           point and add these to the face's contributions.
6:     end for
7:     Add the face's total accumulated energy/gradient/Hessian
       contributions into the final system. {Use mutex to avoid race
       conditions!}
8: end for
```

notation $\tilde{\mathbf{a}}_3 = \mathbf{a}_1 \times \mathbf{a}_2$ and $m = \|\tilde{\mathbf{a}}_3\|$. Then we have:

$$\begin{aligned}
\mathbf{a}_3 &= \frac{\tilde{\mathbf{a}}_3}{m} \\
\mathbf{a}_{3,\alpha} &= \frac{\tilde{\mathbf{a}}_{3,\alpha}}{m} - \frac{\tilde{\mathbf{a}}_3}{m^2} m_{,\alpha} \\
m_{,\alpha} &= \frac{\tilde{\mathbf{a}}_3 \cdot \tilde{\mathbf{a}}_{3,\alpha}}{m}.
\end{aligned}$$

After some simplification, we can rewrite

$$\begin{aligned}
\mathbf{a}_\alpha \cdot \mathbf{a}_{3,\beta} &= -\frac{\mathbf{a}_{\alpha,\beta} \cdot \tilde{\mathbf{a}}_3}{m} \\
\mathbf{a}_{3,\alpha} \cdot \mathbf{a}_{3,\beta} &= \frac{1}{m^2} \tilde{\mathbf{a}}_{3,\alpha} \cdot \tilde{\mathbf{a}}_{3,\beta} - \frac{1}{m^4} (\tilde{\mathbf{a}}_3 \cdot \tilde{\mathbf{a}}_{3,\alpha})(\tilde{\mathbf{a}}_3 \cdot \tilde{\mathbf{a}}_{3,\beta}).
\end{aligned}$$

Thus, we can compute $\mathbf{E}$ and its $\mathbf{z}$ derivatives by the chain and product rules if we know the values, gradients, and Hessians of the simple 'building block' scalar quantities

$$m \quad \mathbf{a}_\alpha \cdot \mathbf{a}_\beta \quad \mathbf{a}_{\alpha,\beta} \cdot \tilde{\mathbf{a}}_3 \quad \tilde{\mathbf{a}}_3 \cdot \tilde{\mathbf{a}}_{3,\alpha} \quad \tilde{\mathbf{a}}_{3,\alpha} \cdot \tilde{\mathbf{a}}_{3,\beta}.$$

We remark that the strain $\mathbf{E}(\mathbf{z})$ and its $\mathbf{z}$ derivatives are functions of our kinematic assumptions alone. As such, all of the above is equally applicable for other constitutive models.

11.3 Code structure

We now give a slightly simplified but fairly explicit description of the actual implementation.

For storage, we create a struct `Entry` that holds a double, a 15-dimensional vector, and a symmetric 15×15 matrix. These should be interpreted as some scalar value together with its gradient and Hessian with respect to $\mathbf{z}$. Next, we define a second struct `Gdata` that holds $\mathbf{G}$ and its gradient and Hessian. In other words, `Gdata` stores 3 `Entry` structs, one for each nonconstant unique entry in $\mathbf{G}$.

Our low-level implementation consists of 'building-block' functions named `add_a1a1`, `add_a1a2`, and so on going through the list above. These functions accept as input an arbitrary double scalar, plus a reference to `Entry`. They add (scalar)·(specified building block) to the input `Entry`.

The next complexity level consists of functions with names like `addG13Linear`, `addG11Quadratic`, and so on for the other terms. These functions take as input the value of ξ^3 and a reference to `Gdata` giving the current status of $\mathbf{G}$, and they call some combination of building-block functions to add the term described in their name into $\mathbf{G}$. So for example, `addG11Quadratic` adds the value, gradient, and Hessian of $(\xi^3)^2\mathbf{a}_{3,1}\cdot\mathbf{a}_{3,1}$ to the $(1,1)$ `Entry` in our `Gdata` object.

We create a function `potentialEnergyHelper` which accepts a reference to `Entry` (called LocalEnergy), a const reference to `Gdata` (called G), and a const reference to a `ConstitutiveModel` class. It also takes in the precomputed basis vectors $\bar{\mathbf{g}}^j$ and the orthotropy direction matrix $\mathbf{D}$. This function computes $\tilde{\mathrm{E}}_{\alpha\beta}$ and its $\mathbf{z}$ derivatives, and then calls a member function in the `ConstitutiveModel` class to build $\psi(\tilde{\mathrm{E}})$, $\frac{\partial\psi}{\partial\tilde{\mathrm{E}}}$, and $\frac{\partial^2\psi}{\partial\tilde{\mathrm{E}}\partial\tilde{\mathrm{E}}}$. The chain rule is then used to produce the derivatives $\frac{\partial\psi}{\partial\mathbf{z}}$ and $\frac{\partial^2\psi}{\partial\mathbf{z}\partial\mathbf{z}}$, and the local energy contribution ψ and its $\mathbf{z}$ derivatives are added to LocalEnergy. The Hessian $\frac{\partial^2\tilde{\mathrm{E}}_{\alpha\beta}}{\partial\mathbf{z}\partial\mathbf{z}}$ is mostly dense, so we have not made many optimizations here. Instead, that computation essentially follows the steps a rudimentary automatic differentation would use.

