An Asymptotic Numerical Method for Inverse Elastic Shape Design

Xiang Chen* Changxi Zheng† Weiwei Xu‡ Kun Zhou*§

*State Key Lab of CAD&CG, Zhejiang University †Columbia University ‡Hangzhou Normal University

Abstract

Inverse shape design for elastic objects greatly eases the design efforts by letting users focus on desired target shapes without thinking about elastic deformations. Solving this problem using classic iterative methods (e.g., Newton-Raphson methods), however, often suffers from slow convergence toward a desired solution. In this paper, we propose an asymptotic numerical method that exploits the underlying mathematical structure of specific nonlinear material models, and thus runs orders of magnitude faster than traditional Newton-type methods. We apply this method to compute rest shapes for elastic fabrication, where the rest shape of an elastic object is computed such that after physical fabrication the real object deforms into a desired shape. We illustrate the performance and robustness of our method through a series of elastic fabrication experiments.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling;

Keywords: elastic fabrication, 3D printing, finite element methods, nonlinear optimization

Links: DL PDF VIDEO CODE

1 Introduction

Elastic objects are ubiquitous in computer animation and video games [Nealen et al. 2006], and have received increasing attention in 3D fabrication of real-world objects [Skouras et al. 2013]. However, designing a desired static equilibrium shape of an elastic object is counterintuitive, because it deforms in various ways under external forces. When designing the static shapes of elastic objects, one wishes to focus on desired target shapes directly without thinking about possible deformations. This motivates the development of an inverse shape design tool automatically computing a rest shape that deforms into a desired target shape under given external forces (see Figure 1(a-c)).

In general, the inverse shape design problem amounts to solving a static equilibrium equation,

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}) + \boldsymbol{g} = \boldsymbol{0}, \tag{1}$$

*xchen.cs@gmail.com, kunzhou@acm.org
†cxz@cs.columbia.edu
‡weiwei.xu.g@gmail.com
§Corresponding author

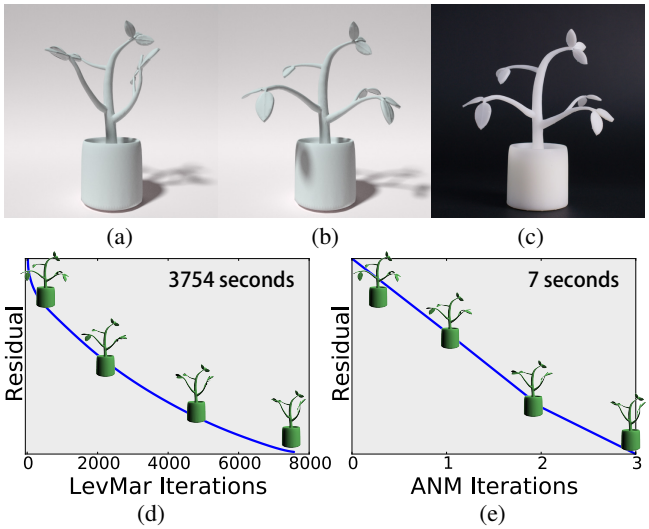

Figure 1: ***Plant:*** *Top: Our method computes the rest shape of a plant model (a), given its desired target shape (b) under gravity. The fabricated object according to the computed rest shape deforms under gravity into a shape (c) that is nearly identical to the desired design shape (b). Bottom: Compared with the Newton-type methods such as the Levenberg-Marquardt solver (d), our ANM method (e) runs orders of magnitude faster.*

where $\boldsymbol{f}$ describes the internal elastic force determined by the material's constitutive model; the deformed shape $\boldsymbol{x}$ and the external force $\boldsymbol{g}$ are the inputs, and $\boldsymbol{X}$ is the rest shape to be found. For internal force models typically encountered in realistic simulation and predictive computational design tasks, the relationship between the rest shape and the deformed shape can be very complex, exhibiting a high degree of nonlinearity.

A number of works have recently investigated this problem to control the shape and motion of elastic materials. For applications of animation control, initial shapes are often computed using fast models such as spring-mass systems [Twigg and Kačić-Alesić 2011], which are efficient but less accurate, and thus not applicable to predictive computational design tasks such as those in 3D fabrication. For rod-like geometries, solutions can be efficiently found using reduced kinematic models (e.g., [Hadap 2006; Derouet-Jourdan et al. 2010; Derouet-Jourdan et al. 2013]), yet no such method exists for general volumetric elastic objects. Thus existing methods for computational design and 3D fabrication (e.g., [Bickel et al. 2012; Skouras et al. 2012]) typically formulate the rest shape computation as a nonlinear (constraint) optimization problem solved by a Newton-type iterative solver. Newton-type methods are known to converge quadratically in the vicinity of a minimum. However, they suffer from slow convergence toward a desired solution, if the problem is highly nonlinear and the initial guess is far away from the solution. Our work is also motivated by applications of 3D elastic fabrication, in which a nonlinear material model is needed for accurate prediction and a good initial guess of the rest shape is often infeasible. We therefore desire a method that is fast, robust and accurate for the computation of elastic rest shapes.

In this paper, we propose to solve the problem of inverse elastic

shape design using the *Asymptotic Numerical Method* (ANM), first introduced in the 1990s [Damil and Potier-Ferry 1990; Cochelin 1994]. ANMs are fundamentally different from traditional Newton-type methods. In a nutshell, the ANM follows a nonlinear solution branch in a stepwise manner: in our case, it continuously changes the external force from 0 to $\boldsymbol{g}$, while tracking the nonlinear solution. However, to harness the idea of ANM for inverse elastic design, a critical mathematical ingredient is lacking. We need to explore the underlying nonlinear structure of the *inverse* internal force function $\boldsymbol{f}$ to develop a local asymptotic expansion, which is critical to enable the ANM to converge quickly and robustly. Unfortunately, for the inverse static equilibrium problem, no such mathematical derivation has yet been developed. As a primary contribution of this paper, we fully develop an asymptotic expansion form for the hyperelastic *neo-Hookean* model, an accurate model that has been widely used in elastic fabrication applications (e.g., [Bickel et al. 2010; Skouras et al. 2013]). The resulting algorithm is highly efficient (see Figure 1(e)), resulting in roughly two orders of magnitude speedups over a traditional Newton-type solver for all our examples (see Figure 1(d)).

We apply our method to compute rest shapes for elastic fabrication. Our design tool computes a rest shape ready for physical fabrication, allowing an end-to-end integration between the shape design step and the fabrication process. We conducted a series of inverse design experiments. The simulation and fabrication results demonstrate the high performance and robustness of our ANM method.

In summary, the main contributions of our work include:

- We derive an asymptotic local expansion for the inverse elastic force of the nonlinear neo-Hookean material model, which allows us to build a fast and robust numerical method to solve the inverse static equilibrium problem.
- We analyze the performance of ANM and compare it to Newton-type methods. Relying on its performance of nonlinear solves, we develop an interactive tool for inverse shape design tasks. And we fabricate the designed shapes to demonstrate its accuracy for realizing desired target shapes.
- We further extend our method to support interactive force adjustment, multi-target inverse shape design as well as forward static equilibrium solves.

2 Related Work

Inverse Shape Design for Computer Animation. There have been many works on designing and optimizing the initial shapes of various deformable models. Twigg and Kačić-Alesić [2011] estimated rest length parameters of a spring-mass system to approximately compensate the mesh sagging effect under gravity. They also considered the contact forces as penalty terms in their optimization. Hadap [2006] computed zero-gravity rest shapes of strands using a multi-body reduced model. Derouet-Jourdan et al. [2010] proposed an inverse design method for 2D dynamic curves to convert a user-input sketch into a dynamic rod model and then optimize the natural curvatures to achieve certain equilibrium states. Derouet-Jourdan et al. [2013] proposed a constrained optimization method for solving the inverse static equilibrium problem of hairs subject to gravity and frictional contacts. These methods were specifically designed for slender structures. In contrast, in our work we are interested in general volumetric objects. Furthermore, as our main target is for fabrication and not mere simulation, we consider not only geometric nonlinearities in the model, but also nonlinearities coming from the hyperelastic properties of fabrication materials. To tackle a fully nonlinear problem, we exploit the mathematical structure of the hyperelastic *neo-Hookean* model to build a fast and robust inverse shape solver based on the asymptotic numerical method. Thus our method differs fundamentally from the Newton-type methods used in most of previous works.

Deformation Control. Controlling desired deformation behaviors has been an active topic in computer animation. For example, to ease the animation production, many algorithms generate output animations with prescribed shape deformations at key frames [Barbič et al. 2009; Barbič et al. 2012b; Hildebrandt et al. 2012]. These methods often rely on reduced models to approximate the dynamics and improve the performance, because they aim to generate plausible rather than physically predictive animations, and need to control a whole sequence of the resulting animation. In addition, Martin et al. [2011] compute an intermediate rest configuration based on provided examples to control object deformations. Coros et al. [2012] control the motion of active deformable characters by computing appropriate rest poses to drive an object's motion. In this paper, we desire a method that computes a physically reproducible elastic shape. We are particularly interested in the deformed target shape, and thus focus on the static equilibrium problem.

