Computer-Aided Design 174 (2024) 103734

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Simulating Thin Shells by Bicubic Hermite Elements

Xingyu Ni [a,d,1], Xuwen Chen [b,d,1], Cheng yu [c,d], Bin Wang [e,*], Baoquan Chen [b,d]

[a] School of CS, Peking University, China
[b] School of IST, Peking University, China
[c] School of EECS, Peking University, China
[d] State Key Laboratory of General Artificial Intelligence, Peking University, China
[e] State Key Laboratory of General Artificial Intelligence, Beijing Institute for General Artificial Intelligence (BIGAI), China

ARTICLE INFO

Keywords:
Thin-shell simulation
High-order finite elements
Hermite interpolation

ABSTRACT

In this study, we present the bicubic Hermite element method (BHEM), a new computational framework devised for the elastodynamic simulation of thin-shell structures. The BHEM is constructed based on quadrilateral Hermite patches, which serve as a unified representation for shell geometry, simulation, collision avoidance, as well as rendering. Compared with the commonly utilized linear FEM, the BHEM offers higher-order solution spaces, enabling the capture of more intricate and smoother geometries while employing significantly fewer finite elements. In comparison to other high-order methods, the BHEM achieves conforming C^1 continuity for Kirchhoff–Love (KL) shells with minimal complexity. Furthermore, by leveraging the subdivision and convex hull properties of Hermite patches, we develop an efficient algorithm for ray-patch intersections, facilitating collision handling in simulations and ray tracing in rendering. This eliminates the need for laborious remodeling of the pre-existing surface as the conventional approaches do. We substantiate our claims with comprehensive experiments, which demonstrate the high accuracy and versatility of the proposed method.

1. Introduction

The mechanical properties of thin-shell structures are commonly described using the Kirchhoff–Love (KL) theory [1], which neglects transverse shear effects and focuses on the in-plane stretching and lateral bending of the shell's midsurface. Under the KL assumption, the energy density function derived from the elastic strain incorporates second-order derivatives of displacements (as elaborated in Section 3). This requirement necessitates $\mathcal{H}^2$ regularity of the geometric representation of midsurface, which implies C^1 continuity, to ensure a well-defined analysis of the elastic energy.

To effectively address the second-order thin-shell energy, many endeavors have been developed, tested, and honed. In the realm of linear finite-element analysis, which is highly favored within the field of computer graphics, a customary solution is to reformulate the bending energy as a specialized function of the dihedral angle based on the discrete differential geometry principle [2–4]. Nevertheless, this discretized edge-based energy allows bending motion along the common edges of elements only and eventually fails to converge towards the shape operator of the smooth surface at the element interfaces, irrespective of the chosen discretization scheme and mesh resolution [5]. Isogeometric Analysis (IGA) [6,7] is an appealing alternative to attaining high-order continuity solution space. It utilizes various basis functions emanating from computer-aided geometric design (CAGD) in finite element analysis for their global smoothness. One of the earliest depictions of this approach was presented by Cirak et al. [8,9], who devised a finite element formulation grounded in Loop subdivision surfaces for Kirchhoff–Love thin-shell simulation. Loop subdivision scheme can easily represent smooth surfaces of arbitrary topology with polygonal mesh data structures. By employing the same convergent shape function for displacement field interpolation, the subdivision finite element scheme [8,9] requires only nodal displacement degrees of freedom while retaining C^1 continuity across elements, which is necessary for thin-shell simulation. The use of more general non-uniform rational B-splines (NURBS) basis functions in the finite element context was also proposed by Hughes et al. [6] in 2005. NURBS patch is more memory-friendly than subdivision surface, but it needs additional constraints to maintain conforming C^1 continuity within multi-patch models [10]. Since the smooth surface does not pass through the coarse control mesh nodes for both NURBS and subdivision FEM scheme, applying boundary conditions and resolving contacts on control nodes need complicated treatment. Hermite elements stand out for their inherent guarantee of C^1 continuity among patches by incorporating shared derivatives as degrees of freedom. This particular type of element has

* Corresponding author.
E-mail addresses: binwangbuaa@gmail.com (B. Wang), baoquan.chen@gmail.com (B. Chen).
[1] Xingyu Ni and Xuwen Chen contributed equally to this paper.

https://doi.org/10.1016/j.cad.2024.103734
Received 17 April 2024; Accepted 22 May 2024
Available online 27 May 2024

been demonstrated to be advantageous when solving Kirchhoff plate problems [11–14]. Even though the curved Hermite surface exactly goes through the nodal position, existing studies still heavily resort to surface triangulation for collision handling and rendering, diminishing the value of high-order methods.

In this paper, in order to fully leverage the advantages provided by the high-order method, while concurrently minimizing the modifications to the FEM simulation pipeline, we develop a novel framework for thin-shell simulation. The most crucial aspect of our framework lies in the utilization of a unified bicubic-Hermite-patches-based representation, which serves as the foundation of midsurface geometric modeling, dynamics simulation, and ray-tracing rendering purposes. Specifically, we present three main contributions to achieving this goal.

First, we develop a C^1-continuous finite element solution for dynamic simulation of Kirchhoff–Love thin shells, dubbed BHEM (Bicubic Hermite Element Method). We listed the derivation process of the governing equations and its weak form (Section 4) of KL thin shells from the first principles of continuum mechanics. In the Hermit polynomial space, the discretized form of the governing equations, along with the gradient and the Hessian matrix (see the supplementary material) of the hyperelastic energy, are provided to facilitate a seamless implicit solve (Section 6.1). Furthermore, the BHEM also incorporates a tailored Hessian matrix which entails much less computational overhead, to cater to applications with restricted time constraints.

Further consolidating the integration between geometry representation and simulation, and bypassing mesh-based collision detection, our second main contribution is a BH-surface intersection detection algorithm that serves for both CCD and rendering tasks (Section 5). In our method, the bicubic Hermite surfaces are transformed into their equivalent Bézier form. Based on the convex hull property of the Bézier surface, a bounding-volume-hierarchy (BVH) tree is constructed through dynamic subdivision of the surface and then pruned using Newton's method to enhance the computation efficiency. Our method ensures the discovery of the first intersection point in both static and dynamic settings. It can also be extended to other spline-based surface intersection detection, such as rational Bézier patches and NURBS, which we believe is a critical missing piece of current IGA research.

Last but not least, we conduct a broad array of experiments meticulously designed to showcase the fidelity and efficiency of the BHEM framework. We first test our method under typical quasi-static settings. The simulation results are highly consistent with the theoretical solutions. Compared with linear FEMs, BHEM can capture rich geometric features with much fewer DoFs, better convergence speed, and less time cost. Furthermore, we demonstrate the applicability of the framework by applying it to a variety of graphics scenarios with complex collisions and diverse boundary conditions. We emphasize the smoothness and highlight the characteristic wrinkles and folds of cloth geometries during dynamic simulations in all these experiments.

2. Related work

2.1. Thin-shell simulation

Physics-based thin shell simulation, such as cloth [15–20], paper [21–24], and skin [25,26], is a long-standing topic in computer graphics owing to their visually appealing geometry and desirable mechanical properties. Following the seminal work of Terzopoulos et al. [27], a series of discrete constitutive models [3–5,28] have been developed for the elastic strain energy by applying geometric operators over the piecewise-linear surface, but with the bending energy being a non-integrable function of the dihedral angle. Researchers have taken many measures to circumvent the issue, such as subdivision FEM [8,29–31] and discontinuous Galerkin (DG) FEM [32], but both with numerical issues due to their unavoidable special treatments to approximate or meet the continuity requirement of a Kirchhoff–Love shell.

Another branch of the solution follows the route of isogeometric analysis, in which the basis functions that express geometric shapes are utilized in FE analysis for the physical field interpolation [10,33]. After the first attempt [34] for using NURBS surface to represent a deformable thin object, NURBS has been widely exploited in cloth motion to describe the characteristic wrinkles or folds [10] and in volumetric object simulation [35]. However, due to its complex math formulation, the NURBS surfaces have involved derivation and high computational cost in graphics applications.

Studies have explored Hermite-interpolated surfaces on triangular elements [36,37], quadrangular elements [38–40], and hybrid elements [41] in engineering and numerical analysis. Triangular Hermite elements use higher-order polynomials as basis functions so as to achieve C^1 continuity, leading to more computational cost. By contrast, quadrangular Hermite elements can have simpler but accurate expressions, though fall short in geometric flexibility. In graphics, Hermite patches were originally adopted for freeform surface modeling [42,43], but they lack applications in deformable objects. In this work, we try to establish a complete framework that enables dynamic simulation and rendering of Hermite-interpolated surfaces.

2.2. Rendering of parametric surfaces

The calculation of ray tracing a parametric surface is to find the *first* intersection point between a ray and a surface, which is equivalent to solving a system of nonlinear equations obtained by combining the parametric equations of the two geometric primitives. A commonly adopted approach to ray-trace a high-order parametric surface is to triangulate the surface and render the approximated mesh. But to reserve the special advantages of high-order representation, it is beneficial to explore an efficient and robust rendering method that solves the problem directly on the parametric surface.

Existing studies have developed various numerical methods for this problem. Resultant elimination [44,45] transforms the problem into finding the minimum of a uni-variant polynomial, which works but requires heavy numerical computation with complex situational discussion, thus lack of robustness. Newton's method [46–51] is a powerful tool that can deliver an accurate solution with a fast convergence speed, but it depends severely on the initial conditions and has no guarantee for a correct solution since it only solves the problem locally. Also, the system would become ill-conditioned if a ray intersects the surface tangentially.

To solve for a global solution robustly, researchers have introduced subdivision methods [52–55] that recursively prune the parameter domain until convergence. Bézier clipping [52,56,57] is a classical subdivision method that utilizes the convex-hull property of Bézier patches. It has been widely used for ray-tracing any surfaces that can be converted into rational Bézier patches, including B-splines [57] and NURBS [58]. Meanwhile, subdivision-based methods often expend more time and space cost than Newton's methods. Therefore, a popular idea is to first conduct subdivision methods to construct a BVH for culling and getting a rough solution as an initial guess and then use Newton's descent to get a final answer [47–49,51]. Simple geometric primitives, such as Chebyshev boxing/sphere [51,59], Convex hulls [49,60], and axis aligned bounding box [47,48] are the popular choices for BVH representations.

2.3. Collision detection of parametric surfaces

Continuous collision detection (CCD) for high-order meshes or parametric surfaces is mostly conducted by detecting collisions between two approximated triangle meshes [61], due to the maturity of traditional CCD techniques for linear FEM. As can be imagined, this would introduce inevitable false positives and negatives where the original surface sinks or extrudes from the linear mesh.

Conducting DCD/CCD directly on parametric surfaces would result in solving a 4D/5D nonlinear system with a complex solution manifold, which is usually reduced to point pairs in implementation [62,63]. Similar to the ray-tracing problem, solutions for CCD are always established on either subdivision methods [62–64] to recursively solve for a global optimum or Newton's methods [10,63,65] to directly solve for a local optimum. Researchers need to get an appropriate initial guess for Newton's method and cull unnecessary checks via assisting strategies including tessellation [10], BVH [66], spatial partition [67], etc. Recently, another promising approach proposed by Zhang et al. [68,69] modeled the CCD problem between polynomial surfaces as a sum-of-squares programming (SOSP) so bypassed linearizing high-order surfaces. This method includes an auxiliary hyperparameter in SOSP that directly determines the certificate of the exact solution, which requires a trade-off between efficiency and effectiveness.

3. Thin-shell theory

To make the paper self-contained, in this section, we will briefly review the thin-shell theory based upon the Kirchhoff–Love assumption, which generally follows the work of Cirak et al. [8] but is not restricted to linearized deformation.

Conventions and notations. The Einstein summation convention is followed, where an index variable appears twice in a single term implying summation of that term over all the values of the index. We further assume that the value of an index denoted by a lowercase Latin letter (e.g., i, j, and k) ranges over the set $\{1,2,3\}$, while that denoted by a lowercase Greek letter (e.g., α, β, and γ) ranges over the set $\{1,2\}$. Besides, indices appearing after commas imply partial derivatives.

3.1. Geometries

We begin by considering a *midsurface* $\Omega \subset \mathbb{R}^3$. As a surface, Ω is parameterized with curvilinear coordinates (ξ^1,ξ^2). The possible values of these coordinates form a parameter space ω, and the parameterization is then given by a mapping $\boldsymbol{x} : \omega \to \Omega$ such that the following properties hold:

- At each point of Ω, the two partial derivatives $\partial\boldsymbol{x}/\partial\xi^1$ and $\partial\boldsymbol{x}/\partial\xi^2$ exist and are linearly independent;
- As a function $\omega \to \mathbb{R}^3$, $\boldsymbol{x}(\xi^1,\xi^2)$, as well as its first- and second-order derivatives, is square-integrable.

The former property allows the definition of the unit normal vector $\boldsymbol{a}_3 = \boldsymbol{a}_1 \times \boldsymbol{a}_2/\|\boldsymbol{a}_1 \times \boldsymbol{a}_2\|$, in which $\boldsymbol{a}_\alpha$ denotes $\partial\boldsymbol{x}/\partial\xi^\alpha$, while the latter property is required by analysis of elastic energy.

According to the Kirchhoff–Love assumption, the midsurface is extruded by a constant distance $h/2$ both along and opposite to the surface normal direction, which forms the volume of a h-thick shell. With ω^h defined as $\omega\times[-h/2,+h/2]$, the extrusion of Ω is described by a function $\boldsymbol{r} : \omega^h \to \Omega^h$ as follows:

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

which lays the foundation of a thin-shell geometry.