Finally, we create the top-level function `potentialEnergy`. This function initializes a `Gdata` object with all entries 0. Then we create the midsurface version of $\mathbf{G}$ by calling all the functions that add ξ^3-constant terms. We pass our `Gdata` object into `potentialEnergyHelper` to add the relevant contributions to the energy and its derivatives. Next, we create the $\xi^3=\frac{\tau}{2}$ version of $\mathbf{G}$ by calling all functions that add ξ^3-linear and ξ^3-quadratic terms and passing in $\xi^3=\frac{\tau}{2}$. Once again, we call `potentialEnergyHelper` to add contributions from this Simpson point. Lastly, we create the $\xi^3=-\frac{\tau}{2}$ version of $\mathbf{G}$ by calling only the functions adding ξ^3-linear terms and passing in $\xi^3=-\tau$. We call `potentialEnergyHelper` a third time and we are done with our energy derivative computation at this Gauss point.

11.4 Optimizing low-level functions

It is possible to store a building-block's gradient and Hessian in some sparse form instead of recomputing the entries each time it is used. This 'precomputation' would still need to be repeated for every Gauss point and for each new energy evaluation, but it can potentially help if the same building block is added many times. The extra step is worth our runtime for some building blocks more than others.

Several of the building-block functions described above require very little computation to add the required terms. For example, the Hessian of $\mathbf{a}_1\cdot\mathbf{a}_1$ has 3 nonzero entries, all of which are constant. Other building-block functions are more complex, but are not called very often. For example, the $\mathbf{a}_{3,1}\cdot\mathbf{a}_{3,2}$ building block is relatively expensive to add, but is only added once since it appears in just one quadratic term.

In our current implementation, the only building block derivatives we prestore are the gradient and Hessian of m, and the gradients of all building blocks involving $\tilde{\mathbf{a}}_3$. We have found this to provide a runtime-efficient balance between the cost of memory allocation and the cost of recomputing the same entries multiple times.

REFERENCES

Fehmi Cirak and Michael Ortiz. 2001. Fully C^1-conforming subdivision elements for finite deformation thin-shell analysis. *Internat. J. Numer. Methods Engrg.* 51, 7 (2001), 813–833. https://doi.org/10.1002/nme.182

Fehmi Cirak, Michael Ortiz, and Peter Schröder. 2000. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *Internat. J. Numer. Methods Engrg.* 47, 12 (2000), 2039–2072. https://doi.org/10.1002/(SICI)1097-0207(20000430)47:12<2039::AID-NME872>3.0.CO;2-1

R. Cools. 1997. Constructing cubature formulae: the science behind the art. *Acta Numerica* 6 (1 1997), 1–54. https://doi.org/10.1017/S0962492900002701

Theodore F. Gast and Craig Schroeder. 2014. Optimization Integrator for Large Time Steps. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, Vladlen Koltun and Eftychios Sifakis (Eds.). The Eurographics Association. https://doi.org/10.2312/sca.20141120

Seth Green and George Turkiyyah. 2004. Second-order accurate constraint formulation for subdivision finite element simulation of thin shells. *Internat. J. Numer. Methods Engrg.* 61, 3 (2004), 380–405. https://doi.org/10.1002/nme.1070

T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. 2005. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* 194, 39-41 (2005), 4135–4195. https://doi.org/10.1016/j.cma.2004.10.008

Bert Jüttler, Angelos Mantzaflaris, Ricardo Perl, and Martin Rumpf. 2016. On numerical integration in isogeometric subdivision methods for PDEs on surfaces. *Computer Methods in Applied Mechanics and Engineering* 302 (2016), 131–146. https://doi.org/10.1016/j.cma.2016.01.005

L. Kharevych, Weiwei Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and M. Desbrun. 2006. Geometric, variational integrators for computer animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '06)*. Eurographics Association, 43–51. https://doi.org/10.2312/SCA/SCA06/043-051

Josef Kiendl, Ming-Chen Hsu, Michael C.H. Wu, and Alessandro Reali. 2015. Isogeometric Kirchhoff-Love shell formulations for general hyperelastic materials. *Computer Methods in Applied Mechanics and Engineering* 291 (2015), 280–303. https://doi.org/10.1016/j.cma.2015.03.010

Quan Long, P. Burkhard Bornemann, and Fehmi Cirak. 2012. Shear-flexible subdivision shells. *Internat. J. Numer. Methods Engrg.* 90, 13 (2012), 1549–1577. https://doi.org/10.1002/nme.3368

R. Radovitzky and M. Ortiz. 1999. Error estimation and adaptive meshing in strongly nonlinear dynamic problems. *Computer Methods in Applied Mechanics and Engineering* 172, 1-4 (1999), 203–240. https://doi.org/10.1016/S0045-7825(98)00230-8

Bernhard Thomaszewski, Markus Wacker, and Wolfgang Straßer. 2006. A Consistent Bending Model for Cloth Simulation with Corotational Subdivision Finite Elements. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 107–116. http://dl.acm.org/citation.cfm?id=1218064.1218079

Roman Vetter, Norbert Stoop, Thomas Jenni, Falk K. Wittel, and Hans J. Herrmann. 2013. Subdivision shell elements with anisotropic growth. *Internat. J. Numer. Methods Engrg.* 95, 9 (2013), 791–810. https://doi.org/10.1002/nme.4536

Anna Wawrzinek, Klaus Hildebrandt, and Konrad Polthier. 2011. Koiter's Thin Shells on Catmull-Clark Limit Surfaces. In *Vision, Modeling, and Visualization (2011)*, Peter Eisert, Joachim Hornegger, and Konrad Polthier (Eds.). The Eurographics Association. https://doi.org/10.2312/PE/VMV/VMV11/113-120

Anna Wawrzinek and Konrad Polthier. 2016. Integration of generalized B-spline functions on Catmull-Clark surfaces at singularities. *Computer-Aided Design* 78 (2016), 60–70. https://doi.org/10.1016/j.cad.2016.05.008 {SPM} 2016.