Fabrication-aware Design. There have been numerous computational tools for designing geometric shapes and materials in applications of 3D printing. Their goal is to produce geometries, mechanical structures and material distributions satisfying fabrication constraints and desired physical properties such as shape balance in a desired pose [Prévost et al. 2013], stress distributions [Stava et al. 2012], articulated characters [Bächer et al. 2012; Calì et al. 2012], and motion sequences [Coros et al. 2013; Ceylan et al. 2013]. Closely related to our work, Skouras et al. [2012] presented a computational design tool for rubber balloons, which estimates a rest shape to be inflated into a given target shape. While they also formulate a static equilibrium problem, their method considers forces due to an elastic membrane model and pressure, whereas we focus on forces due to volumetric elasticity and gravity. Moreover, they use a standard Newton-type method to minimize an energy function, but we introduce a new numerical solver using the ANM.

Bickel et al. [2010] developed a branch-and-bounding algorithm to combine basis materials such as to produce desired elastic properties. Physical face cloning [Bickel et al. 2012] optimizes geometric, physical and actuation parameters to produce skins with prescribed deformation behavior. Chen et al. [2013] proposed a reducer-tuner model to tune the material nodes for 3D printing. And Skouras et al. [2013] optimized both actuator positions and material distribution to control the shapes of deformable characters. Most of these methods use neo-Hookean materials for elastic simulation. While our method also uses the neo-Hookean model to predict the deformation of fabricated objects, we focus on the inverse shape design with a fixed material. The undeformed rest shape is unknown in our problem, while the deformed target shape is given. Moreover, many of these methods formulate an optimization problem with a customized energy function. Our method, in contrast, simply solves a nonlinear equation (1) in an efficient and robust way.

Numerical Continuation Methods. Our proposed asymptotic numerical method follows the basic idea of path following as widely used in traditional numerical continuation methods (see an introduction in [Allgower and Georg 1990]). Classic methods include the Predictor-Corrector method and the Piecewise-Linear method. Both types of methods follow a nonlinear solution branch in a stepwise manner. Although widely used now, these methods have difficulties to determine the iteration step size, which is usually fixed *a priori*. A small step size leads to a slow convergence, while a large step size compromises the accuracy of the results. Our method is similar to [Cochelin 1994], which adaptively selects the step size as large as possible while retaining sufficient accuracy to ensure quadratic convergence at each step. Our key contribution is the development

of asymptotic expansions for the inverse neo-Hookean model.

3 Background

We start our technical presentation from a formal problem formulation. Consider a deformable object represented using a finite element (FE) model, in which a closed surface mesh is discretized into a tetrahedral mesh. We describe the position of the tetrahedral mesh using a vector $\boldsymbol{x}$ that stacks the positions of all the tetrahedral nodes. For a model with N nodes, $\boldsymbol{x}$ has a length of $3N$. Following the notations in continuum mechanics, we use lowercase $\boldsymbol{x}$ to describe the deformed shape (i.e., the input shape) and the uppercase $\boldsymbol{X}$ to denote the undeformed shape (i.e., the output shape). Moreover, we denote a node of the tetrahedral mesh as $\boldsymbol{x}_i$, where $i = 1...N$ is its index.

Static Equilibrium Problem and its Inverse. Given a rest state $\boldsymbol{X}$ of an elastic object, the traditional static equilibrium problem seeks a deformed state $\boldsymbol{x}$ under external forces $\boldsymbol{g}$ to satisfy the static equilibrium equation,

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}) + \boldsymbol{g} = \boldsymbol{0}, \tag{2}$$

where $\boldsymbol{g}$ is a vector of length $3N$, stacking the external forces applied on all the tetrahedral nodes; provided the rest and deformed states, $\boldsymbol{f}$ computes the internal force, whose specific form depends on the considered material model.

The *inverse* problem of Eq. (2) is to solve for the rest state $\boldsymbol{X}$, given the user-specified target state $\boldsymbol{x}$ under external force vector $\boldsymbol{g}$ (see Figure 2 for an example).

Neo-Hookean Constitutive Model. We use the *neo-Hookean* model [Bonet and Wood 1997; Ogden 1997], a hyperelastic material model, to compute the internal force function $\boldsymbol{f}$. It has been widely used in continuum mechanics to predict nonlinear elastic behavior of materials undergoing large deformations as encountered in our case, and has proven a plausible prediction of elastic fabrication in previous works (e.g., [Skouras et al. 2013]).

The neo-Hookean model describes the relationship between geometric deformation and internal strain force in a highly nonlinear way. In particular, the strain energy density function is modeled using the function,

$$W(\boldsymbol{x}, \boldsymbol{X}) = \left(\frac{\mu}{2}(J^{-\frac{2}{3}} I_c - 3) + \frac{\kappa}{2}(J-1)^2\right). \tag{3}$$

Here μ and κ are coefficients describing the material's resistance to shearing and volume change respectively. J and I_c are deformation related quantities: let $\boldsymbol{F}$ denote the deformation gradient [Bonet and Wood 1997] (i.e., $\boldsymbol{F} = \frac{d\boldsymbol{x}}{d\boldsymbol{X}}$), and let $\boldsymbol{C} = \boldsymbol{F}^T\boldsymbol{F}$ denote the right Cauchy-Green deformation tensor. $J = \det(\boldsymbol{F})$ is the determinant of $\boldsymbol{F}$, and $I_c = \mathrm{Tr}(\boldsymbol{C})$, the so-called first invariant of the Cauchy-Green tensor, is the trace of $\boldsymbol{C}$.

The constitutive model defines the internal force function $\boldsymbol{f}$. We refer the reader to the textbook [Bonet and Wood 1997] for a detailed derivation, and only sketch the basic steps here. We first derive the second Piola-Kirchhoff stress tensor,

$$\boldsymbol{S} = 2\frac{\partial W}{\partial \boldsymbol{C}} = \mu J^{-\frac{2}{3}}\boldsymbol{I} - \frac{\mu}{3}J^{-\frac{2}{3}} I_c \boldsymbol{C}^{-1} + \kappa(J-1)J\boldsymbol{C}^{-1}, \tag{4}$$

where $\boldsymbol{I}$ is the 3×3 identity matrix. We then compute the first Piola-Kirchhoff tensor, $\boldsymbol{P} = \boldsymbol{F}\boldsymbol{S}$. Finally, using a piece-wise constant FE approximation, the internal force $\boldsymbol{f}_i$ of a node $\boldsymbol{x}_i$ is $\boldsymbol{f}_i = \sum_{t \in \mathsf{adj}(\boldsymbol{x}_i)} \boldsymbol{P}_t \bar{\boldsymbol{n}}_i^t$, where $t \in \mathsf{adj}(\boldsymbol{x}_i)$ denotes tetrahedra incident to the node $\boldsymbol{x}_i$; $\boldsymbol{P}_t$ is the piece-wise constant first Piola-Kirchhoff stress tensor at t; and $\bar{\boldsymbol{n}}_i^t$ is the effective outward normal of the node $\boldsymbol{x}_i$ at the undeformed tetrahedron t.

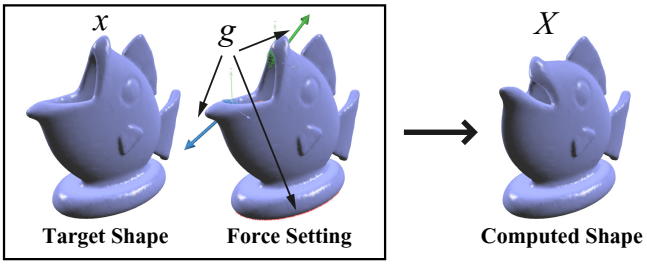

Figure 2: Overview: *The input to inverse elastic shape design includes a desired shape (left) and a set of external forces (middle) to realize the shape. The output is a rest shape (right) that deforms under the given forces into the desired shape.*

Assumptions. Throughout this paper, we assume the user-specified external force always keeps the object deformation in its elastic region. In other words, the object would never reach its yield threshold, and thus our method neglects any plastic and tearing behavior. We also assume the object deformation would not lead to any self-collision.

4 ANM for Inverse Elastic Shape Design

ANM has been successfully applied to many problems from nonlinear structural analysis. In particular, it has been demonstrated that ANM can offer superior performance and robustness over traditional Newton-type methods for highly nonlinear material models [Zahrouni et al. 1999; Lazarus et al. 2013]. ANM in essence is a multi-step iterative method, but very often converges much faster than the Newton-type methods (see Figure 1). In this section, we present our core algorithm to solve the inverse problem. The same algorithm can be applied to solve the static equilibrium problem using a slightly different derivation. We therefore defer the details of the static equilibrium solve to §5.3.