To analyze shell deformation, the geometric difference between deformed and undeformed (reference) configurations needs evaluation. We assume that each point of ω^h always maps to the same material point, and use symbols with overbars to indicate quantities in the reference configuration. A function $\bar{\boldsymbol{r}} : \omega^h \to \bar{\Omega}^h$ can be similarly defined by

$$\bar{\boldsymbol{r}}(\xi^1,\xi^2,\xi^3) = \bar{\boldsymbol{x}}(\xi^1,\xi^2) + \xi^3\bar{\boldsymbol{a}}_3(\xi^1,\xi^2), \quad -\frac{h}{2} \le \xi^3 \le +\frac{h}{2}. \tag{2}$$

3.2. Strains

Given the parametric description of shell geometry, we acquire the tangent basis vectors of Ω^h as follows:

$$\boldsymbol{g}_i = \frac{\partial\boldsymbol{r}}{\partial\xi^i} = \begin{cases} \boldsymbol{a}_\alpha + \xi^3\boldsymbol{a}_{3,\alpha}, & i=\alpha<3, \quad \text{(a)} \\ \boldsymbol{a}_3, & i=3. \quad \text{(b)} \end{cases} \tag{3}$$

Dot products of these vectors result in covariant components of the metric tensor. To be specific, $g_{ij} = \boldsymbol{g}_i \cdot \boldsymbol{g}_j$ is formulated by

$$g_{ij} = \begin{cases} a_{\alpha\beta} - 2b_{\alpha\beta}\xi^3 + c_{\alpha\beta}(\xi^3)^2, & i=\alpha<3 \wedge j=\beta<3, \quad \text{(a)} \\ 1, & i=j=3, \quad \text{(b)} \\ 0, & \text{otherwise}, \quad \text{(c)} \end{cases} \tag{4}$$

where $a_{\alpha\beta} = \boldsymbol{a}_\alpha \cdot \boldsymbol{a}_\beta$, $b_{\alpha\beta} = \boldsymbol{a}_{\alpha,\beta} \cdot \boldsymbol{a}_3$, and $c_{\alpha\beta} = \boldsymbol{a}_{3,\alpha} \cdot \boldsymbol{a}_{3,\beta}$ correspond to the 1, 2, and 3 fundamental forms of S, respectively.

With $\{\bar{\boldsymbol{g}}_i\}$ being the basis of the tensor space, the *Green–Lagrange strain* is defined as half the difference between the metric tensors in the deformed and undeformed configurations:

$$E_{ij} = \frac{1}{2}(g_{ij} - \bar{g}_{ij}). \tag{5}$$

As will be readily seen, E_{ij} can be expanded as

$$\begin{cases} E_{\alpha\beta} = A_{\alpha\beta} - 2B_{\alpha\beta}\xi^3 + C_{\alpha\beta}(\xi^3)^2, & \text{(a)} \\ E_{3\alpha} = E_{\alpha 3} = 0, & \text{(b)} \\ E_{33} = 0, & \text{(c)} \end{cases} \tag{6}$$

followed by definitions $A_{\alpha\beta} = (a_{\alpha\beta} - \bar{a}_{\alpha\beta})/2$, $B_{\alpha\beta} = (b_{\alpha\beta} - \bar{b}_{\alpha\beta})/2$, and $C_{\alpha\beta} = (c_{\alpha\beta} - \bar{c}_{\alpha\beta})/2$. Note that the 0-order term $A_{\alpha\beta}$ in Eq. (6a) represents the *membrane strain*, while the other terms $B_{\alpha\beta}\xi^3 + C_{\alpha\beta}(\xi^3)^2$ characterize the *curvature strain*. With the linear stain assumed in the thickness direction, the highest-order term of Eq. (6a), specifically $C_{\alpha\beta}(\xi^3)^2$, is omitted in the following calculation.

3.3. Energies

As suggested by previous studies [8,70], the formula of the elastic strain energy V_e is derived from the Green–Lagrange strain based upon a St. Venant–Kirchhoff constitutive model, which is given in the form of areal density $\bar{\mathcal{V}}_\mathrm{e}$ as

$$\bar{\mathcal{V}}_\mathrm{e} = \frac{\mathrm{d}V_\mathrm{e}}{\mathrm{d}\bar{\Omega}} = \left(A_{\alpha\beta}A_{\gamma\delta}h + \frac{1}{3}B_{\alpha\beta}B_{\gamma\delta}h^3\right)\bar{H}^{\alpha\beta\gamma\delta}, \tag{7}$$

in which $\bar{H}^{\alpha\beta\gamma\delta}$ is defined by

$$\bar{H}^{\alpha\beta\gamma\delta} = \frac{\lambda}{2}\bar{a}^{\alpha\beta}\bar{a}^{\gamma\delta} + \mu\bar{a}^{\beta\gamma}\bar{a}^{\alpha\delta}. \tag{8}$$

Here, $\bar{a}^{\alpha\beta}$ is a contravariant tensor component, which can be calculated by the matrix inversion $(\bar{a}^{\alpha\beta})_{2\times2} = (\bar{a}_{\alpha\beta})^{-1}_{2\times2}$. λ and μ, known as the first and the second *Lamé parameters*, are deduced from Young's modulus Y and Poisson's ratio ν as $\lambda = Y\nu/(1-\nu^2)$ and $\mu = Y/2(1+\nu)$, respectively.

The motion of a thin shell is also influenced by the kinetic energy T. With $\bar{\rho}$ denoting mass density, similar to V_e, T is also written in the form of areal density $\bar{\mathcal{T}}$ as

$$\bar{\mathcal{T}} = \frac{\mathrm{d}T}{\mathrm{d}\bar{\Omega}} = \frac{1}{2}\bar{\rho}h(\dot{\boldsymbol{x}} \cdot \dot{\boldsymbol{x}}), \tag{9}$$

which is derived in the material space by integrating energy density along the thickness and ignoring high-order infinitesimals.

3.4. Equations of motion

Finally, the equations of motion for a thin shell can be constructed by analyzing the interchange of energy. Since the volumetric shell is replaced by its midsurface, we should also reduce any force that applies to the shell into surface force by integration. Suppose that the areal density of external force applied on $\bar{\Omega}$ is $\bar{\boldsymbol{f}}$, and the linear density on $\bar{\Gamma} = \partial\bar{\Omega}$ is $\bar{\boldsymbol{t}}$, D'Alembert's principle states that for any virtual deformation $\delta\boldsymbol{x}$,

$$\iint_{\bar{\Omega}} \left(\delta\bar{\mathcal{T}} + \delta\bar{\mathcal{V}}_\mathrm{e}\right)\mathrm{d}\bar{\Omega} = \iint_{\bar{\Omega}} \bar{\boldsymbol{f}} \cdot \delta\boldsymbol{x}\,\mathrm{d}\bar{\Omega} + \int_{\bar{\Gamma}} \bar{\boldsymbol{t}} \cdot \delta\boldsymbol{x}\,\mathrm{d}\bar{\Gamma}, \tag{10}$$

in which $\delta\bar{\mathcal{T}} = \bar{\rho}h\ddot{\boldsymbol{x}} \cdot \delta\boldsymbol{x}$ can be interpreted as the virtual work done by the inertia force, and the variation of $\bar{\mathcal{V}}_\mathrm{e}$ can be expanded as

$$\delta\bar{\mathcal{V}}_\mathrm{e} = \left(hA_{\gamma\delta}\,\delta a_{\alpha\beta} + \frac{1}{3}h^3 B_{\gamma\delta}\,\delta b_{\alpha\beta}\right)\bar{H}^{\alpha\beta\gamma\delta}, \tag{11}$$

owing to the exchange symmetry of indices.

It is noteworthy that the surface area element $\mathrm{d}\bar{\Omega}$ can be expressed in the parameter space by

$$\mathrm{d}\bar{\Omega} = \sqrt{\bar{a}}\,\mathrm{d}\xi^1\mathrm{d}\xi^2, \tag{12}$$

$$\bar{a} = (\bar{\boldsymbol{a}}_1 \times \bar{\boldsymbol{a}}_2)^2 = \bar{a}_{11}\bar{a}_{22} - \bar{a}_{12}\bar{a}_{21}. \tag{13}$$

4. The bicubic Hermite element method

4.1. Geometric discretization

To numerically analyze the statics and dynamics of a thin shell, we divide its midsurface into a collection of bicubic Hermite patches that share information via nodes, where the bicubic Hermite interpolation is used to naturally maintain the C^1 smoothness and square-integrability (Section 3.1) in surface reconstruction.

As shown in Fig. 5, each patch corresponds to an axis-aligned rectangle in the parameter space, and each rectangle side is entirely shared by two adjacent patches. In the context of such discretization, with the function value and its derivatives (i.e., $\boldsymbol{x}$, $\partial\boldsymbol{x}/\partial\xi^1$, $\partial\boldsymbol{x}/\partial\xi^2$, and $\partial^2\boldsymbol{x}/\partial\xi^1\partial\xi^2$) stored at nodes, any point on a single patch satisfying $\xi^1 \in [\xi^1_{\min}, \xi^1_{\max}]$ and $\xi^2 \in [\xi^2_{\min}, \xi^2_{\max}]$ can be expressed with the piecewise bicubic Hermite interpolation (see the supplementary material), such that the midsurface can be parameterized by

$$\boldsymbol{x}(\xi^1,\xi^2) = \sum_{I=1}^{N} \boldsymbol{\Phi}^I(\xi^1,\xi^2)\,\boldsymbol{q}_I, \tag{14}$$

in which $\boldsymbol{q}_I \in \mathbb{R}^3$ $(I = 1, 2, 3, \ldots, N)$, treated as *generalized coordinates*, denotes a value or derivative that is stored at nodes, and each $\boldsymbol{q}_I$ corresponds to a shape function $\boldsymbol{\Phi}^I(\xi^1,\xi^2)$. These shape functions have compact supports, so the summand $\boldsymbol{\Phi}^I\boldsymbol{q}_I$ takes nonzero values only if $\boldsymbol{q}_I$ is stored at the 4 nodes of the patch that (ξ^1,ξ^2) lies in. Typically, the number of such $\boldsymbol{q}_I$ is 16.

The bicubic Hermite patches can be readily used to discretize surfaces that are homeomorphic to a rectangle by embedding them in a Cartesian grid and taking grid cells as patches (Fig. 5). More complex surfaces, e.g., cylinders and torus, can also be divided into BH patches with the help of continuity boundary conditions (Section 4.2), which is illustrated in Figs. 6(a) and 6(b).

4.2. Boundary conditions

Continuity boundary conditions. For BHEM simulations, it is significant to incorporate the continuity boundary conditions in order to complete geometric discretization. As an example, in the discretization of a cylinder (Fig. 6(a)), a node with an azimuthal coordinate of 2π is identical to that with an azimuthal coordinate of 0. Whenever we encounter such a situation, only one of the overlapping nodes needs maintenance. Any calculation that involves these nodes is referred to as the maintained one.

Positional constraints. The BHEM prescribes two categories of positional constraints, namely Dirichlet and Neumann boundary conditions, which respectively constrain $\boldsymbol{x}(\xi^1,\xi^2)$ and its first-order derivatives at specific points. When at nodes, these constraints can be simply realized by removing their corresponding generalized coordinates from the solution variables and taking their influence back to the governing equations as a given term. Based on this, a constraint imposed on an element's whole boundary curve can be realized through the combination of Dirichlet constraints and Neumann constraints along the boundary tangent direction for every node on the boundary. Furthermore, to apply the constraints at arbitrary points of the midsurface, Lagrange multipliers should be introduced, as done in conventional finite element methods.

4.3. Governing equations

Now we consider discretizing Eq. (10) using the variational method based on the BH patches. Taking $\delta\bar{\mathcal{V}}_\mathrm{e} = \partial\bar{\mathcal{V}}_\mathrm{e}/\partial\boldsymbol{q}_I \cdot \delta\boldsymbol{q}_I$, substituting Eq. (14) into Eq. (10) yields

$$\sum_{I=1}^{N}\left(\iint_{\bar{\Omega}}\left(\boldsymbol{\Phi}^I\bar{\boldsymbol{f}}^* - \frac{\partial\bar{\mathcal{V}}_\mathrm{e}}{\partial\boldsymbol{q}_I}\right)\mathrm{d}\bar{\Omega} + \int_{\bar{\Gamma}}\boldsymbol{\Phi}^I\bar{\boldsymbol{t}}\,\mathrm{d}\bar{\Gamma}\right)\cdot\delta\boldsymbol{q}_I = 0, \tag{15}$$

$$\bar{\boldsymbol{f}}^* = \bar{\boldsymbol{f}} - \bar{\rho}h\sum_{J=1}^{4N}\boldsymbol{\Phi}^J\ddot{\boldsymbol{q}}_J, \tag{16}$$

which indicates that Eq. (10) is always true for any virtual deformation interpolated through Eq. (14). Due to the arbitrariness of $\delta\boldsymbol{q}_I$, every coefficient of $\delta\boldsymbol{q}_I$, i.e., the terms in the outermost parentheses of Eq. (15), must equal to zero. Thus, for any I $(1 \le I \le N)$, the following equations hold:

$$\iint_{\bar{\Omega}}\boldsymbol{\Phi}^I\bar{\boldsymbol{f}}\,\mathrm{d}\bar{\Omega} + \int_{\bar{\Gamma}}\boldsymbol{\Phi}^I\bar{\boldsymbol{t}}\,\mathrm{d}\bar{\Gamma} - \iint_{\bar{\Omega}}\frac{\partial\bar{\mathcal{V}}_\mathrm{e}}{\partial\boldsymbol{q}_I}\,\mathrm{d}\bar{\Omega} - \sum_{J=1}^{N}M^{IJ}\ddot{\boldsymbol{q}}_J = \boldsymbol{0}, \tag{17}$$

where each coefficient of the mass matrix is defined as

$$M^{IJ} = \iint_{\bar{\Omega}}\bar{\rho}h\boldsymbol{\Phi}^I\boldsymbol{\Phi}^J\,\mathrm{d}\bar{\Omega} = \iint_{\Omega}\bar{\rho}h\boldsymbol{\Phi}^I\boldsymbol{\Phi}^J\sqrt{\bar{a}}\,\mathrm{d}\xi^1\mathrm{d}\xi^2. \tag{18}$$

These constitute the governing equations of a dynamic BHEM shell. For static analysis, the last term in the left-hand side of Eq. (17) is omitted.