4.1 Asymptotic Numerical Method

Our goal is to solve Eq. (2), in which $\boldsymbol{x}$ is provided by the user, and $\boldsymbol{X}$ is the unknown rest shape. First, consider a parameterized version (a so-called *homotopy*) of Eq. (2),

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}) + \lambda \boldsymbol{g} = \boldsymbol{0}, \tag{5}$$

where λ is a loading parameter in the range $[0, 1]$. When $\lambda = 0$, the solution of Eq. (5) is clearly $\boldsymbol{X} = \boldsymbol{x}$ up to a rigid transformation, since only an undeformed shape produces a vanishing internal force. When $\lambda = 1$, its solution is what we desired, i.e., the solution of Eq. (2). The basic idea of ANM is derived from numerical continuation methods [Allgower and Georg 1990]: in a step-wise manner, it changes the parameter λ by following an implicitly defined curve $\lambda(a)$ starting from $\lambda(0) = 0$. At each step, a new a is selected and a solution $\boldsymbol{X}(a)$ that solves $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}(a)) + \lambda(a)\boldsymbol{g} = 0$ is computed. With a carefully optimized way of selecting a at each step, it changes λ using as few steps as possible until $\lambda(a) = 1$ is reached, and at every step the computation is very efficient.

We note the importance of introducing the variable a to parameterize a curve traveling through the $(\boldsymbol{X}, \lambda)$ space. The reason is that directly increasing λ from 0 to 1 and solving $\boldsymbol{X}$ at each step has proven numerically unstable for nonlinear equations. In proximity of bifurcation points (i.e., the turning points on $\boldsymbol{X}$-λ manifold), a small increase of λ can lead to a dramatic change of the corresponding

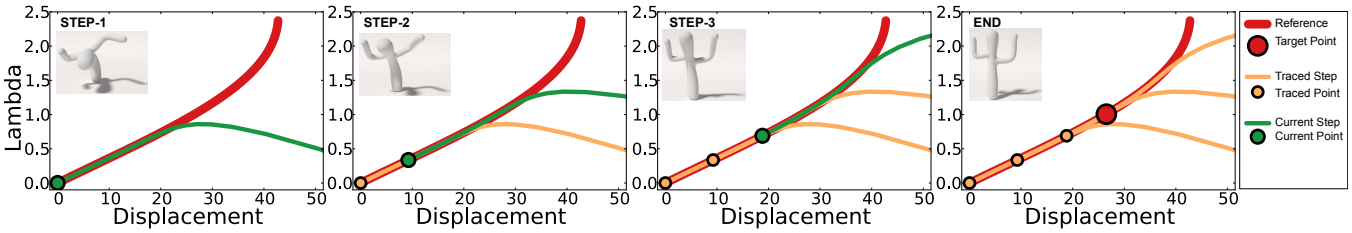

Figure 3: Visualization of ANM Steps: *The x-axis indicates the norm of $\|\boldsymbol{X}(a) - \boldsymbol{x}\|_2$, while the y-axis indicates the corresponding $\lambda(a)$ such that $\boldsymbol{X}(a)$ solves $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}(a)) + \lambda(a)\boldsymbol{g} = 0$. Given a target cactus shape named as* bifur3 *(in the inset), the ANM first computes an asymptotic expansion of $(\boldsymbol{X}(a), \lambda(a))$ (green curve in the first figure) at $a = 0$ to track the solution branch locally. It then changes the value of a along the expansion branch as far as possible, until the convergence radius is reached. From there, it refines the solution and creates a new expansion (green curve in the second figure). This process is repeated, until $\lambda(a) = 1$ is reached.*

solution $\boldsymbol{X}$, causing slow convergence or even instability in the iterative solver. We refer the reader to [Allgower and Georg 1990] for a detailed explanation of the motivation of introducing a.

Tracking Solution using Asymptotic Expansions. Consider a single step of ANM. Let a_0 denote the current parameter value of a step. We have $\lambda_0 = \lambda(a_0)$ and the corresponding solution $\boldsymbol{X}_0$ that satisfies $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}_0) + \lambda_0 \boldsymbol{g} = 0$. Without an explicit definition of $\lambda(a)$, ANM expresses $\lambda(a)$ and its corresponding solution using a power series expansion around a_0

$$\begin{bmatrix} \boldsymbol{X}(a) \\ \lambda(a) \end{bmatrix} \approx \begin{bmatrix} \boldsymbol{X}_0 \\ \lambda_0 \end{bmatrix} + \sum_{k=1}^{n} (a - a_0)^k \begin{bmatrix} \boldsymbol{X}_k \\ \lambda_k \end{bmatrix}, \tag{6}$$

where n is the truncation order; the set of coefficients, $\{\boldsymbol{X}_k, \lambda_k\}, k = 1...n$, are what we need to compute at the current step. After establishing this local power series, we start to change a toward the value satisfying $\lambda(a) = 1$. Inevitably, as we move a away from a_0, the asymptotic expansion of $\boldsymbol{X}(a)$ deviates away from the true solution. Once an estimated residual of the approximated $\boldsymbol{X}(a)$ exceeds a given threshold (detailed in §4.3), we stop the a value there and refine the solution $\boldsymbol{X}$ using a few Newton-Raphson steps (less than 3 in our experiments) initialized with $\boldsymbol{X}(a)$. It is well-known that Newton's method converges quadratically if the initial guess is close enough to a solution. In fact, that is always the case here, since the initial $\boldsymbol{X}(a)$ resulting from the evaluation of the power series is already quite close to the true solution. After finding an accurate solution $\boldsymbol{X}(a)$, we move on to a new step, and use $a_0 = a$ there. We repeat these steps until $\lambda(a) = 1$ is reached. At that point, the solution $\boldsymbol{X}(a)$ solves our original Eq. (2). An outline of our ANM algorithm is listed in Algorithm 1, and an example is illustrated in Figure 3.

To complete the details of the ANM process, we need to address two questions: (i) how do we solve for the expansion coefficients efficiently? And (ii) how do we estimate the residual of $\boldsymbol{X}(a)$ in order to decide when to proceed to the next step? We address these questions in the next two subsections.

4.2 Computation of Asymptotic Coefficients

Algorithm 1 ANM Tracing

Set $\boldsymbol{X}_0 = \boldsymbol{x}$, $\lambda_0 = 0$, $a_0 = 0$; {initial starting point}
while $\lambda < 1$ **do**
  Solve the polynomial coefficients $\{\boldsymbol{X}_k, \lambda_k\}, k = 1...n$;
  Calculate reliable change of a based on residual estimation;
  Refine $\boldsymbol{X}(a)$ by Newton-Raphson method;
  Set $\boldsymbol{X}_0 = \boldsymbol{X}(a)$, $\lambda_0 = \lambda(a)$, $a_0 = a$;
end while

4.2.1. Mathematical Insights Before diving into our derivation details of computing the coefficients $\{\boldsymbol{X}_k, \lambda_k\}$ for Eq. (6), we first present the critical insights that lead to fast solves for the coefficients. Suppose for a moment the force function $\boldsymbol{f}$ has a quadratic form of $\boldsymbol{X}$. Namely,

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}) = \boldsymbol{L}_0 + \boldsymbol{L}[\boldsymbol{X}] + \boldsymbol{Q}[\boldsymbol{X}, \boldsymbol{X}], \tag{7}$$

where $\boldsymbol{L}[\star]$ and $\boldsymbol{Q}[\star, \star]$ are respectively a linear and bilinear vector valued operators of vector inputs. Substituting the expansion (6) of $\boldsymbol{X}(a)$ into this expression yields a quadratic series,

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}(a)) &= \boldsymbol{L}_0 + \boldsymbol{L}[\boldsymbol{X}_0] + \boldsymbol{Q}[\boldsymbol{X}_0, \boldsymbol{X}_0] \\ &\quad + (a - a_0)\left(\boldsymbol{L}[\boldsymbol{X}_1] + 2\boldsymbol{Q}[\boldsymbol{X}_0, \boldsymbol{X}_1]\right) \\ &+ \sum_{k=2}^{n} (a - a_0)^k \left(\boldsymbol{L}[\boldsymbol{X}_k] + 2\boldsymbol{Q}[\boldsymbol{X}_0, \boldsymbol{X}_k] + \sum_{t=1}^{k-1} \boldsymbol{Q}[\boldsymbol{X}_t, \boldsymbol{X}_{k-t}] \right). \end{aligned} \tag{8}$$