Properties of the mass matrix. Due to the compact support of the shape functions $\boldsymbol{\Phi}$, a coefficient M^{IJ} is nonzero only when the affected scopes of $\boldsymbol{\Phi}^I$ and $\boldsymbol{\Phi}^J$ overlap, i.e., the corresponding generalized coordinates of $\boldsymbol{\Phi}^I$ and $\boldsymbol{\Phi}^J$ belong to the nodes of the same patch. This implies the sparsity of the mass matrix. Furthermore, with $\boldsymbol{\Phi}^I$ being a polynomial of no more than degree 3, $\boldsymbol{\Phi}^I\boldsymbol{\Phi}^J$ reaches a sixth-order at most in each parametric dimension. Thus we conclude that M^{IJ} can be calculated precisely without much effort, given that $\sqrt{\bar{a}}$ is also a polynomial concerning the position.

External forces. We briefly introduce how to calculate the external force term in Eq. (17), taking $\iint_{\bar{\Omega}}\boldsymbol{\Phi}^I\bar{\boldsymbol{f}}\,\mathrm{d}\bar{\Omega}$ as an example. For an areal force, the integration cannot be avoided. Typical examples are gravity force (given by $\bar{\boldsymbol{f}}_\mathrm{grav} = \bar{\rho}h\boldsymbol{g}$ with $\boldsymbol{g}$ standing for the gravity acceleration) and pressure force (given by $\bar{\boldsymbol{f}}_\mathrm{press} = p\boldsymbol{n}$ with p denoting the magnitude of pressure). On the other hand, a point force $\bar{\boldsymbol{f}}_\mathrm{pt}$ exerted at an arbitrary point can be reformulated as an areal force multiplied by a Dirac δ function. Given the point $\boldsymbol{x}_\mathrm{pt}(\xi^1_\mathrm{pt},\xi^2_\mathrm{pt})$, the closed integral form of $\bar{\boldsymbol{f}}_\mathrm{pt}$ can be calculated by

$$\iint_{\bar{\Omega}}\boldsymbol{\Phi}^I\bar{\boldsymbol{f}}_\mathrm{pt}\delta(\boldsymbol{x}_\mathrm{pt})\,\mathrm{d}\bar{\Omega} = \sum_I\boldsymbol{\Phi}^I(\xi^1_\mathrm{pt},\xi^2_\mathrm{pt})\bar{\boldsymbol{f}}_\mathrm{pt}, \tag{19}$$

which implies that a point force only influences the 16 generalized coordinates of its nearest 4 nodes.

5. Ray-patch intersection detection

We can equivalently transform a BH surface into a Bézier surface by regarding the cubic polynomials as linear combinations of *Bernstein basis polynomials* of degree 3:

$$\boldsymbol{x}(\xi^1,\xi^2) = \sum_{i=0}^{3}\sum_{j=0}^{3}B^i(\xi^1)B^j(\xi^2)\boldsymbol{p}_{ij}, \tag{20}$$

where $B^i(x)$ is defined as $\binom{3}{i}x^i(1-x)^{3-i}$ and $\boldsymbol{p}_{ij} \in \mathbb{R}^3$ denote the control points. A Bézier form provides us with a strong property that a surface lies completely within the convex hull of its control points, and thus also completely within the bounding box of them in any given Cartesian coordinate system, which lays the foundation of our ray-surface intersection detection algorithms.

The surface intersection is an important geometric operation in CAGD. We mainly focus on finding the intersection point between a ray and a surface in this paper. According to whether the surface moves during ray propagation, two kinds of intersections are detected.

5.1. Static ray–patch intersection

Given a ray defined as

$$\boldsymbol{x}_{\text{ray}}(\tau) = \boldsymbol{x}_0 + \tau \boldsymbol{d}, \tag{21}$$

where $\boldsymbol{d}$ represents the direction and τ denotes the (pseudo) time, the goal of intersection tests is to find (ξ^1, ξ^2) and minimum $\tau > 0$ such that $\boldsymbol{x}_{\text{ray}}(\tau) = \boldsymbol{x}(\xi^1, \xi^2)$ holds. The convex-hull property of Bézier surfaces implies that for any ξ^1, ξ^2, and τ satisfying this intersection equation, the following inequalities hold:

$$\min_{i,j\in\{0,1,2,3\}} \left\{\boldsymbol{p}_{ij} \cdot \hat{\boldsymbol{e}}_k\right\} \le (\boldsymbol{x}_0 + \tau \boldsymbol{d}) \cdot \hat{\boldsymbol{e}}_k \le \max_{i,j\in\{0,1,2,3\}} \left\{\boldsymbol{p}_{ij} \cdot \hat{\boldsymbol{e}}_k\right\}. \tag{22}$$

Here, $\hat{\boldsymbol{e}}_k$ ($k \in \{1, 2, 3\}$) denotes the kth vector of the standard basis. These inequalities correspond to an intersection test between the ray and an axis-aligned bounding box (AABB). By solving Eq. (22) for τ, we can obtain an interval $[\tau_{\min}, \tau_{\max}]$ or determine that no $\tau > 0$ satisfies all the inequalities, indicating that the ray does not intersect the current surface.

When the former case is encountered after solving Eq. (22), a divide-and-conquer strategy is employed. The Bézier surface is split into four smaller Bézier surfaces using De Casteljau's algorithm [71] at specific parameter coordinates. This recursive process allows for ray-AABB tests to be performed, enabling the elimination of surfaces that cannot be intersected by the ray. This continues until either all surfaces are discarded or an arbitrarily small interval of τ is obtained.

The above subdivision-based algorithm is trivial but effective and robust, converging linearly to the accurate intersection point. To minimize the number of surfaces requiring collision checks, the selection of the next surface is improved by using a min-heap of candidate surfaces. Every time a new surface is generated, it is inserted into the heap with $\tau_{\min}$ as the key. When moving to another surface, the top of the heap is chosen. The first surface that satisfies the termination condition provides an accurate approximation of the minimum τ among all the intersection points.

Finally, Newton's method is employed to determine where to subdivide surfaces by directly solving $\boldsymbol{x}(\xi^1, \xi^2) = \boldsymbol{x}_{\text{ray}}(\tau)$. The central parameter coordinates of a surface serve as the initial guess for Newton's method during the subdivision process. Within a fixed number of iterations, if Newton's method converges to $\xi^1 \in [u_{\min}, u_{\max}]$ and $\xi^2 \in [v_{\min}, v_{\max}]$, we obtain an intersection point and use the corresponding τ to update the current optimal solution τ_{opt} if τ is smaller and positive. Note that this intersection point may not be the closest one. We remove it from the search space by abandoning the neighborhood of (ξ^1, ξ^2) in the parameter space and subdividing the other region as illustrated in Fig. 9(b). In the case Newton's method fails to converge, the surface is split from its midpoint as shown in (Fig. 9(a)). The entire algorithm is summarized in Alg. 1.

5.2. Dynamic ray–patch intersection

Assuming constant velocity within a single time step Δt, a time-varying Bézier surface is given by

$$\boldsymbol{x}(\xi^1, \xi^2, t) = \sum_{i=0}^{3}\sum_{j=0}^{3} B_i^3(\xi^1) B_j^3(\xi^2)(\boldsymbol{p}_{ij} + t\dot{\boldsymbol{p}}_{ij}), \tag{23}$$

where $\boldsymbol{p}_{ij}$ and $\dot{\boldsymbol{p}}_{ij}$ are the initial position and velocity of the (i, j)-th control point, and $t \in [0, \Delta t]$ indicates the time. The trajectory of a moving point can be depicted as a parameterized ray as

$$\boldsymbol{x}_{\text{p}}(t) = \boldsymbol{x}_{\text{p0}} + t\dot{\boldsymbol{x}}_{\text{p}}, \tag{24}$$

where $\boldsymbol{x}_{\text{p0}}$ and $\dot{\boldsymbol{x}}_{\text{p}}$ denote the initial position and velocity.

According to the convex-hull property, a necessary condition for the intersection to occur at the moment t is that the point lies inside the axis-aligned bounding box of the surface at that moment, which can be written as

$$\min_{i,j} \left\{(\boldsymbol{p}_{ij} + t\dot{\boldsymbol{p}}_{ij}) \cdot \hat{\boldsymbol{e}}_k\right\} \le (\boldsymbol{x}_{\text{p0}} + t\dot{\boldsymbol{x}}_{\text{p}}) \cdot \hat{\boldsymbol{e}}_k \le \max_{i,j} \left\{(\boldsymbol{p}_{ij} + t\dot{\boldsymbol{p}}_{ij}) \cdot \hat{\boldsymbol{e}}_k\right\}, \tag{25}$$

with $i, j \in \{0, 1, 2, 3\}$. By solving Eq. (25) for t similar to the approach described in Section 5.1 for τ, we can employ a similar subdivision-based algorithm.

For each inequality on the left-hand side, the result is equivalent to the union of the intervals solved from the inequalities

$$(\boldsymbol{p}_{ij} + t\dot{\boldsymbol{p}}_{ij}) \cdot \hat{\boldsymbol{e}}_k \le (\boldsymbol{x}_{\text{p0}} + t\dot{\boldsymbol{x}}_{\text{p}}) \cdot \hat{\boldsymbol{e}}_k, \tag{26}$$

where i and j ranges in $\{0, 1, 2, 3\}$. The same interpretation holds for the right-hand side of Eq. (25). The solution of Eq. (25) is the intersection of the two unions, which can be found using segment trees or greedy algorithms. If no feasible solution exists within $[0, \Delta t]$, no intersection is detected during the time step. Otherwise, we can utilize the divide-and-conquer framework described in Section 5.1 to recursively detect possible intersections. Similarly, Newton's method can be employed for acceleration.

ALGORITHM 1: Static Intersecting Detection

Input: The ray $\boldsymbol{x}_{\text{ray}}$ and the surface.
Output: The nearest intersection point (ξ^1, ξ^2) or **null** indicating no intersecion.

```
τ_opt ← ∞;
Construct a min heap;
if IntersectsAABB(x_ray, the whole surface) then Push the
  whole surface into the heap with its τ_min as the key;
else return null;
while the heap is not empty do
    The current surface ← the top of the heap;
    if τ_min > τ_opt then break;
    if the surface is small enough then return the middle
      parameter coordinates of the surface ;
    Find an intersection point by Newton's method;
    if converged then
        Update τ_opt and (ξ¹_opt, ξ²_opt);
        Subdivide the surface by the scheme in Fig. 9(b);
    else
        Subdivide the surface by the scheme in Fig. 9(a);
    end
    foreach subdivided surface do
        if IntersectsAABB(x_ray, the subdivided surface) then
            Push the surface into the heap with its τ_min as the key;
        end
    end
end
if τ_opt ≠ +∞ then return (ξ¹_opt, ξ²_opt);
else return null;
```

6. Implementation

6.1. Implicit BHEM solver

Having established spatially discretized equations of motion (Section 4.3), we here discuss how to discretize Eq. (17) in time.

As a preparation, we assemble the mass matrix $\boldsymbol{M} \in \mathbb{R}^{3N\times 3N}$ by a 16-point Gauss–Legendre quadrature method (see the supplementary material) according to the component definition in Eq. (18). The generalized coordinates and the corresponding generalized forces

exerted on them are also numerically integrated and stacked into a $3N$-dimensional $\boldsymbol{q}$ and $\boldsymbol{F}$, respectively. Then Eq. (17) is reformulated as

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{F}, \tag{27}$$

which is further discretized by the implicit Euler scheme in time:

$$\begin{cases} \dot{\boldsymbol{q}}^{n+1} = \dot{\boldsymbol{q}}^{n} + \Delta t\, \boldsymbol{M}^{-1}\boldsymbol{F}(\boldsymbol{q}^{n+1}, \dot{\boldsymbol{q}}^{n+1}, t^{n+1}), & \text{(a)} \\ \boldsymbol{q}^{n+1} = \boldsymbol{q}^{n} + \Delta t\, \dot{\boldsymbol{q}}^{n+1}. & \text{(b)} \end{cases} \tag{28}$$

By convention, we adopt Newton's method to solve the above equations iteratively, where quadratic and cubic line searchers referring to the widely used library *ArcSim* [21,72] are integrated. During each Newton step, the linear system is solved by a direct sparse LDLT Cholesky factorizations [73].

Besides, for the aforementioned positional constraints at arbitrary points, we use the augmented Lagrangian method [74] to solve the optimization problem, which acquires a better convergence rate than a pure method of Lagrange multipliers.

The Hessian matrix. In a Newton's method, the analytical form of $\partial \boldsymbol{F}/\partial \boldsymbol{q}_I$ $(I = 1, 2, 3, \ldots, N)$ is required. The most tricky part of the derivatives is the contribution of the elastic force, namely the Hessian matrix of V_e w.r.t. the generalized coordinates. We have carefully derived the elastic energy's first- and second-order derivatives for the BHEM solver and provided the concrete formulations in the supplementary material. In some situations, an inexact Hessian matrix may reduce the time cost of the solver's convergence, due to the high overhead of assembling the exact one. This alternative, which we called the *pseudo* Hessian matrix, is also given in the appendix. Moreover, when facing with ill-conditioned Hessian matrix, a diagonal regularizer is added to make the matrix positive definite.

6.2. Collision handling

We utilize the point-to-surface intersection detection scheme for CCD through sampling strategies adjusted to specific scenarios. For collision detection between shells and colliders with simple geometry shapes, such as spheres, cylinders, or planes, the midsurface is uniformly sampled. As for the collision detection with colliders possessing complex shapes, we sample the collider surface, or directly take the surface vertices as the sample points, if the collider has a triangular mesh. For the thin-shell self-collision, we iteratively sample each Hermite patch and perform CCD on sampling points against all the other surfaces. Penetrations occurring within a single surface can be handled by subdividing the current surface with the scheme illustrated in Fig. 9(b) and treating the 4 subdivided regions recursively.

After we have examined all the collision primitive pairs and got the list of earliest simultaneous collision pairs, we roll back to the collision moment. This prohibits penetration throughout the simulation. Then we follow the impulse-based method [75] to compute the simultaneous collision responses according to the conservation of momentum. Using a zero restitution coefficient [2], we model each collision as the momentum change at the collision point, which is formulated as a linear constraint of generalized velocities. Then we handle the multiple collisions in one batch by solving the linear system of constraints for the impulses in a least-square sense and update velocities. When friction exists, we compute the applied frictional impulses on the tangent direction of collision points according to the Column cone and solve for the normal and tangent updates together.

To avoid the resolved collision being detected at the beginning of the next round of CCD, we additionally update the displacement by pushing out each collision point a subtle distance opposite the normal direction. This is also modeled as a constraint linear to the generalized positions. Then the position constraints are resolved in a similar way after the velocity updates.

6.3. Rendering

In a ray-tracing rendering framework, the most expensive part is to test whether, where, and when a ray intersects the object. Traditional subdivision-based methods own their advantage in robustness but often fall in efficiency. We implement our ray-tracing algorithm in the rendering system of *pbrt-v4* [76] using our algorithm of static ray-patch intersection detection. Our enhancements to the subdivision-based algorithm have significantly accelerated performance, achieving an approximate 10-fold increase in speed compared to theoretically quadratic-convergent algorithms, such as Bézier clipping, as shown in Table 1 and Fig. 10. Newton's method tends to converge efficiently when applied to patches of low curvature. For example, the convergence percentages for each of the subdivisions in the three scenes depicted in Fig. 10 are 99%, 54%, and 34%, respectively. Additionally, Newton's method is adept at identifying the nearest intersection point. Consequently, the specially designed subdivision scheme can effectively prune much of the division tree.

7. Experimental results

We design a wide range of validation tests and simulation experiments to evaluate the validity, fidelity, and effectiveness of our framework from various aspects. All of the experiments are run on a 3.50 GHz 13th Gen Intel® Core™ i5-13600KF desktop with 32 GB RAM. The parameters and statistics are reported in Table 2.

In comparison with linear FEMs, we include two different bending formulas: the hinge energy proposed by Grinspun et al. [3] and the mid-edge bending (the discrete Koiter's shell energy) [22,70] implemented in *LibShell* [77]. It should be noted that the latter, along with the energy model employed in BHEM, directly originates from the continuous equations of the first and second fundamental forms. For the sake of fairness, we have standardized the linear search algorithm in each method.

7.1. Validation

Wrinkled sheets. We first validate the accuracy and convergence of our BHEM method with a standard stretched sheet experiment, which was first proposed by the pioneering work of Cerda et al. [78] and later used in both physical engineering [79] and computer graphics [18] community as well. In this experiment, a rectangular thin sheet is pulled apart from its two ends. Due to the high Poisson ratio, the sheet compresses in the perpendicular (vertical) and generates horizontal wrinkles.

We simulate this problem with the same physical parameter settings as in the work of Chen et al. [18], which are $0.25\,\mathrm{m} \times 0.1\,\mathrm{m}$ for size, $0.1\,\mathrm{mm}$ for thickness, Poisson ratio $\nu = 0.5$ and Young's modulus $Y = 1\,\mathrm{MPa}$. According to the conclusion of Chen et al. [18], the triangular finite-element-based method needs a high resolution of up to tens of thousands of vertices to generate correct wrinkle patterns in this example. As shown in Fig. 12, the BHEM starts to give apparent wrinkles at a resolution of merely 10^2 patches (1452 DoFs in total). As the resolution increases over 15^2, the difference in the shape of wrinkles is already imperceptible.

We further verify this observation through a quantitative experiment that measures the peak amplitude of wrinkles produced with increased resolution. According to the physical experiment results reported by Wang et al. [79], the sheet is expected to produce wrinkles with a peak amplitude of $0.35\,\mathrm{mm}$. The curve illustrated in Fig. 16 shows that our BHEM can stably yield $0.34\,\mathrm{mm}$ peak amplitude when the simulation resolution is greater than 30^2 patches (11.5k DoFs in total). While the peak amplitude can only reach $0.31\,\mathrm{mm}$ for the linear FEM simulated on a triangular mesh with 130k vertices (390k DoFs in total).

Fig. 1. A sheet of square cloth (30 × 30 patches) drapes on a ball, exhibiting rich wrinkle patterns in the process of reaching a steady state.

Fig. 2. A sheet of square cloth (30 × 30 patches) drapes on an armadillo model, where the sharp edges of the model are reflected by the cloth deformation.

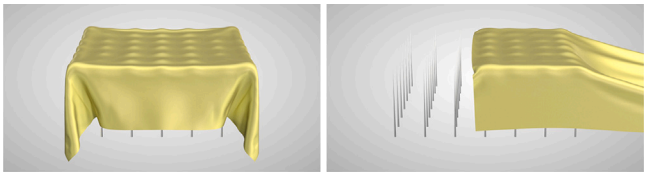


Fig. 3. A sheet of cloth (30 × 30 patches) falls on a needle array and then gets pulled away from aside. The bulges on the cloth surface are pushed by the needle tips. This sharp geometry deformation caused by the tiny contact demonstrates well the fine resolution of the patch interpolation.

Fig. 4. The folding process of an oriental paper parasol (280 × 5 patches), simulated by jointly controlling nodal positions and their first-order derivatives. Bending and subtle wrinkles along the ribs can be observed.

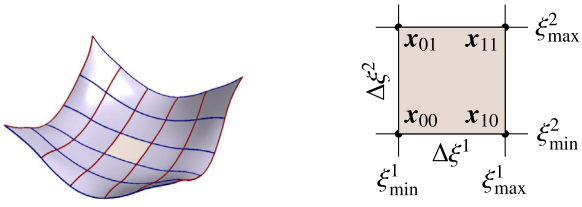
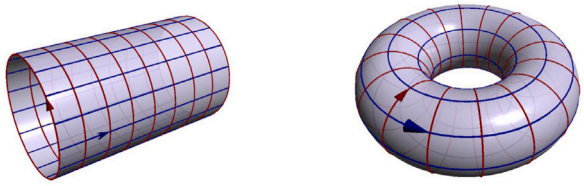


Fig. 5. A surface that is homeomorphic to a rectangle is embedded into a Cartesian grid, of which grid cells are taken as patches. Each BH patch corresponds to an axis-aligned rectangle in the parameter space.

In Fig. 13, we also demonstrate the efficiency of BHEM by comparing its convergence curves with those of randomly triangulated thin shells by aforementioned linear FEMs at various resolutions. Generally speaking, linear FEM solvers require fewer iterations but significantly more time to achieve a converged solution. This is probably because the BH system mixes DoFs of different orders, which, in the meanwhile, allows the BH surface to present similar high-frequency visual effects with fewer DoFs.

Draped cloth. We conduct a set of comparative experiments to further demonstrate the superiority of our geometric discretization format. In this scenario, a square piece of cloth, with its four corners moved inward a bit and clamped, drapes from a flattened configuration under the influence of gravity. In Fig. 14, we demonstrate the results obtained with our BHEM and *LibShell* at varying discretization resolutions respectively. The comparison shows that BHEM is capable of producing more detailed wrinkles with a comparable number of DoFs (11.5k DoFs for 30^2-patch BH surface, and 12k DoFs for 4k-vertex FE surface). Additionally, at low discretization resolutions, BHEM (432 DoFs for 5^2 patches) tens to yield an over-smoothed surface, whereas *LibShell* (1.5k DoFs for 500 vertices) generates artificial wrinkles, as illustrated in the first column. At high discretization resolutions, although the quality produced by *LibShell* improved, however, BHEM can achieve similar quality with only 11.5k DoFs (30^2 BH patches), thereby requiring significantly fewer iterations and less time to converge, as depicted in Fig. 15. The convergence speed of different methods is provided in Fig. 15, including results from the linear FEM with hinge-based bending.



(a) Cylinders; (b) Torus.

Fig. 6. Some complex surfaces can be divided into BH patches with the help of continuity boundary conditions.

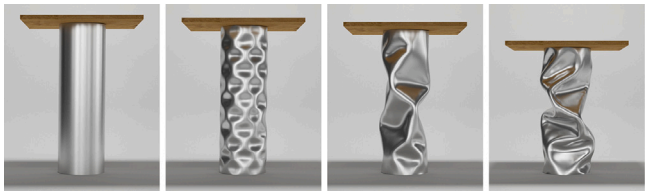


Fig. 7. A hollow cylindrical shell (20 × 20 patches) deforms severely under the gradually increased compression. Initially, the cylinder is mounted on the ground; as the wood plank gets pushed down, the deformation of the cylinder manifests locally at first; at some point, the unshaping takes a sudden change; further compression produces severe deformation and high degree of self contact.

Parametric surface rendering. Our proposed ray-surface intersection algorithm would be regarded as a substantial enrichment for the current off-the-shell ray-tracing rendering engine. By simply substituting the current ray-polygon intersection detection module in *pbrt-v4* [76] with our algorithm, *pbrt-v4* can realize parametric surface ray-tracing rendering with good visual effects, as shown in the rightmost columns of Figs. 8 and 11. The other columns are the rendering results for the polygon meshes generated from the parametric surface subdivision. From left to right, the subdivision level gradually increases. A parametric surface naturally processes continuous normal vectors on its surface. Discontinuities in surface normal vectors can result in visual artifacts, such as abnormal reflections on the teapot and cluttered caustics on the ground.

2D cantilever beam and 3D lateral buckling. We further validate our method by comparing two experiments with their theoretical solutions [80]. The first is the cantilever beam experiment, where a beam is fixed at one end and bent under gravity. Its master curve uniquely determines the aspect ratio H/W of the cantilever beam under equilibrium as a function of the dimensionless parameter $\Gamma^* = 12(1-\nu^2)\rho g L^3/Yh^2$. The second is the lateral buckling experiment with the plate lying vertically in the (x, z) plane. We let the plate hang and sag

(a) 265 vertices; (b) 1041 vertices; (c) 4129 vertices; (d) 16 449 vertices; (e) 32 patches.

Fig. 8. Parametric surface rendering. A Utah teapot represented in bicubic patches is rendered by our scheme, shown in (e). The teapots rendered by traditional methods with each patch triangulated into 4/16/64/256 facets are shown in (a)/(b)/(c)/(d), where artifacts can be seen at the mouth and silhouettes of the teapots, especially for coarse mesh decimation.

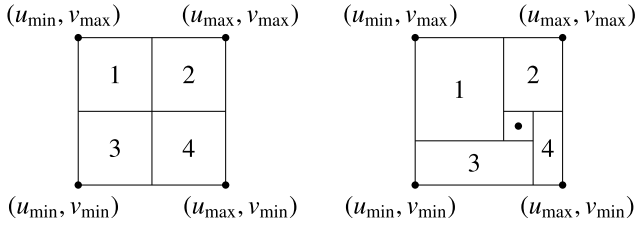


Fig. 9. Ray surface intersection subdivide schemes. Left: splitting at midpoint; Right: abandoning the neighborhood of intersection point and splitting the rest region.

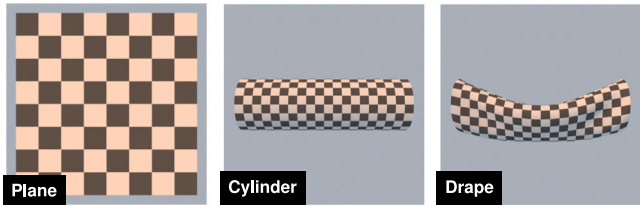
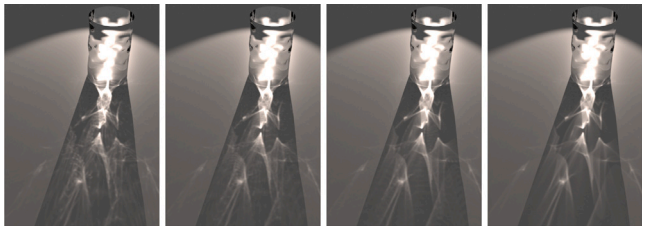
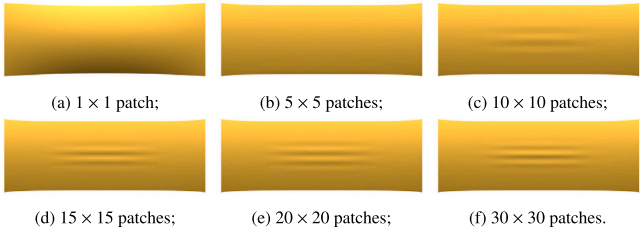


Fig. 10. Three test cases with only 1 patch for Plane and 10×10 patches for Cylinder and Drape each. The scenes all have the same setting of lights and 16 samples per pixel.



(a) 14 520 vertices; (b) 57 840 vertices; (c) 230 880 vertices; (d) 450 patches.

Fig. 11. Parametric surface rendering. The caustic lighting effect amplifies the imperfection of the surface, such as the un-smooth surface normal distribution. Fine-grained caustic rays clutter the ground for the scenarios with low-resolution mesh objects.