Here we only keep the resulting expansion terms whose order is no greater than n, since the truncation order of Eq. (6) is n. Recall that we also have an nth-order expansion of $\lambda(a)$ in Eq. (6). Following Cochelin's method [1994], we substitute both expansions in the equation $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}(a)) + \lambda(a)\boldsymbol{g} = 0$, and establish a group of equations by matching the coefficients of every order of $(a - a_0)$. The zeroth-order coefficients match already, since $\boldsymbol{X}_0$ is the solution at λ_0, satisfying $\boldsymbol{L}_0 + \boldsymbol{L}[\boldsymbol{X}_0] + \boldsymbol{Q}[\boldsymbol{X}_0, \boldsymbol{X}_0] + \lambda_0 \boldsymbol{g} = 0$. The first-order coefficients need to satisfy the equation,

$$\boldsymbol{L}[\boldsymbol{X}_1] + 2\boldsymbol{Q}[\boldsymbol{X}_0, \boldsymbol{X}_1] + \lambda_1 \boldsymbol{g} = \boldsymbol{0}. \tag{9}$$

This is a linear system for $\boldsymbol{X}_1$, because when $\boldsymbol{X}_0$ is known, the bilinear form $\boldsymbol{Q}[\boldsymbol{X}_0, \boldsymbol{X}_1]$ becomes linear with respect to $\boldsymbol{X}_1$, i.e., $\boldsymbol{Q}[\boldsymbol{X}_0, \boldsymbol{X}_1] = \mathsf{A}\boldsymbol{X}_1$, where A is a matrix computed by reducing the bilinear operator $\boldsymbol{Q}$ with $\boldsymbol{X}_0$. For any order $k > 1$, if the coefficients $(\boldsymbol{X}_j, \lambda_j)$ for lower order $j = 1...k-1$ are solved, we then have a linear system for $\boldsymbol{X}_k$,

$$\boldsymbol{L}[\boldsymbol{X}_k] + 2\boldsymbol{Q}[\boldsymbol{X}_0, \boldsymbol{X}_k] + \sum_{t=1}^{k-1} \boldsymbol{Q}[\boldsymbol{X}_t, \boldsymbol{X}_{k-t}] + \lambda_k \boldsymbol{g} = \boldsymbol{0}. \tag{10}$$

However, both $\boldsymbol{X}_k$ and λ_k are unknowns, yielding an under-constrained linear system with $3N + 1$ unknowns and $3N$ equations in (10). To get a full-rank system, we introduce one more constraint as suggested by Cochelin et al. [1994],

$$(\boldsymbol{X}(a) - \boldsymbol{X}_0)^T \boldsymbol{X}_1 + (\lambda(a) - \lambda_0)\lambda_1 = a. \tag{11}$$

Essentially, this constraint requires that the parameter a measures the projection of state increment $(\boldsymbol{X} - \boldsymbol{X}_0, \lambda - \lambda_0)$ on the local tangent vector $(\boldsymbol{X}_1, \lambda_1)$. After substituting the power series of $\boldsymbol{X}(a)$

and $\lambda(a)$ into Eq. (11), and equating the coefficients of powers of $(a - a_0)$, we have one more equation for every $(\boldsymbol{X}_k, \lambda_k)$,

$$\boldsymbol{X}_k^T \boldsymbol{X}_1 + \lambda_k \lambda_1 = \delta_{k1}, \quad k = 1...n, \tag{12}$$

where δ_{k1} is the Kronecker delta: $\delta_{k1} = 1$ if $k = 1$, and zero otherwise. When $k = 1$, Eq. (12) requires $[\boldsymbol{X}_1, \lambda_1]$ to have a unit 2-norm. We therefore simply normalize one solution of the under-constrained linear system (9). When $k > 1$, putting the constraint (12) together with Eq. (10) yields a full-rank linear system of $(\boldsymbol{X}_k, \lambda_k)$ (see details in Appendix A).

We note that all the linear systems for $(\boldsymbol{X}_k, \lambda_k)$ are very fast to solve. Indeed, as detailed in Appendix A, all the linear systems,

$$\mathsf{A} \begin{bmatrix} \boldsymbol{X}_k \\ \lambda_k \end{bmatrix} = \boldsymbol{b}_k,$$

share the same matrix A, which is exactly the Jacobian of the nonlinear function $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}) + \lambda \boldsymbol{g}$ at $(\boldsymbol{X}_0, \lambda_0)$. Namely,

$$\mathsf{A} = \frac{\partial}{\partial(\boldsymbol{X}, \lambda)} \left(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}) + \lambda \boldsymbol{g} \right) \Big|_{(\boldsymbol{X}_0, \lambda_0)}. \tag{13}$$

Recall that as described in §4.1, after we change a to a new value, we refine the solution $X(a)$ using the Newton-Raphson method, in which we compute exactly the Jacobian (13) and solve a linear system to advance a Newton's step. We can therefore cache the factorization of A and reuse it for solving the coefficients $(\boldsymbol{X}_k, \lambda_k)$ using fast back substitution.

4.2.2. Construction of Quadratic Form So far, a critical assumption we have relied on is the quadratic force function (7) for constructing linear systems of $(\boldsymbol{X}_k, \lambda_k)$. Unfortunately, the force function $\boldsymbol{f}$ resulting from the neo-Hookean model is not as simple as a quadratic form. However, as a key contribution of this paper, we show below that $\boldsymbol{f}$ can be transformed into a quadratic form by introducing auxiliary variables.

A Simple Example. We illustrate the basic idea using a simple non-quadratic function $f(x) = x^{3/2}$. Suppose x is expressed as a polynomial of a, i.e., $x(a) = \sum_{k=0}^{n} x_k a^k$. Our goal here is to express $f(x)$ also as a polynomial of a. While f is not a quadratic function, we can rewrite it as a quadratic form by introducing a new variable y such that

$$f(x, y) = xy \text{ and } x = y^2.$$

This transformation is not an ad-hoc approximation; it is precisely equivalent to the original $\boldsymbol{f}$. Now if y is also expressed as a polynomial of a (i.e., $y(a) = \sum_{k=0}^{n} y_k a^k$), then substituting the expansions of $x(a)$ and $y(a)$ into $f(x, y)$ yields a power series,

$$f(x, y) = \sum_{k=0}^{n} f_i a^k, \text{ where } f_k = x_0 y_k + x_k y_0 + \sum_{r=1}^{k-1} x_r y_{k-r}. \tag{14}$$

Here again we discard the expansion terms whose orders are greater than n. To establish the relationships between the coefficients x_i and $y_i, i = 0...k$, we substitute both $x(a)$ and $y(a)$ into $x = y^2$. Equating the coefficients of every order of a, we obtain an equation of y_k,

$$x_k = 2 y_0 y_k + \sum_{r=1}^{k-1} y_r y_{k-r}.$$

This equation allows us to express y_k as a *linear* function of x_k, because the relationship between x and y is *quadratic* (i.e., $x = y^2$). Consequently, after substituting y_k into the expansion coefficient f_k in (14), f_k is still linear with respect to x_k. Therefore we can establish order-by-order linear equations for x_k as described in §4.2.1. In summary, this simple example demonstrates that for any nonlinear function f, we can introduce auxiliary variables to transform f into a quadratic form. As long as the auxiliary variables have quadratic relationships, we can express f as a power series of a, such that the kth-order coefficient is linear with respect to x_k. Thus we can compute x_k via a linear solve.

Computing Coefficients $(\boldsymbol{X}_k, \lambda_k)$. Harnessing the same idea, we now compute $(\boldsymbol{X}_k, \lambda_k)$. Here we only sketch the computational steps for a practical implementation, and provide the detailed formulas in Appendix A.

First, since in our inverse problem the deformed state $\boldsymbol{x}$ is given, it is convenient to express the stress tensor in the deformed configuration. In particular, we compute the Cauchy stress tensor based on the expression of the second Piola-Kirchhoff tensor (4):

$$\boldsymbol{\sigma} = J^{-1} \boldsymbol{F} \boldsymbol{S} \boldsymbol{F}^T = \mu J^{-\frac{5}{3}} \boldsymbol{b} - \frac{\mu}{3} J^{-\frac{5}{3}} I_c \boldsymbol{I} + \kappa (J - 1) \boldsymbol{I}. \tag{15}$$

With a piece-wise constant FE approximation, the internal force $\boldsymbol{f}_i$ at a node $\boldsymbol{x}_i$ is

$$\boldsymbol{f}_i = \sum_{t \in \mathsf{adj}(\boldsymbol{x}_i)} \boldsymbol{\sigma}^t \boldsymbol{n}_i^t,$$

where $t \in \mathsf{adj}(\boldsymbol{x}_i)$ again indicates a tetrahedron incident to the node $\boldsymbol{x}_i$; $\boldsymbol{\sigma}^t$ is the piece-wise constant Cauchy stress tensor at t; and $\boldsymbol{n}_i^t$ is the outward normal of the node $\boldsymbol{x}_i$ at the deformed tetrahedron t.