(a) 1×1 patch; (b) 5×5 patches; (c) 10×10 patches;
(d) 15×15 patches; (e) 20×20 patches; (f) 30×30 patches.

Fig. 12. Wrinkled sheets. Sheets of different resolutions buckle under uniaxial stretching. (a) There appears only one artificial bump on a single patch. (b) No apparent wrinkle on a shell composed of 5×5 patches. (c) A 10×10-patch sheet manages to produce several shallow wrinkles. (d) Clearer wrinkles are produced by a 15×15-patch sheet. (e) Though wrinkles are finer, the difference is subtle compared with the previous. (f) The wrinkle pattern converges when it is finer than 30×30 patches.

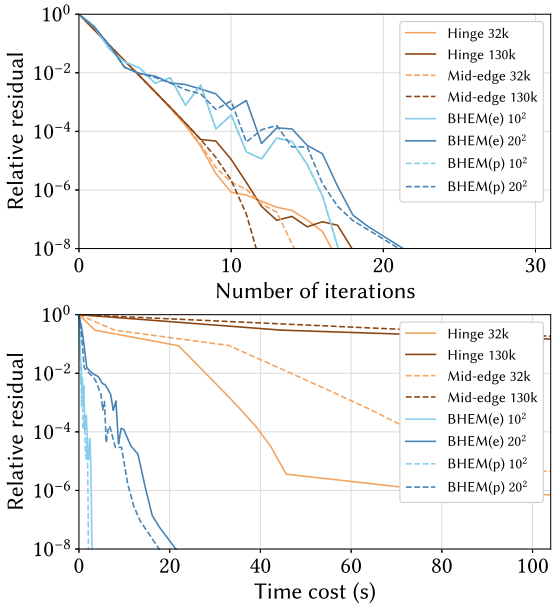


Fig. 13. Performance curves of wrinkled sheets with respect to different resolutions. Linear FEM solvers employing hinge-based bending and mid-edge bending are denoted as *hinge* and *mid-edge*, respectively. We use *BHEM(p)* and *BHEM(e)* to distinguish between BHEM solvers that utilize pseudo-Hessian and exact Hessian matrices, respectively. Note that the light-blue solid and dashed lines overlap in the upper subplot. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

under its weight, waiting for a bifurcation to occur. Its master curve determines when the plate buckles in the third direction as Γ^* increases under a given aspect ratio w/L. We conduct our experiments using methods and physical parameters provided by Romero et al. [80]. The beam in the cantilever test is discretized in 2×100 patches. And the plate is discretized in 20 patches per meter. Our results perfectly match the master curves as shown in Figs. 18 and 19.

7.2. Experiments

Locking. As a high-order method, the BHEM formulation alleviates locking issues remarkably. It does not need any special treatment, e.g., dynamic remeshing, to get plausible effects, even with very few degrees of freedom. As shown in the top row of Fig. 20, a squared piece of cloth, which is composed of 10^2 patches, is initially pinned at its two diagonal corners. As the pinning points slide along the boundary, the cloth can naturally fold down along any direction as illustrated in the bottom row.

Twisting cloth. Our method allows precise and intuitive control over the first-order derivatives of an arbitrary point on the surface. In this example, the center point of a squared piece of cloth is fixed. Its two first-order partial derivatives (indicated by the red arrows in Fig. 17 are rotated with a constant speed in the horizontal plane. This prescript motion results in a persistent wrinkling perpendicular to the first

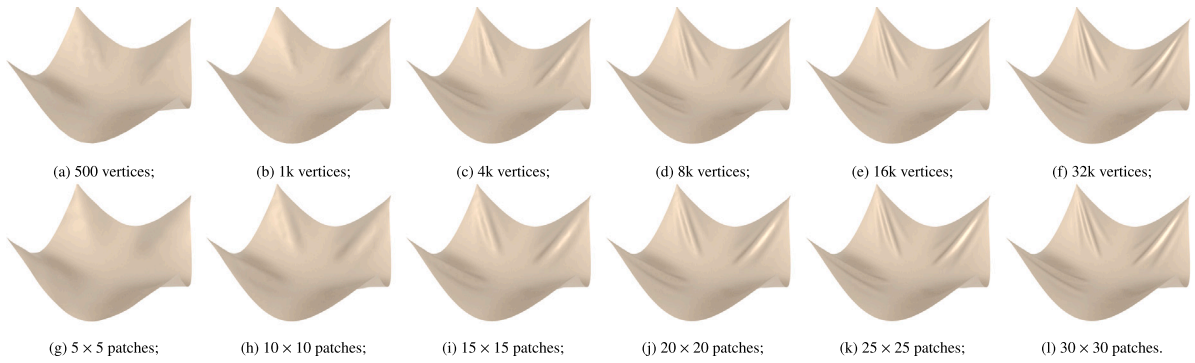


Fig. 14. Draped cloth. The top row and bottom row show the results obtained by *LibShell* and BHEM, respectively. The resolution of the discretized surfaces increases from left to right. We choose the examples with the highest resolutions and similar plausible wrinkles and test the performance of the solvers under these resolutions. The performance curves are shown in Fig. 15.

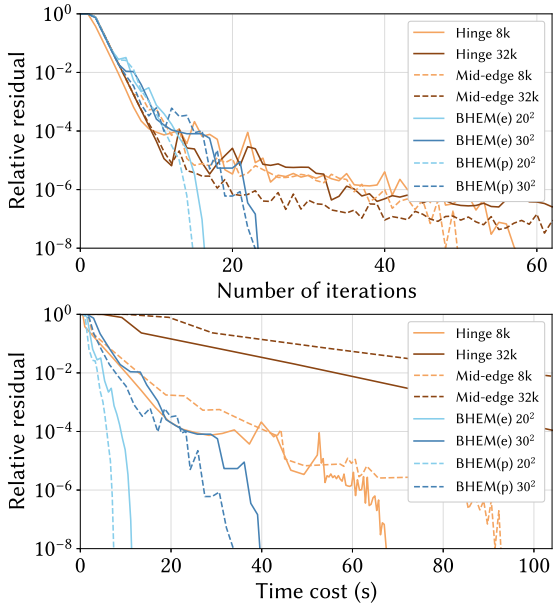


Fig. 15. Performance curves of draped cloth with respect to different resolutions. Same abbreviations are used as in Fig. 13.

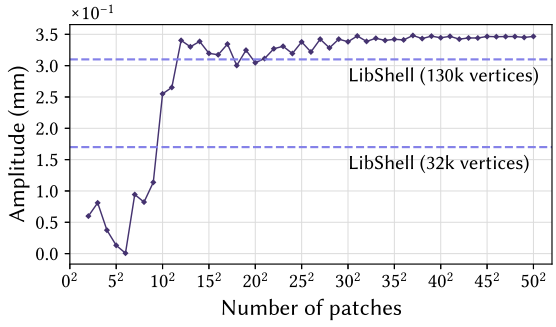


Fig. 16. Statistic of the wrinkle's peak value. The simulated amplitude increases with the refinement of the sheet and converges rapidly to a physical real value (3.5×10^{-1} mm). Results of *LibShell* with 32k and 130k vertices are plotted in horizontal dashed lines.

derivative directions and eventually invokes the buckling swirl around the center. In a pure displacement-based method, a similar result can only be achieved through prescript at least three nodes' motion.

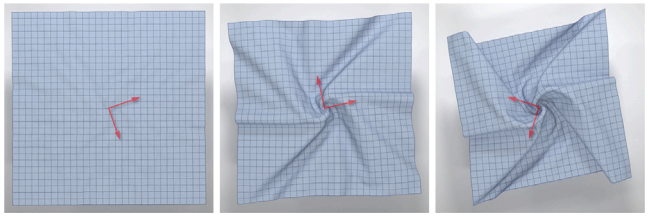


Fig. 17. Twisting cloth (30 × 30 patches). A buckling swirl around the center as the first-order derivatives (indicated by the red arrows) of the center point rotate horizontally with a constant speed.

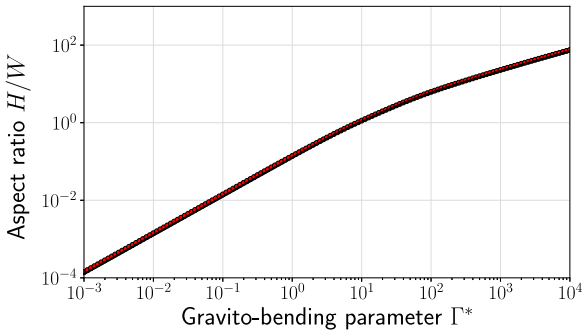


Fig. 18. Comparisons with theoretical solutions on the cantilever test. We simulate 140 Γ^* values and superimpose the data (red dots) onto the master curve (black line). Our results perfectly match the master curve.

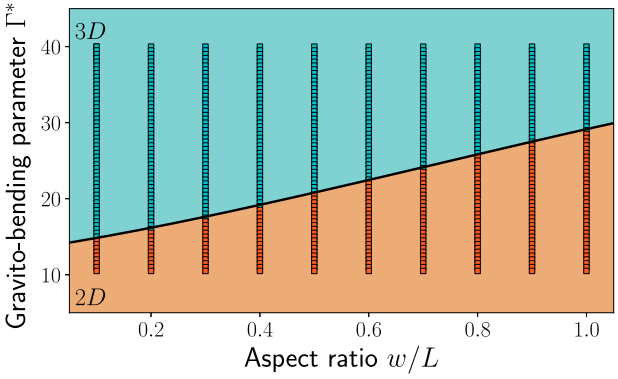
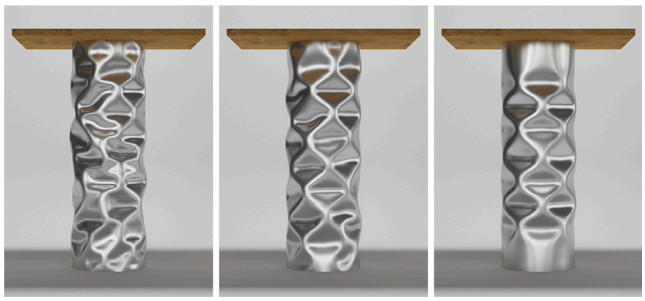


Fig. 19. Comparisons with theoretical solutions on the lateral buckling test. The master curve separates two areas colored to indicate whether the plate has buckled in 3D (turquoise) or lies in 2D (orange). Our results perfectly match the theoretical solution. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 20. A thin sheet (10 × 10 patches) with its two diagonal ends fixed bends by gravity. After reaching a steady state, the fixed points slide smoothly along the opposite edges without any locking artifacts from discretization. We emphasize that only 10 × 10 patches are used here.



(a) $h = 0.5$mm (b) $h = 1$mm (c) $h = 2$mm

Fig. 21. Hollow cylindrical shells (30 × 30 patches) with different thickness buckles under the gradually increased axial compression. Inside each shell, there is a rigid cylindrical mandrel. Its radius differs from that of the shell by 10%.

Cloth draping on objects. Both resting and dynamic contact can be faithfully detected and resolved within our framework. Two square sheets of cloth, both of which are composed of 30^2 patches, drape on a parametric sphere and a triangular meshed armadillo are demonstrated in Figs. 1 and 2 respectively. Rich deformation details due to the interference from external objects can be observed in the results. Please see the supplemental video for more visual evidence.

Cloth sliding over needles. In this example, we drop a sheet of cloth on a needle array and then pull it away from aside. The subtle bulges on the cloth surface, which are pushed out by the needle tips, can be clearly observed in Fig. 3. In the supplemental video, we can easily notice that when pulling the cloth over the needle array, the cloth exhibits natural choppy movement around the needle tips due to the contact interaction. These two points verify that our BHEM formulation and collision detection algorithm can handle sharp geometry features robustly.

Folding an oriental paper parasol. By jointly controlling node positions and their first-order derivatives, we can mimic the folding process of an oriental paper parasol driven by the motion of its rib as displayed in Fig. 4.

Cylindrical shell buckling. In Fig. 7, a hollow cylindrical shell severely buckles under the gradually increased compression is simulated with our method. When there is an inner mandrel, local buckles can no longer grow unrestrained because their radial displacements are arrested by the mandrel. Consequently, as illustrated in Fig. 21, each buckle folds along its central ridge, orthogonal to both the pressure and the axis of the cylinder. Under compression, more buckles emerge, forming a densely packed, diamond-like pattern across the entire cylinder. Given that the bending stiffness of a shell is proportional to the cube of its thickness, thicker shells display more regular, rapid, and stable buckling development, as shown in the accompanying video.

Table 1
Rendering performance. The three algorithms from left to right are: our subdivision-based algorithm that splits only at the midpoint, our algorithm with Newton's method, and Bézier clipping.

	Subdivision	Subdivision (Opt.)	Bézier clipping
Plane	58.3 s	4.5 s	6.0 s
Cylinder	91.6 s	9.7 s	14.2 s
Drape	86.0 s	14.8 s	33.9 s

Furthermore, our experiment setup ensures a constant ratio between the mandrel radius and the shell thickness, thereby keeping the annular gap unchanged. As depicted in Fig. 21, the size of the buckle remains consistent regardless of variations in thickness, which aligns well with the conclusion presented in Seffen et al. [81].

8. Conclusions and discussions

In this study, we propose a new computational framework designed for elastodynamic simulation of thin-shell structures. The central piece of our framework is a high-order finite element formulation equipped with an implicit Euler solver. This formulation, based on bicubic Hermite interpolation, naturally ensures conforming C^1 continuity. Capitalizing on the advancements in surface modeling and rendering, we have crafted an intersection detection paradigm that is custom-tailored for bicubic Hermite surfaces. This unified approach empowers us to achieve high-fidelity CCD and rendering without resorting to any form of auxiliary tessellation mesh. The significance of research on BHEM spans various applications: (1) facilitating a seamless flow of information throughout the design-to-production workflow (CAD, CAE, and CAM), where geometric consistency with IGA is needed, (2) enhancing inverse optimization and control, where fewer DoFs (compact representation) aid in fast convergence, (3) enabling locking-free simulation with rigorous strain limiting, which necessitates sufficient DoFs of each vertex, and (4) advancing development of conforming contact solver, where accurately capturing contact regions is essential.