Our goal now is to construct an asymptotic expansion of $\boldsymbol{f}_i$ at every node $\boldsymbol{x}_i$, and apply the expansion in ANM to balance the external force $\boldsymbol{g}$ scaled by λ. Namely,

$$\sum_{t \in adj(\boldsymbol{x}_i)} (\boldsymbol{\sigma}_0^t + \sum_{k=1}^{n} a^k \boldsymbol{\sigma}_k^t) \boldsymbol{n}_i^t + (\lambda_0 + \sum_{k=1}^{n} a^k \lambda_k) \boldsymbol{g}_i = \boldsymbol{0}, \tag{16}$$

where $\{\boldsymbol{\sigma}_k^t, k = 0...n\}$ are the coefficients of the ANM expansion of the Cauchy Stress tenor at a tetrahedron t.

The stress tensor $\boldsymbol{\sigma}$ has a highly nonlinear relationship with $\boldsymbol{X}$ through the constitutive model (15). Following the same idea as in §4.2.2, we introduce a number of auxiliary variables, and eventually compute the k-th order expansion coefficient of $\boldsymbol{\sigma}$ as

$$\begin{aligned} \boldsymbol{\sigma}_k = \mu & \left(s_k^{(5)} \boldsymbol{b}_0 + s_0^{(5)} \boldsymbol{b}_k + \sum_{r=1}^{k-1} s_r^{(5)} \boldsymbol{b}_{k-r} \right) \\ & - \frac{\mu}{3} \boldsymbol{I} \left(s_k^{(5)} (I_c)_0 + s_0^{(5)} (I_c)_k + \sum_{r=1}^{k-1} s_r^{(5)} (I_c)_{k-r} \right) + \kappa J_k \boldsymbol{I}, \end{aligned} \tag{17}$$

where J_k, $(I_c)_k$ and $\boldsymbol{b}_k$ are the k-th order expansion coefficients of expansion of J, I_c and $\boldsymbol{b}$ respectively; and $s_k^{(5)}$ is the k-th order coefficient of the auxiliary variable $s^{(5)} = J^{-\frac{5}{3}}$. Here we omit t for the sake of simplicity. For the inverse problem of neo-Hookean elasticity, we introduced 12 auxiliary variables to transform the nonlinear internal force function into a quadratic form (See detailed formulas in Appendix A). Eventually, $\boldsymbol{\sigma}_k$ is linear with respect to $\boldsymbol{X}_k$, the kth-order expansion coefficient of $\boldsymbol{X}(a)$. We note that these auxiliary variables serve for "chaining" the asymptotic expansions from $\boldsymbol{X}(a)$ to $\boldsymbol{\sigma}^t$. They will not increase the system's DoFs, i.e., they are never involved in the linear solves.

Since we have an asymptotic form to express $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}(a))$ now, we follow exactly the process introduced in §4.2.1 and construct linear equations for solving $(\boldsymbol{X}_k, \lambda_k), k = 1...n$,

$$\mathsf{A} \boldsymbol{X}_k = -\lambda_k \boldsymbol{g} + \boldsymbol{f}_{(k)}^{nl}, \tag{18}$$

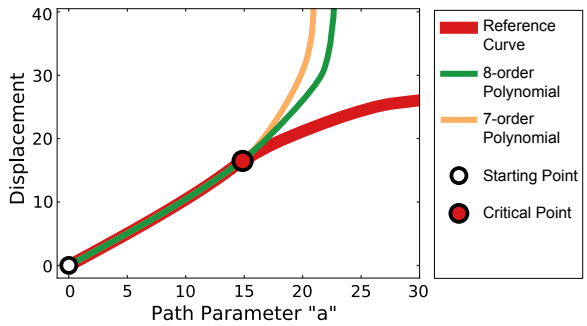

Figure 4: Convergence radius of path parameter a: *We plot $\|\boldsymbol{X}(a)-\boldsymbol{x}\|_2$ as the parameter a changes using an accurate solution curve in red, an 8-order expansion curve in green, and a 7-order expansion curve in orange. Both expansion curves closely track the accurate solution within the convergence radius. Once a critical point is reached, they diverge rapidly.*

where $\boldsymbol{f}^{nl}_{(k)}$ assembles all terms not related to $\boldsymbol{X}_k$. Again, please refer Appendix A for the details of computing A and $\boldsymbol{f}^{nl}_{(k)}$. As introduced in §4.2.1, assembling Eq. (18) together with the linear constraint (12) on curve parameter a leads to a full-rank linear system for $(\boldsymbol{X}_k, \lambda_k)$. Again, A has to be factorized only once to solve Eq. (18) for all k.

4.3 Residual Estimation of $\boldsymbol{X}(a)$

Once a local expansion of $\boldsymbol{X}(a)$ is computed, we can proceed along the curve of $\boldsymbol{X}(a)$ until the approximation error exceeds a given threshold. To this end, we need to quickly estimate the residual of $\boldsymbol{X}(a)$. As suggested by [Cochelin 1994], a simple estimation is motivated by the observation that when $a - a_0$ is within the convergence radius of the power series, the difference between two consecutive approximation orders remains small. However, when $a - a_0$ reaches the convergence radius, they separate rapidly (see Figure 4). Therefore, we estimate the residual as

$$r = \frac{\|\boldsymbol{X}(a)_{\text{order } n} - \boldsymbol{X}(a)_{\text{order } n-1}\|}{\|\boldsymbol{X}(a)_{\text{order } n} - \boldsymbol{X}(0)\|} = \frac{\|\boldsymbol{X}_n(a-a_0)^n\|}{\left\|\sum_{k=1}^{n} \boldsymbol{X}_k(a-a_0)^k\right\|}. \tag{19}$$

As described in §4.1, r serves as an indication to decide how far we move away from a_0 before starting a new step. In practice, we use $r \leq 1E-6$ and estimate the convergence radius a_r as

$$a_r = \left(r\frac{\|\boldsymbol{X}_1\|}{\|\boldsymbol{X}_n\|}\right)^{\frac{1}{n-1}}. \tag{20}$$

If the power series $\lambda(a)$ reaches the value 1 for $a \in [a_0, a_0+a_r]$, the ANM terminates with the solution $\boldsymbol{X}(a)$. Evaluating this condition amounts to a polynomial root finding of $\lambda(a) = 1$, for which we use Brent's method [Brent 2013].

4.4 Discussion

We close this section by remarking the performance of our method. ANM is a multi-step method, similar to Newton-type methods. However, as shown in our experiments (see Table 2), it significantly outperforms Newton-type methods for the problems in our work. The reasons are threefold: (i) it requires no user-provided initial guess; when $a = 0$, the solution $\boldsymbol{X}(a) = \boldsymbol{x}$ is always guaranteed. (ii) The asymptotic local expansion at a point a_0 essentially constructs a high-order "tangential" space similar to the gradient computed in Newton's methods. Therefore, even with a relatively large change, $a - a_0$, the expansion $\boldsymbol{X}(a)$ remains close to the true solution, thus allowing for large steps. (iii) Our method ensures that $\boldsymbol{X}(a)$ stays close to the true solution at all times. Consequently, the Newton iterations for refining the approximation at each step converge rapidly. Moreover, the linear system factorization computed during the Newton steps can be reused for quick computation of expansion coefficients $(\boldsymbol{X}_k, \lambda_k)$.

5 Extensions

After laying out our core algorithm for the inverse elastic shape solves, we now proceed to extend it to an interactive tool for the tasks of inverse shape design.

5.1 Interactive Force Adjustment

An additional attractive feature of ANM is that once the solution is found, it allows the user to interactively adjust the external force and update the solution instantly. Our basic idea here is to use an asymptotic expansion to quickly update the rest shape $\boldsymbol{X}$ resulting from different force scale λ.

In particular, recall that at the end of our ANM steps, we find a rest shape $\boldsymbol{X}$ that solves $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}) + \boldsymbol{g} = 0$. We repeat one more step, and compute the expansion coefficients of Eq. (6) at $(\boldsymbol{X}, 1)$ as presented in §4.2. With this asymptotic expansion at $(\boldsymbol{X}, 1)$, our system allows the user to adjust the external force scale by changing the λ value. Suppose that the updated force is $\bar{\lambda}\boldsymbol{g}$. We first solve the parameter a using the second equation of Eq. (6),

$$\bar{\lambda} = 1 + \sum_{k=1}^{n} (a - a_1)^k \lambda_k,$$

where $\lambda(a_1) = 1$. This amounts to solving a polynomial root finding problem. Among the multiple roots, we use the one closest to a_1. Lastly, we substitute the resulting value of a into the first equation of Eq. (6) to update the rest shape $\boldsymbol{X}$. The entire computation involves only a polynomial root finding for a and a subsequent evaluation of expansion series, both of which can be performed interactively.

As the user-adjusted $\bar{\lambda}$ deviates from $\lambda = 1$, the estimation of $\boldsymbol{X}$ using the expansion degrades. Using the same residual estimation as in §4.3, we start a new ANM step to update $\boldsymbol{X}$ when the residual estimation exceeds a threshold $\epsilon = 1E-6$.

One limitation of this scheme is that it allows only the change of the force scale λ but not its direction. Our system allows the user to adjust the force direction with slightly higher computational expense. When the user updates the force direction, changing the force from $\boldsymbol{g}$ to $\boldsymbol{g}_1$, we start a new ANM process to track the solution with a parameterized inverse equilibrium equation,

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}) + \lambda(\boldsymbol{g}_1 - \boldsymbol{g}) + \boldsymbol{g} = \boldsymbol{0}.$$

When $\lambda = 0$, the solution that solves $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}) + \boldsymbol{g} = 0$ is known. And we use ANM to track the solution until $\lambda = 1$ using the method in §4. If the updated force $\boldsymbol{g}_1$ is sufficiently close to $\boldsymbol{g}$, we can still provide interactive feedbacks. Otherwise, the user needs to wait for a few seconds to see the shape update.

5.2 Multi-Target Inverse Shape Design

Our inverse design system thus far takes as the input a single target shape $\boldsymbol{x}$ and applied force $\boldsymbol{g}$. However, in practice, as a fabricated object may function in multiple ways, it is often desirable to apply different forces on an object to get different deformed shapes (see Figure 9). Formally, given T target shapes $\boldsymbol{x}_t, t = 1...T$ along with the corresponding external forces $\boldsymbol{g}_t, t = 1...T$, we seek a *single* rest pose $\bar{\boldsymbol{X}}$ for physical fabrication such that with each force $\boldsymbol{g}_t$,

it deforms into a shape $\boldsymbol{x}_t$. Now we have a system of equations $\boldsymbol{f}(\boldsymbol{x}_t, \bar{\boldsymbol{X}}) + \boldsymbol{g}_t = 0, t = 1...T$, which is over-constrained and thus may have no solution. In other words, the rest shape $\bar{\boldsymbol{X}}$ satisfying all equations does generally not exist.

We therefore seek to find a rest shape $\bar{\boldsymbol{X}}$ that deforms to every target shape $\boldsymbol{x}_t$ as close as possible. One straightforward formulation of this task is an optimization problem: find a shape $\bar{\boldsymbol{X}}$ that minimizes the sum of some residual measurement,

$$\bar{\boldsymbol{X}} = \arg\min_{\boldsymbol{X}} \sum_{t=1}^{T} \|\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{X}) + \boldsymbol{g}_t\|. \tag{21}$$

However, this is a nonlinear optimization problem suffering from many challenges of performance and convergence (see Table 2), and it can also be problematic since small residual forces can mean large differences in geometry.

To enable a responsive design tool, we desire a fast estimation of $\bar{\boldsymbol{X}}$. We notice that in the expansion Eq. (6), the coefficients $\boldsymbol{X}_i, i = 0...n$ are individual vectors, forming a local reduced subspace to represent $\bar{\boldsymbol{X}}$. While in Eq. (6) the magnitude of each spanning vector $\boldsymbol{X}_i$ is computed as a monomial $(a - a_0)^i$, here we relax it further to allow any value for the estimation of $\bar{\boldsymbol{X}}$.

Concretely, for every target shape $\boldsymbol{x}_t$ and related external force $\boldsymbol{g}_t$, we solve a rest shape $\bar{\boldsymbol{X}}_t$ individually. At every solution $\bar{\boldsymbol{X}}_t$, we also compute the expansion coefficients denoted as $\boldsymbol{X}_{t,i}$. We assemble all these coefficients into a basis matrix

$$\boldsymbol{U} = [\boldsymbol{X}_{1,0} \ \ldots \ \boldsymbol{X}_{1,n} \ \boldsymbol{X}_{2,0} \ \ldots \ \boldsymbol{X}_{2,n} \ \ldots \ \boldsymbol{X}_{T,0} \ \ldots \ \boldsymbol{X}_{T,n}],$$

and then find a shape $\bar{\boldsymbol{X}}$ represented by this basis matrix to approximate the individual solution $\bar{\boldsymbol{X}}_t$ in a least-squares sense:

$$\boldsymbol{q} = \arg\min_{\boldsymbol{q}} \sum_{t=1}^{T} \|\boldsymbol{U}\boldsymbol{q} - \bar{\boldsymbol{X}}_t\|_2^2,$$

and compute $\bar{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{q}$. This computation involves only a small scale least-squares solve with the reduced basis $\boldsymbol{U}$, which is much faster than solving the nonlinear problem Eq. (21) (see the video).

The rest shape $\bar{\boldsymbol{X}}$ computed in this way will generally not result in the deformed shape agreeing with $\boldsymbol{x}_t$ for every target shape. We therefore solve the static equilibrium shapes $\boldsymbol{x}_t^*$ from the rest shape $\bar{\boldsymbol{X}}$ to present the user their differences from the desired shape $\boldsymbol{x}_t$. We present the details of our fast static equilibrium solver in the next subsection.

5.3 ANM for Static Equilibrium Problem

While we primarily target inverse elastic design problems, ANM can also be used to solve the forward problem which computes the deformed shapes of elastic objects in static equilibrium. In particular, we apply ANM to solve $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{X}) + \boldsymbol{g} = 0$, in which the rest shape $\boldsymbol{X}$ and external force $\boldsymbol{g}$ are known, and the deformed shape $\boldsymbol{x}$ is unknown. In our inverse design system the benefit of a fast static equilibrium solver is twofold: it accelerates our calibration process to identify simulation material parameters that match the fabrication material, and provides quick feedback to the user about the difference between the resulting deformed shape $\boldsymbol{x}_t^*$ and the desired shape $\boldsymbol{x}_t$ in a multi-target design pipeline. More broadly, as the static equilibrium problem appears in many computational design (e.g., [Umetani et al. 2011]) and parameter selection (e.g. [Miguel et al. 2012]) problems, a fast solver can potentially improve the performance of those methods.

The ANM approach to solving the static equilibrium problem follows exactly the same steps presented in §4. Here we only highlight the important formulas different from the ones in §4, and refer the reader to the supplemental material for detailed formulas. First, since we are solving for $\boldsymbol{x}$ now, the expansion corresponding to Eq. (6) becomes

$$\begin{bmatrix} \boldsymbol{x}(a) \\ \lambda(a) \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_0 \\ \lambda_0 \end{bmatrix} + \sum_{k=1}^{n} (a - a_0)^k \begin{bmatrix} \boldsymbol{x}_k \\ \lambda_k \end{bmatrix}. \tag{22}$$

Next, we need to compute the internal force $\boldsymbol{f}$ using the first Piola-Kirchhoff tensor, $\boldsymbol{P} = \boldsymbol{F}\boldsymbol{S}$, that expresses the internal force $\boldsymbol{f}_i$ at a node $\boldsymbol{x}_i$ based on the undeformed nodal normal in material space,

$$\boldsymbol{f}_i = \sum_{t \in \mathsf{adj}(\boldsymbol{x}_i)} \boldsymbol{P}^t \bar{\boldsymbol{n}}_i^t,$$

where again $t \in \mathsf{adj}(\boldsymbol{x}_i)$ indicates a tetrahedron incident to the node $\boldsymbol{x}_i$; $\boldsymbol{P}^t$ is the piece-wise constant first Piola-Kirchhoff stress tensor at t; and $\bar{\boldsymbol{n}}_i^t$ is the outward normal of the node $\boldsymbol{x}_i$ of an undeformed tetrahedra t. Lastly, similar to Eq. (16), we construct an expansion of $\boldsymbol{P}$ and use it to solve for the expansion coefficients $\{\boldsymbol{x}_k, \lambda_k\}, k = 1...n$ at every ANM step.

The ANM provides fast performance for the static equilibrium solve. As shown in Table 3, compared to the static equilibrium solver with kinetic damping used in Umetani et al. [2011], the ANM offers 7× speedup on average for all the examples. We note that in [Umetani et al. 2011] the authors explicitly coarsen a simulated cloth (2D) mesh to achieve interactivity in their solver. For 3D deformable objects designed for physical fabrication, however, it is necessary to use a high-resolution tetrahedral mesh for accurate prediction. Therefore, a direct application of the method in [Umetani et al. 2011] is not practical.

6 Experimental Results

We have implemented the described method in an inverse shape design system based on the Vega FEM Library [Barbič et al. 2012a], and evaluated it with 7 models, whose statistics are summarized in Table 1. From the output of our algorithm, we first creates molds using 3D printed rigid models, and then cast elastomer material of type PU8400 (see [SunPe] for the detailed material parameters) to form final elastic objects.