Rotation dependence. The basis functions for tensor-product bicubic Hermite splines are $(\xi^1)^i(\xi^2)^j$ $(0 \le i, j \le 3)$, which include a complete set of rotation-free basis functions, specifically $(\xi^1)^i(\xi^2)^j$ $(0 \le i + j \le 3)$. Despite that the geometric representation of BHE is affected by the directions of parameterization, the differences only exist in several high-order bases.

Comparisons with relevant methods. The BHEM achieves an impressive balance between computational efficiency and geometric continuity, compared to relevant methods. For instance, subdivision surfaces also ensure C^1 continuity but do not interpolate control points, resulting in difficulties of enforcing boundary conditions and detecting collisions. Specifically, subdivision surfaces utilizing Loop's scheme [8] employ quartic shape functions, which demand more computation than ours. Meanwhile, those using Catmull–Clark's scheme [30] generate bicubic B-splines like ours, but their values and derivatives of the shape functions cannot be evaluated analytically. In the case of DG-FEM, although it employs lower-order (quadratic) shape functions, it necessitates an additional penalty between each pair of adjacent elements, which is expected to quickly offset the extra computational cost of evaluating higher-order polynomials as mesh size increases. Moreover, for the same number of DoFs, DG-FEM has been observed to be less accurate than the FEMs used in CG [32].

Limitations and future work. Some challenges remain unresolved and await deeper investigation. First, in order to maintain $\mathcal{G}^1$-continuous everywhere, our current framework is limited to working with surfaces that are homeomorphic to squares, cylinders, or tori. For more complex shapes (e.g., spheres), special treatments, such as integrating hinge-based bending energy, are still necessary around $\mathcal{G}^0$ seams and

Table 2
Here we list the parameters and the time consumed in each example, including the degrees of freedom, the total number of nonzero elements in the BHEM system matrix, the total number of sampling points on the external collider, the properties of the cloth (Young's Modulus Y [Pa], Poisson's ratio ν and thickness h [mm], mass density ρ [10^3kg/m^3]), Rayleigh damping coefficient α, time step Δt [ms], time consumed in one integration step t_{int} [s], in one CCD step t_{obj} [s], and whether we detect self-collision (SC, '–' means that CCD has not been performed in this example).

Figure	Example	#DoFs	#NNZs	#SPs	Y	ν	h	ρ	α	Δt	t_{int}	t_{obj}	SC
12	Wrinkled[a]	–	–	–	1×10^6	0.5	0.1	0.93	–	–	63.0	–	–
14	Draped[a]	–	–	–	1×10^5	0.3	1	0.93	0	10	6.01	–	–
17	Twisting	1.15×10^4	1.19×10^6	–	1×10^4	0.3	1	0.2	50	2	0.20	–	–
20	Locking[a]	–	–	–	1×10^4	0.3	1	0.2	2	2	3.38	4.37	Yes
1	Ball	1.15×10^4	1.17×10^6	–	1×10^5	0.3	0.1	0.2	2.5	2	4.80	0.0023	No
2	Armadillo	1.15×10^4	1.19×10^6	2.8k	8.21×10^5	0.243	0.32	0.4726	2.5	1	2.90	1.84	No
3	Needles	1.15×10^4	1.19×10^6	3.6k	1×10^4	0.3	0.1	0.2	5	2	5.05	26.66	Yes
4	Umbrella	2.02×10^4	1.76×10^6	–	1×10^6	0.5	0.1	0.93	5	2	12.47	–	–
7	Can	5.3×10^3	4.97×10^5	–	1×10^8	0.3	2	2.7	5	2	4.55	–	–
21	Cans[a]	1.15×10^4	1.13×10^6	–	1×10^9	0.47	–	1.4	5	0.04	5.29	0.0024	No

[a] The statistics is obtained by 30 × 30-patch shells for *Wrinkled* and *Draped*, 20 × 20-patch shells for *Locking*, and 1 mm-thick shell for *Cans*.

points. Second, while Hermite interpolation excels in depicting complex characteristics like wrinkles or folds in cloth motion, it encounters difficulties when addressing plastic deformation and impact dynamics. The high-order formulation of Hermite interpolation exacerbates the inherent non-linearity and non-smoothness of these tasks, which need sophisticated modeling and analysis. Besides, this paper only introduces an algorithm of ray-patch intersection detection, which alone is insufficient for achieving non-penetration in CCD. To overcome this limitation, we intend to expand our algorithm to include patch–patch intersection detection. In addition, as our current collision-resolving scheme struggles with intricate contact scenarios like tying ribbons into a reef knot, we view the integration of Incremental Potential Contact (IPC) as a natural and significant follow-up step.

CRediT authorship contribution statement

Xingyu Ni: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Xuwen Chen:** Data curation, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Cheng yu:** Resources, Software, Visualization, Writing – review & editing. **Bin Wang:** Conceptualization, Formal analysis, Project administration, Supervision, Writing – original draft, Writing – review & editing. **Baoquan Chen:** Funding acquisition, Project administration, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgement

This work was supported in part by National Key R&D Program of China (2022ZD0160801) and Shenzhen Collaborative Innovation Program (CJGJZD2021048092601003). The Houdini Education license is credited for the video generations.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cad.2024.103734.

References

[1] Ciarlet PG. Theory of shells. Mathematical elasticity, vol. 3, Devon, England, UK: North Holland; 2000.
[2] Bridson R, Fedkiw R, Anderson J. Robust treatment of collisions, contact and friction for cloth animation. ACM Trans Graph 2002;21(3):594–603.
[3] Grinspun E, Hirani AN, Desbrun M, Schröder P. Discrete shells. In: Proceedings of the 2003 ACM SIGGRAPH/eurographics symposium on computer animation. Goslar, DEU: Eurographics Association; 2003, p. 62–7.
[4] Bergou M, Wardetzky M, Harmon D, Zorin D, Grinspun E. Discrete quadratic curvature energies. In: ACM SIGGRAPH 2006 courses. New York, NY, USA: Association for Computing Machinery; 2006, p. 20–9.
[5] Gingold Y, Secord A, Han JY, Grinspun E, Zorin D. A discrete model for inelastic deformation of thin shells. In: ACM SIGGRAPH/eurographics symposium on computer animation. Goslar, DEU: Eurographics Association; 2004, p. 1–12.
[6] Hughes T, Cottrell J, Bazilevs Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput Methods Appl Mech Engrg 2005;194(39):4135–95.
[7] Cottrell JA, Hughes TJR, Bazilevs Y. Isogeometric analysis: Toward integration of CAD and FEA. 1st ed. Hoboken, NJ, USA: Wiley Publishing; 2009.
[8] Cirak F, Ortiz M, Schröder P. Subdivision surfaces: a new paradigm for thin-shell finite-element analysis. Internat J Numer Methods Engrg 2000;47(12):2039–72.
[9] Cirak F, Ortiz M. Fully C1-conforming subdivision elements for finite deformation thin-shell analysis. Internat J Numer Methods Engrg 2001;51(7):813–33.
[10] Lu J, Zheng C. Dynamic cloth simulation by isogeometric analysis. Comput Methods Appl Mech Engrg 2014;268:475–93.
[11] Phusakulkajorn W. Finite element based solutions of thin-shell problems with a small strain [Master's thesis], The University of Manchester; 2013.
[12] Greco L, Cuomo M. An implicit G1-conforming bi-cubic interpolation for the analysis of smooth and folded Kirchhoff–Love shell assemblies. Comput Methods Appl Mech Engrg 2021;373:113476.
[13] Greco L, Cuomo M, Contrafatto L. A quadrilateral G1-conforming finite element for the Kirchhoff plate model. Comput Methods Appl Mech Engrg 2019;346:913–51.
[14] Zienkiewicz O, Taylor R, David F. The finite element method for solid and structural mechanics. Oxford: Butterworth-Heinemann; 2005.
[15] Baraff D, Witkin A. Large steps in cloth simulation. In: Proceedings of the 25th annual conference on computer graphics and interactive techniques. New York, NY, USA: Association for Computing Machinery; 1998, p. 43–54.
[16] Choi K-J, Ko H-S. Stable but responsive cloth. ACM Trans Graph 2002;21(3):604–11.
[17] Tamstorf R, Jones T, McCormick SF. Smoothed aggregation multigrid for cloth simulation. ACM Trans Graph 2015;34(6).
[18] Chen Z, Chen H-Y, Kaufman DM, Skouras M, Vouga E. Fine wrinkling on coarsely meshed thin shells. ACM Trans Graph 2021;40(5).
[19] Wang H. GPU-based simulation of cloth wrinkles at submillimeter levels. ACM Trans Graph 2021;40(4).
[20] Zhang JE, Dumas J, Fei YR, Jacobson A, James DL, Kaufman DM. Progressive simulation for cloth quasistatics. ACM Trans Graph 2022;41(6).
[21] Narain R, Pfaff T, O'Brien JF. Folding and crumpling adaptive sheets. ACM Trans Graph 2013;32(4).
[22] Chen H-Y, Sastry A, van Rees WM, Vouga E. Physical simulation of environmentally induced thin shell deformation. ACM Trans Graph 2018;37(4).
[23] Burgoon R, Wood ZJ, Grinspun E. Discrete shells origami. In: Jackson DJ, editor. 21st international conference on computers and their applications, CATA-2006, Seattle, Washington, USA, March 23–25, 2006, proceedings. ISCA; 2006, p. 180–7.
[24] Pfaff T, Narain R, de Joya JM, O'Brien JF. Adaptive tearing and cracking of thin sheets. ACM Trans Graph 2014;33(4).

[25] Rémillard O, Kry PG. Embedded thin shells for wrinkle simulation. ACM Trans Graph 2013;32(4).
[26] Li P, Kry PG. Multi-layer skin simulation with adaptive constraints. In: Proceedings of the 7th international conference on motion in games. New York, NY, USA: Association for Computing Machinery; 2014, p. 171–6.
[27] Terzopoulos D, Platt J, Barr A, Fleischer K. Elastically deformable models. In: Proceedings of the 14th annual conference on computer graphics and interactive techniques. New York, NY, USA: Association for Computing Machinery; 1987, p. 205–14.
[28] Bridson R, Marino S, Fedkiw R. Simulation of clothing with folds and wrinkles. In: Proceedings of the 2003 ACM SIGGRAPH/eurographics symposium on computer animation. Goslar, DEU: Eurographics Association; 2003, p. 28–36.
[29] Thomaszewski B, Wacker M, Straßer W. A consistent bending model for cloth simulation with corotational subdivision finite elements. In: Proceedings of the 2006 ACM SIGGRAPH/eurographics symposium on computer animation. Goslar, DEU: Eurographics Association; 2006, p. 107–16.
[30] Clyde D, Teran J, Tamstorf R. Modeling and data-driven parameter estimation for woven fabrics. In: Proceedings of the ACM SIGGRAPH / eurographics symposium on computer animation. New York, NY, USA: Association for Computing Machinery; 2017, p. 1–11.
[31] Guo Q, Han X, Fu C, Gast T, Tamstorf R, Teran J. A material point method for thin shells with frictional contact. ACM Trans Graph 2018;37(4).
[32] Kaufmann P, Martin S, Botsch M. Implementation of discontinuous Galerkin Kirchhoff-love shells. CTIT technical reports series 622, 2009.
[33] Wawrzinek A, Hildebrandt K, Polthier K. Koiter's thin shells on catmull-clark limit surfaces. In: Eisert P, Hornegger J, Polthier K, editors. Vision, modeling, and visualization (2011). Berlin, Germany: The Eurographics Association; 2011, p. 113–20.
[34] Qin H, Terzopoulos D. Triangular NURBS and their dynamic generalizations. Comput Aided Geom Design 1997;14(4):325–47.
[35] Trusty T, Chen H, Levin DIW. The shape matching element method: Direct animation of curved surface models. ACM Trans Graph 2021;40(4).
[36] Clough R, Tocher J. Finite element stiffness matrices for analysis of plates in bending. In: Proceedings of the conference on matrix methods in structural mechanics. 1965, p. 515–45.
[37] Bell K. A refined triangular plate bending finite element. Internat J Numer Methods Engrg 1969;1(1):101–22.
[38] Bogner F. The generation of interelement compatible stiffness and mass matrices by the use of interpolation formulae. In: Proceedings of the conference on matrix methods in structural mechanics. 1965, p. 397–444.
[39] Petera J, Pittman JFT. Isoparametric Hermite elements. Internat J Numer Methods Engrg 1994;37:3489–519.
[40] Beheshti A. Novel quadrilateral elements based on explicit Hermite polynomials for bending of Kirchhoff–Love plates. Comput Mech 2018;62:1199–211.
[41] Šolín P, Segeth K. Hierarchic higher-order hermite elements on hybrid triangular/quadrilateral meshes. Math Comput Simulation 2007;76(1):198–204, Mathematical Modelling and Computational Methods in Applied Sciences and Engineering.
[42] Farin G. Curves and surfaces for CAGD: A practical guide. 5th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 2001.
[43] Salomon D. Curves and surfaces for computer graphics. Berlin, Heidelberg: Springer-Verlag; 2005.
[44] Kajiya JT. Ray tracing parametric patches. SIGGRAPH Comput Graph 1982;16(3):245–54.
[45] Manocha D. Solving systems of polynomial equations. IEEE Comput Graph Appl 1994;14(2):46–55.
[46] Benthin C. Realtime ray tracing on current CPU architectures [Ph.D. thesis], Saarbrücken, Germany: Saarland University; 2006, p. 1–194.
[47] Abert O, Geimer M, Muller S. Direct and fast ray tracing of NURBS surfaces. In: Symposium on interactive ray tracing. Salt Lake City, UT, USA: IEEE; 2006, p. 161–8.
[48] Geimer M, Abert O. Interactive ray tracing of trimmed bicubic bézier surfaces without triangulation. In: International conference in central europe on computer graphics and visualization. Bory, Plzen, Czech Republic: University of West Bohemia; 2005, p. 71–8.
[49] Martin W, Cohen E, Fish R, Shirley P. Practical ray tracing of trimmed NURBS surface. J Graph Tools 2000;5.
[50] Toth DL. On ray tracing parametric surfaces. SIGGRAPH Comput Graph 1985;19(3):171–9.
[51] Barth W, Stuerzlinger W. Efficient ray tracing for bezier and B-spline surfaces. Comput Graph 1993;17:423–30.
[52] Nishita T, Sederberg TW, Kakimoto M. Ray tracing trimmed rational surface patches. In: Proceedings of the 17th annual conference on computer graphics and interactive techniques. New York, NY, USA: Association for Computing Machinery; 1990, p. 337–45.
[53] Campagna S, Slusallek P, Seidel H. Ray tracing of spline surfaces: Bézier clipping, Chebyshev boxing, and bounding volume hierarchy - a critical comparison with new results. Vis Comput 1997;13(6):265–82.
[54] Rogers DF. Procedural elements for computer graphics. USA: McGraw-Hill, Inc.; 1984.
[55] Woodward C. Ray tracing parametric surfaces by subdivision in viewing plane. In: Theory and practice of geometric modeling. Berlin, Heidelberg: Springer-Verlag; 1989, p. 273–87.
[56] Sederberg T, Nishita T. Curve intersection using Bézier clipping. Comput Aided Des 1990;22(9):538–49.
[57] Tejima T, Fujita M, Matsuoka T. Direct ray tracing of full-featured subdivision surfaces with bezier clipping. J Comput Graph Tech (JCGT) 2015;4(1):69–83.
[58] Efremov A, Havran V, Seidel H-P. Robust and numerically stable Bézier clipping method for ray tracing NURBS surfaces. In: Proceedings of the 21st spring conference on computer graphics. New York, NY, USA: Association for Computing Machinery; 2005, p. 127–35.
[59] Slusallek P, Seidel H-P. Vision - An architecture for global illumination calculations. IEEE Trans Vis Comput Graphics 1995;1(1):77–96.
[60] Pabst H, Springer J, Schollmeyer A, Lenhardt R, Lessig C, Froehlich B. Ray casting of trimmed NURBS surfaces on the GPU. In: Symposium on interactive ray tracing. Salt Lake City, UT, USA: IEEE; 2006, p. 151–60.
[61] Ferguson Z, Jain P, Zorin D, Schneider T, Panozzo D. High-order incremental potential contact for elastodynamic simulation on curved meshes. In: ACM SIGGRAPH 2023 conference proceedings. New York, NY, USA: Association for Computing Machinery; 2023, p. 1–11.
[62] Von Herzen B, Barr AH, Zatz HR. Geometric collisions for time-dependent parametric surfaces. SIGGRAPH Comput Graph 1990;24(4):39–48.
[63] Snyder JM, Woodbury AR, Fleischer K, Currin B, Barr AH. Interval methods for multi-point collisions between time-dependent curved surfaces. In: Proceedings of the 20th annual conference on computer graphics and interactive techniques. New York, NY, USA: Association for Computing Machinery; 1993, p. 321–34.
[64] Hughes M, DiMattia C, Lin M, Manocha D. Efficient and accurate interference detection for polynomial deformation. In: Proceedings computer animation '96. 1996, p. 155–66.
[65] Lu J. Isogeometric contact analysis: Geometric basis and formulation for frictionless contact. Comput Methods Appl Mech Engrg 2011;200(5):726–41.
[66] Krishnan S, Gopi M, Lin M, Manocha D, Pattekar A. Rapid and accurate contact determination between spline models using ShellTrees. Comput Graph Forum 1998;17(3):315–26.
[67] Teschner M, Kimmerle S, Heidelberger B, Zachmann G, Raghupathi L, Fuhrmann A, et al. Collision detection for deformable objects. Comput Graph Forum 2005;24(1):61–81.
[68] Marschner Z, Zhang P, Palmer D, Solomon J. Sum-of-squares geometry processing. ACM Trans Graph 2021;40(6).
[69] Zhang P, Marschner Z, Solomon J, Tamstorf R. Sum-of-squares collision detection for curved shapes and paths. In: ACM SIGGRAPH 2023 conference proceedings. New York, NY, USA: Association for Computing Machinery; 2023, p. 1–11.
[70] Weischedel C. A discrete geometric view on shear-deformable shell models [Ph.D. thesis], Göttingen, Niedersachsen, Germany: Georg-August-Universität Göttingen; 2012.
[71] Boehm W, Müller A. On de Casteljau's algorithm. Comput Aided Geom Design 1999;16(7):587–605.
[72] Narain R, Samii A, O'Brien JF. Adaptive anisotropic remeshing for cloth simulation. ACM Trans Graph 2012;31(6).
[73] Guennebaud G, Jacob B, et al. Eigen v3. 2010, http://eigen.tuxfamily.org.
[74] Hestenes MR. Multiplier and gradient methods. J Optim Theory Appl 1969;4:303–20.
[75] Harmon D, Vouga E, Tamstorf R, Grinspun E. Robust treatment of simultaneous collisions. ACM Trans Graph 2008;27(3):1–4.
[76] Pharr M, Jakob W, Humphreys G. Physically based rendering, fourth edition: From theory to implementation. 4th ed. Cambridge, MA, USA: The MIT Press; 2023.
[77] Vouga E, Jourdan D. LibShell. 2021, https://github.com/evouga/libshell.
[78] Cerda E, Mahadevan L. Geometry and physics of wrinkling. Phys Rev Lett 2003;90:074302.
[79] Wang T, Fu C, Xu F, Huo Y, Potier-Ferry M. On the wrinkling and restabilization of highly stretched sheets. Internat J Engrg Sci 2019;136:1–16.
[80] Romero V, Ly M, Rasheed AH, Charrondière R, Lazarus A, Neukirch S, et al. Physical validation of simulators in computer graphics: A new framework dedicated to slender elastic structures and frictional contact. ACM Trans Graph 2021;40(4).
[81] Seffen KA, Stott SV. Surface texturing through cylinder buckling. J Appl Mech 2014;81(6):061001.

Simulating Thin Shells by Bicubic Hermite Elements (Supplementary Document)

XINGYU NI, XUWEN CHEN, CHENG YU, Peking University, China
BIN WANG, State Key Laboratory of General Artificial Intelligence, BIGAI, China
BAOQUAN CHEN, State Key Laboratory of General Artificial Intelligence, Peking University, China

A MATHEMATICAL TOOLS

A.1 Bicubic Hermite Interpolation

Thoroughly, the midsurface in BHEM is parameterized by

$$\boldsymbol{x}(\xi^1,\xi^2)=\sum_{p,q,r,s\in\{0,1\}} w_{pq,rs}(\theta^1,\theta^2)\boldsymbol{x}_{pq,rs}, \tag{S1}$$

in which $\boldsymbol{x}_{pq,rs}$ represents 4 generalized coordinates of node $\boldsymbol{x}_{pq}$, with r and s denoting the order of partial derivatives w.r.t. ξ^1 and ξ^2, respectively. See Fig. 5 of the main text. The weight function $w_{pq,rs}$ takes the form of

$$w_{pq,rs}(\theta^1,\theta^2)=w_{p,r}(\theta^1)\,w_{q,s}(\theta^2), \tag{S2}$$

with $\theta^\alpha\in[0,1]$ defined as $(\xi^\alpha-\xi^\alpha_{\min})/\Delta\xi^\alpha$.

As written in Eq. (S2), bicubic Hermite interpolation indicates using a cubic basis function in each dimension of the parameter space. To be specific, $w_{p,r}$ is defined as follows:

$$w_{p,r}(\theta)=\begin{cases} f(\theta), & p=0\wedge r=0, \quad \text{(S3a)}\\ f(1-\theta), & p=1\wedge r=0, \quad \text{(S3b)}\\ g(\theta), & p=0\wedge r=1, \quad \text{(S3c)}\\ -g(1-\theta), & p=1\wedge r=1, \quad \text{(S3d)}\end{cases}$$

where $f(\theta)=2\theta^3-3\theta^2+1$ and $g(\theta)=\theta^3-2\theta^2+\theta$ hold. $w_{q,s}$ is defined similarly by replacing indices.

Furthermore, taking the derivative of Eq. (S1) yields

$$\boldsymbol{a}_1(\xi^1,\xi^2)=\sum_{p,q,r,s\in\{0,1\}}\frac{1}{\Delta\xi^1}w'_{p,r}(\theta^1)\,w_{q,s}(\theta^2)\,\boldsymbol{x}_{pq,rs}, \tag{S4}$$

$$\boldsymbol{a}_2(\xi^1,\xi^2)=\sum_{p,q,r,s\in\{0,1\}}\frac{1}{\Delta\xi^2}w_{p,r}(\theta^1)\,w'_{q,s}(\theta^2)\,\boldsymbol{x}_{pq,rs}, \tag{S5}$$

where

$$w'_{p,r}(\theta)=\begin{cases} f'(\theta), & p=0\wedge r=0, \quad \text{(S6a)}\\ -f'(1-\theta), & p=1\wedge r=0, \quad \text{(S6b)}\\ g'(\theta), & p=0\wedge r=1, \quad \text{(S6c)}\\ g'(1-\theta), & p=1\wedge r=1, \quad \text{(S6d)}\end{cases}$$

with $f'(\theta)=6\theta^2-6\theta$ and $g'(\theta)=3\theta^2-4\theta+1$ holding. $w'_{q,s}(\theta)$ can be still formalized by replacing indices.

It is clear that all the following equations hold: $f(0)=1$, $f(1)=0$, $g(0)=0$, $g(1)=0$, $f'(0)=0$, $f'(1)=0$, $g'(0)=1$, and $g'(1)=0$, which means that the values of $\boldsymbol{x}$, $\boldsymbol{a}_1$, and $\boldsymbol{a}_2$ on a common edge are merely related to the sampled values at the two end nodes of the edge. This implies C^1-smoothness on the whole surface — the interpolation function itself and its first-order partial derivatives remain continuous across cells, while the second-order partial derivatives come to discontinuity of first kind only at the common edges.

Authors' addresses: Xingyu Ni, Xuwen Chen, Cheng Yu, Peking University, China; Bin Wang, State Key Laboratory of General Artificial Intelligence, BIGAI, China; Baoquan Chen, State Key Laboratory of General Artificial Intelligence, Peking University, China.

A.2 Gauss–Legendre Quadrature

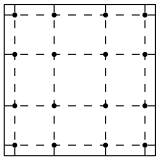


Fig. S1. Quadrature points in a cell.

Any two-dimensional integral that takes the form of

$$I=\int_{\xi^2_{\min}}^{\xi^2_{\max}}\int_{\xi^1_{\min}}^{\xi^1_{\max}} f(\xi^1,\xi^2)\,\mathrm{d}\xi^1\mathrm{d}\xi^2, \tag{S7}$$

can be computed approximately using the summation of n sampled points that

$$I\approx\sum_{i=1}^{n} w_i f(\xi^1_i,\xi^2_i). \tag{S8}$$

We pick $n=16$ in our framework, with the quadrature points arranged by 4×4 in a single cell. As shown in Fig. S1, the vertical dashed lines correspond to

$$\theta^1=\frac{\xi^1_{\min}+\xi^1_{\max}}{2}\pm\frac{\Delta\xi^1}{2}\sqrt{\frac{3}{7}\pm\frac{2}{7}\sqrt{\frac{6}{5}}}, \tag{S9}$$

respectively, and the horizontal dashed lines correspond to

$$\theta^2=\frac{\xi^2_{\min}+\xi^2_{\max}}{2}\pm\frac{\Delta\xi^2}{2}\sqrt{\frac{3}{7}\pm\frac{2}{7}\sqrt{\frac{6}{5}}}, \tag{S10}$$

respectively. The inside vertical lines equip a weight factor of $(18+\sqrt{30})\Delta\xi^1/72$, while the outside ones equip a weight factor of $(18-\sqrt{30})\Delta\xi^1/72$. The inside horizontal lines equip a weight factor of $(18+\sqrt{30})\Delta\xi^2/72$, while the outside ones equip a weight factor of $(18-\sqrt{30})\Delta\xi^2/72$. The final weight w_i of each point is the product of the two weights of the vertical and horizontal lines it lies on.

B DERIVATIVES OF THE ELASTIC ENERGY

We bypass the medium of deformation gradients and compute the first- and second-order partial derivatives of V_e w.r.t. nodal degrees of freedom $\{\boldsymbol{q}_I\}$ directly.