Inverse Shape Design System. As depicted in the supplemental video, a typical shape design session with our system begins with loading a desired target shape $\boldsymbol{x}$. While the user can manually specify the fabrication material parameters (e.g., the shear and bulk modulus and density), we usually calibrate the parameters against specific fabrication materials *a priori*. The user only needs to specify what kind of materials will be used in the final fabrication. Next, the

Model	#Vertices	#Elements	#Targets	Gravity (N)	External (N)
bar	4552	19552	1	0.045	n/a
plant	14842	47077	1	2.283	n/a
holder	18753	72355	1	6.276	8.467
hanger	24323	98131	2	0.988	0.8
eagle	19307	71235	1	0.501	n/a
dinosaur	10673	32953	4	0.474	0.25/0.3/0.08
bifur3	5093	24478	1	122	1111

Table 1: *Statistics of experimental models. From left to right: Number of vertices, number of elements, number of design targets, and the magnitude of gravity and external forces. The external forces listed here are the total amount of forces applied on the objects, and their individual directions, positions and magnitudes are labeled in Figure 7-9.*

Model	LevMar solver		ANM solver		Speedup
	Time	#Iter.	Time	#Iter.	
bar	25m18s	5402	2.38s	2	638×
	8m23s	1750			211×
plant	1h2m34s	7594	7.07s	3	531×
	28m13s	3455			239×
holder	4m25s	95	12.82s	3	21×
	9m20s	216			43×
hanger	16m22s	629	26.83s	1/3	37×
	17m21s	675			38×
eagle	1h50m2s	4691	4.49s	1	1470×
	40m24s	780			539×
dinosaur	22m51s	993	28.17s	1/2/4/2	49×
	23m39s	1036			50×
bifur3	18m15s	1996	5.22s	3	210×
	14m53s	1613			171×

Table 2: *Statistics for inverse problem solves (with 6-threads OpenMP). Note that we report the timings of the LevMar solver with two kinds of initial guesses. For eagle and dinosaur, LevMar solver does not converge to the desired solution.*

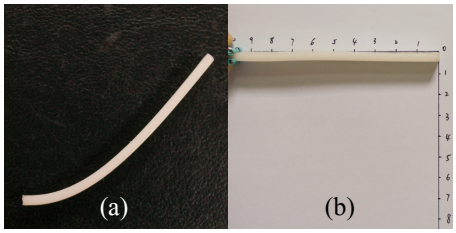

Figure 5: Simple Test Case: *(a) Our method computes a bending rest shape of an elastic bar holding in plane. (b) Under gravity, the bar restores into a straight bar.*

user specifies the external forces: gravity force is determined by the shape and density, but the user needs to indicate its direction (i.e., the downward direction); one also needs to fix certain parts of the shape to impose constraint force; and optionally the user can also specify the position, direction and magnitude of any additional forces to reflect the intended use of the physical object. For example, the user can constrain the base of a fish-shape cellphone holder, and specify two opposite forces that open its mouth to clamp the cellphone (see Figure 7).

Our system computes the resulting shape $\boldsymbol{X}$, typically within 10 seconds, and presents it to the user. After finding the rest shape $\boldsymbol{X}$, it allows the user to adjust the external forces (in addition to gravity) to explore other similar output shapes interactively.

Inverse Static Equilibrium Solver. We measured computation times of our ANM on a desktop PC with Intel i7-3770K CPU. Table 2 lists the timings and the number of ANM steps used in our experiments. We use a truncation order $n = 20$ for all examples. For simple shapes such as the elastic bar (Figure 5), the ANM took only 2.38 seconds to find the solution. It spent more time on the phone holder (Figure 7), because it undergoes a large deformation with the given force ($\boldsymbol{g} = 40$N), and thus the ANM costs more to reach from $\boldsymbol{g} = 0$ to $\boldsymbol{g} = 40$. For multi-target shape design, including the hanger (Figure 8) and the dinosaur (Figure 9), we compute the rest shapes using individual targets and estimate the final rest shape as introduced in §5.2. The timings in Table 2 are measured as the total amount of time spent on the entire solves.

In Table 2, we also compare the performance of our method to a standard Netwon-type method for solving the inverse static equilibrium problem. Our implementation of the Newton-type solver is

Model	Forward simulation		ANM solver		Speedup
	Time	#Iter.	Time	#Iter.	
bar	21.07s	216	3.25s	3	6×
plant	1m18s	454	9.27s	4	8×
holder	45.34s	110	17.99s	4	3×
hanger	51.71s	78	15.93s	3	3×
	72.45s	102	16.13s	3	4×
eagle	59.11s	214	8.11s	2	7×
dinosaur	1m3s	300	5.84s	4	11×
	1m5s	304	5.81s	4	11×
	1m15s	356	6.99s	5	11×
	1m9s	330	5.83s	4	12×
bifur3	51.57s	302	6.31s	4	8×

Table 3: *Statistics for forward problem solves (with 6-threads OpenMP). For forward simulation, we use Implicit-Newmark with kinetic damping [Umetani et al. 2011].*

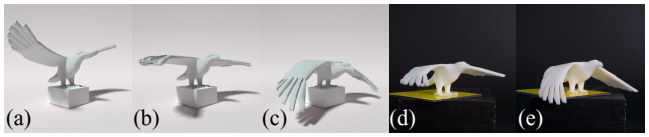

Figure 6: *We compute a rest shape of the eagle model (a) given its target shape under gravity (b), and fabricate the rest shape (d), which fails to be horizontal as it should be. The fabrication of the design target directly is shown in (e), while its simulation is (c).*

based on the Levenberg-Marquardt (LM) method [Levenberg 1944], because the multi-target inverse problem may not have an accurate solution, and can only be solved in a nonlinear least-squares sense. In practice, we use the widely used LM library [Lourakis Jul. 2004]. As shown in Table 2, our method is 2-3 orders of magnitude faster than the LM solver for all our examples. Since the performance of the Newton-type methods varies with different initial guesses, we explored different initial guess strategies and report the timings of the LM solver with two kinds of initial guesses. The first strategy uses the target shapes as the initial guess, while the second strategy computes an initial guess using a static equilibrium solver: let $\boldsymbol{x}$ denote the target shape; we first use it as a rest shape, inverse the external force $\boldsymbol{g}$ into $-\boldsymbol{g}$, and solve the static equilibrium state $\boldsymbol{x}'$ satisfying $f(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{g} = 0$. We use $\boldsymbol{x}'$ as the initial guess and benchmark the performance. This strategy improves the LM solves most of the time, but it requires extra time to compute $\boldsymbol{x}'$. More importantly, even without taking into account the overhead of computing $\boldsymbol{x}'$, our method still runs 1-2 orders of magnitude faster.

Forward Static Equilibrium Solver. Table 3 summarizes the timings of static equilibrium solver and compares against a kinetic damping solver as used in [Umetani et al. 2011]. Even though kinetic damping was reported to be faster than the conventional Newton-type method, our ANM solver still performs about 3-10 times faster over the kinetic damping method for all our examples.

Physical Validation. We first evaluate our algorithm with a simple bar model. We require that the deformed shape of the bar be horizontal under gravity. Figure 5(a) shows a photo of the rest shape of a fabricated bar based on our computation, while Figure 5(b) shows that the fabricated rest shape deforms into the expected horizontal shape with little approximation error. The robustness of our solver with respect to external forces is demonstrated in Figure 7. In the design stage, external forces with a magnitude of 40N, are exerted at two sides of the fish mouth to open it as a holder (Figure 7(b)). The real deformation under the external force is demonstrated in Figure 7(d), while the rest shape is shown in Figure 7(c).