B.1 The First-Order Derivatives (Force)

With the total elastic potential energy defined by $V_\mathrm{e} = \iint_{\bar{\Omega}} \bar{\mathcal{V}}_\mathrm{e}\,\mathrm{d}\bar{\Omega}$, according to Eq. (11) in the main text, the first-order partial derivatives of the elastic energy are given by

$$\begin{aligned}\frac{\partial V_\mathrm{e}}{\partial \boldsymbol{q}_I} &= \iint_{\bar{\Omega}} \frac{\partial \bar{\mathcal{V}}_\mathrm{e}}{\partial \boldsymbol{q}_I}\,\mathrm{d}\bar{\Omega} \\ &= \iint_{\omega} \frac{\partial \bar{\mathcal{V}}_\mathrm{e}}{\partial \boldsymbol{q}_I}\sqrt{\bar{a}}\,\mathrm{d}\xi^1\mathrm{d}\xi^2 \\ &= \iint_{\omega} \left(\tau \frac{\partial a_{\alpha\beta}}{\partial \boldsymbol{q}_I} A_{\gamma\delta} + \frac{1}{3}\tau^3 \frac{\partial b_{\alpha\beta}}{\partial \boldsymbol{q}_I} B_{\gamma\delta}\right) \bar{H}^{\alpha\beta\gamma\delta}\sqrt{\bar{a}}\,\mathrm{d}\xi^1\mathrm{d}\xi^2, \end{aligned} \tag{S11}$$

with partial derivatives of $a_{\alpha\beta}$ and $b_{\alpha\beta}$ are calculated by

$$\frac{\partial a_{\alpha\beta}}{\partial \boldsymbol{q}_I} = \Phi^I_{,\alpha}\boldsymbol{a}_\beta + \Phi^I_{,\beta}\boldsymbol{a}_\alpha, \tag{S12}$$

$$\begin{aligned}\frac{\partial b_{\alpha\beta}}{\partial \boldsymbol{q}_I} =& \Phi^I_{,\alpha\beta}\boldsymbol{a}_3 + \frac{1}{\sqrt{a}}\Big(\Phi^I_{,1}\boldsymbol{a}_2 \times \boldsymbol{a}_{\alpha,\beta} + \Phi^I_{,2}\boldsymbol{a}_{\alpha,\beta} \times \boldsymbol{a}_1 \\ &- \boldsymbol{a}_{\alpha,\beta}\cdot\boldsymbol{a}_3(\Phi^I_{,1}\boldsymbol{a}_2 \times \boldsymbol{a}_3 + \Phi^I_{,2}\boldsymbol{a}_3 \times \boldsymbol{a}_1)\Big). \end{aligned} \tag{S13}$$

In order to facilitate numerical calculation, we rewrite Eq. (S11) in a matrix form using *Voigt notation* as

$$\frac{\partial V_\mathrm{e}}{\partial \boldsymbol{q}_I} = \iint_{\omega} \left(\tau \left(\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{q}_I}\right)^\mathrm{T} \boldsymbol{H}\boldsymbol{\alpha} + \frac{1}{12}\tau^3 \left(\frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{q}_I}\right)^\mathrm{T} \boldsymbol{H}\boldsymbol{\beta}\right)\sqrt{\bar{a}}\,\mathrm{d}\xi^1\mathrm{d}\xi^2, \tag{S14}$$

where $\boldsymbol{H} = (H^{ij})_{3\times3}$ is a square matrix, and $\boldsymbol{\alpha} = (\alpha_i)_{3\times1}$ and $\boldsymbol{\beta} = (\beta_i)_{3\times1}$ are column vectors. It should be noted that a mixed layout is used here — first-order partial derivatives are always written as column vectors. Considering the symmetry of the quantities, the matrix and vectors in this integral can be written as

$$\boldsymbol{H} = \begin{pmatrix} (\lambda+2\mu)(\bar{a}^{11})^2 & \lambda\bar{a}^{11}\bar{a}^{22} + 2\mu(\bar{a}^{12})^2 & (\lambda+2\mu)\bar{a}^{11}\bar{a}^{12} \\ & (\lambda+2\mu)(\bar{a}^{22})^2 & (\lambda+2\mu)\bar{a}^{12}\bar{a}^{22} \\ \text{sym.} & & (\lambda+\mu)(\bar{a}^{12})^2 + \mu\bar{a}^{11}\bar{a}^{22} \end{pmatrix}, \tag{S15}$$

$$\boldsymbol{\alpha} = \begin{pmatrix} A_{11} & A_{22} & 2A_{12} \end{pmatrix}^\mathrm{T}, \tag{S16}$$

$$\boldsymbol{\beta} = 2\begin{pmatrix} B_{11} & B_{22} & 2B_{12} \end{pmatrix}^\mathrm{T}, \tag{S17}$$

$$\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{q}_I} = \frac{1}{2}\begin{pmatrix} \frac{\partial a_{11}}{\partial \boldsymbol{q}_I} & \frac{\partial a_{22}}{\partial \boldsymbol{q}_I} & 2\frac{\partial a_{12}}{\partial \boldsymbol{q}_I} \end{pmatrix}^\mathrm{T}, \tag{S18}$$

$$\frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{q}_I} = \begin{pmatrix} \frac{\partial b_{11}}{\partial \boldsymbol{q}_I} & \frac{\partial b_{22}}{\partial \boldsymbol{q}_I} & 2\frac{\partial b_{12}}{\partial \boldsymbol{q}_I} \end{pmatrix}^\mathrm{T}. \tag{S19}$$

B.2 The Second-Order Derivatives (Hessian)

The Hessian matrix is given by

$$\begin{aligned}\frac{\partial^2 V_\mathrm{e}}{\partial \boldsymbol{q}_J \partial \boldsymbol{q}_I} &= \iint_{\bar{\Omega}} \frac{\partial^2 \bar{\mathcal{V}}_\mathrm{e}}{\partial \boldsymbol{q}_J \partial \boldsymbol{q}_I}\,\mathrm{d}\bar{\Omega} \\ &= \iint_{\omega} \left(\tau\,\boldsymbol{G}_1 + \frac{1}{12}\tau^3\,\boldsymbol{G}_2\right)\sqrt{\bar{a}}\,\mathrm{d}\xi^1\mathrm{d}\xi^2, \end{aligned} \tag{S20}$$

in which the following equations hold:

$$\boldsymbol{G}_1 = \left(\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{q}_I}\right)^\mathrm{T} \boldsymbol{H}\left(\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{q}_J}\right) + \sum_{i=1}^{3}\sum_{j=1}^{3} \frac{\partial^2 \alpha_i}{\partial \boldsymbol{q}_J \partial \boldsymbol{q}_I} H^{ij}\alpha_j, \tag{S21}$$

$$\boldsymbol{G}_2 = \left(\frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{q}_I}\right)^\mathrm{T} \boldsymbol{H}\left(\frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{q}_J}\right) + \sum_{i=1}^{3}\sum_{j=1}^{3} \frac{\partial^2 \beta_i}{\partial \boldsymbol{q}_J \partial \boldsymbol{q}_I} H^{ij}\beta_j. \tag{S22}$$

Therefore, the remaining work is to calculate $\partial^2\alpha_i/\partial\boldsymbol{q}_J\partial\boldsymbol{q}_I$ and $\partial^2\beta_i/\partial\boldsymbol{q}_J\partial\boldsymbol{q}_I$. The concrete form of the former can be easily deduced as

$$\frac{\partial^2 \alpha_1}{\partial \boldsymbol{q}_J \partial \boldsymbol{q}_I} = \Phi^I_{,1}\Phi^J_{,1}\boldsymbol{I}, \tag{S23}$$

$$\frac{\partial^2 \alpha_2}{\partial \boldsymbol{q}_J \partial \boldsymbol{q}_I} = \Phi^I_{,2}\Phi^J_{,2}\boldsymbol{I}, \tag{S24}$$

$$\frac{\partial^2 \alpha_3}{\partial \boldsymbol{q}_J \partial \boldsymbol{q}_I} = \left(\Phi^I_{,1}\Phi^J_{,2} + \Phi^I_{,2}\Phi^J_{,1}\right)\boldsymbol{I}, \tag{S25}$$

while it takes some time to compute the latter, which satisfies

$$\begin{aligned} 2\frac{\partial^2 B_{\alpha\beta}}{\partial \boldsymbol{q}_J \partial \boldsymbol{q}_I} =& \frac{\partial^2 b_{\alpha\beta}}{\partial \boldsymbol{q}_J \partial \boldsymbol{q}_I} \\ =& \Phi^I_{,\alpha\beta}\frac{\partial \boldsymbol{a}_3}{\partial \boldsymbol{q}_J} + \Phi^J_{,\alpha\beta}\left(\frac{\partial \boldsymbol{a}_3}{\partial \boldsymbol{q}_I}\right)^\mathrm{T} + \Phi^I_{,1}\Phi^J_{,1}\boldsymbol{D}^{11} \\ &+ \Phi^I_{,2}\Phi^J_{,2}\boldsymbol{D}^{22} + \Phi^I_{,1}\Phi^J_{,2}\boldsymbol{D}^{12} + \Phi^I_{,2}\Phi^J_{,1}\boldsymbol{D}^{21}, \end{aligned} \tag{S26}$$

in which coefficients of the first-order terms are

$$\frac{\partial \boldsymbol{a}_3}{\partial \boldsymbol{q}_J} = \frac{1}{\sqrt{a}}\left(-\Phi^J_{,1}[\boldsymbol{a}_2] + \Phi^J_{,2}[\boldsymbol{a}_1] - \boldsymbol{a}_3 \otimes (\Phi^J_{,1}\boldsymbol{t}_1 + \Phi^J_{,2}\boldsymbol{t}_2)\right), \tag{S27}$$

$$\frac{\partial \boldsymbol{a}_3}{\partial \boldsymbol{q}_I} = \frac{1}{\sqrt{a}}\left(-\Phi^I_{,1}[\boldsymbol{a}_2] + \Phi^I_{,2}[\boldsymbol{a}_1] - \boldsymbol{a}_3 \otimes (\Phi^I_{,1}\boldsymbol{t}_1 + \Phi^I_{,2}\boldsymbol{t}_2)\right), \tag{S28}$$

and coefficients of the second-order terms are respectively calculated by dot products of $\boldsymbol{a}_{\alpha,\beta}$ with $\partial^2\boldsymbol{a}_3/\partial{\boldsymbol{a}_1}^2$, $\partial^2\boldsymbol{a}_3/\partial{\boldsymbol{a}_2}^2$, $\partial^2\boldsymbol{a}_3/\partial\boldsymbol{a}_2\partial\boldsymbol{a}_1$, and $\partial^2\boldsymbol{a}_3/\partial\boldsymbol{a}_1\partial\boldsymbol{a}_2$. For calculation, we provide relatively simple formulae as follows:

$$\begin{aligned}\boldsymbol{D}^{11} =& \frac{1}{a}\Big(\beta_{\alpha\beta}(3\boldsymbol{t}_1 \otimes \boldsymbol{t}_1 + \boldsymbol{a}_2 \otimes \boldsymbol{a}_2 - a_{22}\boldsymbol{I}) \\ &-\boldsymbol{s}_2 \otimes \boldsymbol{t}_1 - \boldsymbol{t}_1 \otimes \boldsymbol{s}_2\Big), \end{aligned} \tag{S29}$$

$$\begin{aligned}\boldsymbol{D}^{22} =& \frac{1}{a}\Big(\beta_{\alpha\beta}(3\boldsymbol{t}_2 \otimes \boldsymbol{t}_2 + \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 - a_{11}\boldsymbol{I}) \\ &-\boldsymbol{s}_1 \otimes \boldsymbol{t}_2 - \boldsymbol{t}_2 \otimes \boldsymbol{s}_1\Big), \end{aligned} \tag{S30}$$

$$\begin{aligned}\boldsymbol{D}^{12} =& \frac{1}{a}\Big(\beta_{\alpha\beta}(3\boldsymbol{t}_1 \otimes \boldsymbol{t}_2 - 2\boldsymbol{a}_1 \otimes \boldsymbol{a}_2 + \boldsymbol{a}_2 \otimes \boldsymbol{a}_1 + a_{12}\boldsymbol{I}) \\ &-\boldsymbol{s}_2 \otimes \boldsymbol{t}_2 - \boldsymbol{t}_1 \otimes \boldsymbol{s}_1 - \sqrt{a}[\boldsymbol{a}_{\alpha,\beta}]\Big), \end{aligned} \tag{S31}$$

$$\begin{aligned}\boldsymbol{D}^{21} =& \frac{1}{a}\Big(\beta_{\alpha\beta}(3\boldsymbol{t}_2 \otimes \boldsymbol{t}_1 - 2\boldsymbol{a}_2 \otimes \boldsymbol{a}_1 + \boldsymbol{a}_1 \otimes \boldsymbol{a}_2 + a_{12}\boldsymbol{I}) \\ &-\boldsymbol{s}_1 \otimes \boldsymbol{t}_1 - \boldsymbol{t}_2 \otimes \boldsymbol{s}_2 + \sqrt{a}[\boldsymbol{a}_{\alpha,\beta}]\Big), \end{aligned} \tag{S32}$$

where t_1, t_2, s_1, and s_2 are defined as

$$\boldsymbol{t}_1 = \boldsymbol{a}_2 \times \boldsymbol{a}_3, \tag{S33}$$

$$\boldsymbol{t}_2 = \boldsymbol{a}_3 \times \boldsymbol{a}_1, \tag{S34}$$

$$\boldsymbol{s}_1 = \boldsymbol{a}_{\alpha,\beta} \times \boldsymbol{a}_1, \tag{S35}$$

$$\boldsymbol{s}_2 = \boldsymbol{a}_2 \times \boldsymbol{a}_{\alpha,\beta}. \tag{S36}$$

Here we use $[\cdot]$ to represent the cross product matrix of a vector, which is defined as

$$\left[\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}\right] = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix}. \tag{S37}$$

The pseudo Hessian. In the calculation of the Hessian matrix, the term $\partial^2 \beta_i / \partial \boldsymbol{q}_J \partial \boldsymbol{q}_I$ takes up the most time. Thanks to the low magnitude of this term, we can subtract it to construct an inexact Hessian matrix, which significantly reduces the time consumption per step. Note that an inexact Hessian in general shows better positiveness than its exact version, and thus it may also reduce the number of iterations in certain cases.