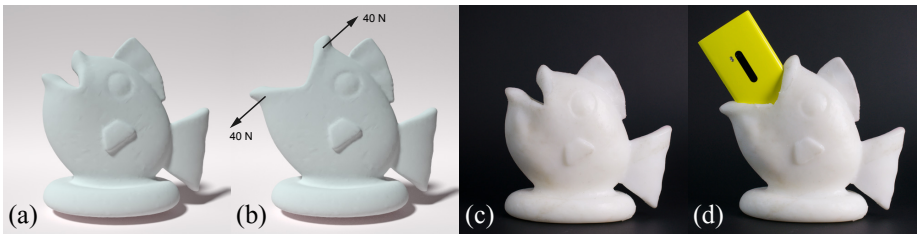

Figure 7: Phone Holder: *We compute a rest shape of a phone holder (a) based on its target shape under working forces (b) for clamping a cell phone. We then fabricate the computed rest shape (c). As shown in (d), its mouth clamps a cell phone tightly as predicted.*

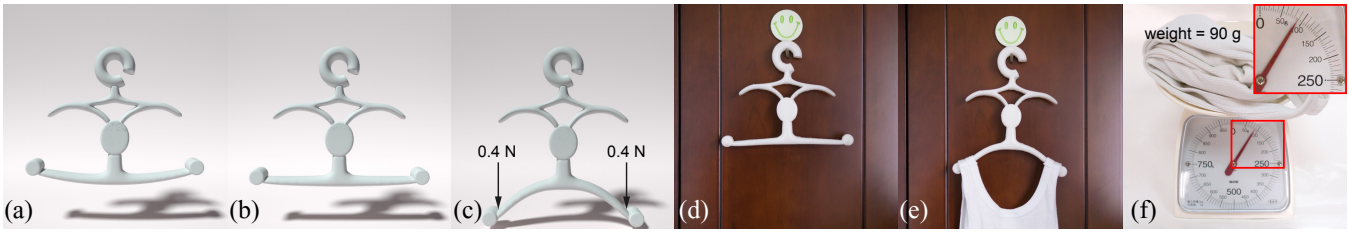

Figure 8: Hanger: *We compute a rest shape of hanger model (a) given its target shape under gravity (b) and target shape under working forces (c). The fabrication of the rest shape has a horizontal bottom bar under gravity (d). The shape under the target work load (e) is visually similar to the designed target shape. The weight of cloth is shown in (f).*

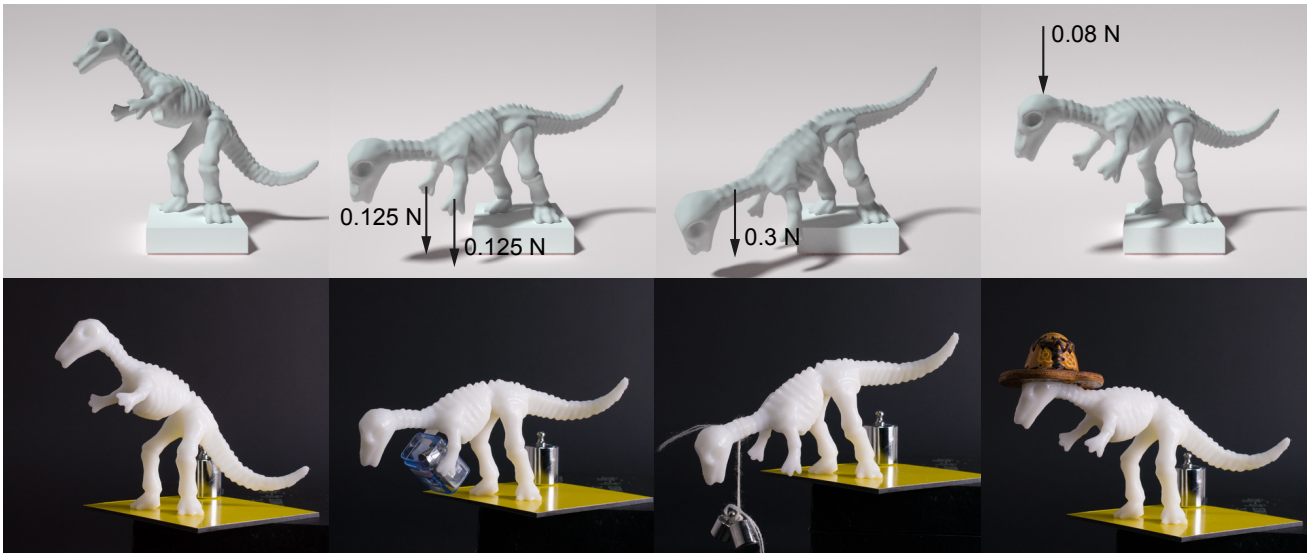

Figure 9: Multi-target Dinosaur: *We compute a rest shape for the dinosaur model given multiple designed target shapes (top row). When applying the forces to the fabricated shape, we observe deformations that are visually similar to the corresponding target shapes (bottom row).*

We verify the multi-target optimization algorithm as shown in Figure 8. The two target shapes under gravity and an additional external force (0.4N at both sides of the clothes stand) are shown in Figure 8(b) and 8(c), respectively. The rest shape obtained from optimization is shown in Figure 8(a). To validate the accuracy of our simulation, we hang a piece of cloth with a weight of 90 grams on the fabricated object. The resulting deformed shape is visually very similar to the designed shape.

We demonstrate the second example of multi-target shape design in Figure 9, in which we specify four targeted poses deformed by forces exerted on the dinosaur's hands, neck, and head respectively in addition to gravity.

Failure Cases and Limitations. Although our method works well under various scenarios, among all our examples we observed one failure case (shown in Figure 6(d)), for which the deformation of the fabricated shape is still visually different from the desired shape. We believe the major reason is the insufficient accuracy of the fabrication technology using elastomer material. In addition, the use of finite element approximation is less predictive for thin structures such as the wings of the eagle, for which a large number of tetrahedra are needed to well resolve the thin features. Moreover, the inaccuracy of the neo-Hookean model is another possible reason. We also note that for the same test case, Levenberg-Marquardt solver even fails to find a solution.

Given an arbitrary user input, it is possible that no feasible inverse shape exists. Currently our method stops when a solution is found or the maximum number of iterations is reached. For the latter case, we check whether the resulting shape satisfies the static equilibrium equation and report failure cases to the user.

To successfully apply ANM in other constitutive models, the crucial part is to derive the appropriate transformation in order to obtain a quadratic formulation of the internal force function. The trans-

formations presented in this work were derived manually, and it is unclear to what extent these derivations can be automated. Neverthless, symbolic differentiation software may be used to automate the computation of derivatives.

7 Conclusion

We have introduced an asymptotic numerical method for solving inverse static equilibrium problems for nonlinear neo-Hookean elasticity. We have applied this method to compute rest shapes for elastic fabrication. Our method runs orders of magnitude faster than traditional Newton-type methods for all the examples considered in this work. We demonstrate the performance and robustness of our method by designing and fabricating various elastic objects.

While ANM offers promising performance to solve nonlinear equations in our inverse design problem, future work should investigate its further applications in elastic fabrication. For example, while our method is applicable to anisotropic materials, it needs an extension of the derived formulas to incorporate anisotropy. In addition, applying ANM in design problems with shape constraints is an interesting direction, and might lead to new computational elastic design tools. Moreover, incorporating the support of partially deformed targets could benefit users in specific design situations. We are also interested in the simulation and fabrication of different kinds of elastic materials. For instance, it is of much interest to investigate whether we can fabricate elastic human skin, which can go beyond graphics and find applications in robotics and other areas.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive comments; Timothy Sun for dubbing the video; and Meng Zhang for the rendering efforts. This work is partially supported by NSFC (No. 61272305, No. 61303136 and No. 61322204), the National Program for Special Support of Eminent Professionals of China, Columbia Junior Faculty startup fund, Lenovo's Program for Young Scientists as well as generous gifts from Intel.

A Derivation Details of the Inverse Problem

For the inverse problem of neo-Hookean elasticity, we introduce a set of auxiliary variables to establish quadratic relationships between $\boldsymbol{\sigma}$ and $\boldsymbol{X}$ as described in §4.2.2:

$$\begin{cases} \boldsymbol{\sigma} = \mu J^{-\frac{5}{3}}\boldsymbol{b} - \dfrac{\mu}{3}J^{-\frac{5}{3}}I_c\boldsymbol{I} + \kappa(J-1)\boldsymbol{I} \\ s^{(5)} = s^{(2)}J^{inv} \\ s^{(2)} = s^{(1)}s^{(1)} \\ J^{inv} = s^{(2)}s^{(1)} \\ JJ^{inv} = 1 \\ J = \epsilon_{lmn}\tilde{\boldsymbol{F}}_{lm}\boldsymbol{F}_{n3} \\ \tilde{\boldsymbol{F}}_{lm} = \boldsymbol{F}_{l1}\boldsymbol{F}_{m2} \\ I_c = \boldsymbol{I} : \boldsymbol{b} \\ \boldsymbol{b} = \boldsymbol{F}\boldsymbol{F}^T \\ \boldsymbol{F} = [\boldsymbol{x}][\boldsymbol{X}]^{inv} \\ [\boldsymbol{X}][\boldsymbol{X}]^{inv} = \boldsymbol{I} \end{cases}$$

where $s^{(5)} = J^{-\frac{5}{3}}$, $s^{(2)} = J^{-\frac{2}{3}}$, $s^{(1)} = J^{-\frac{1}{3}}$, and $\boldsymbol{F}_{ij}$ indicates the element (i,j) of the deformation gradient $\boldsymbol{F}$. $[\boldsymbol{x}]$ and $[\boldsymbol{X}]$ denote two matrices for each tetrahedron in the deformed and material space respectively. Namely,

$$[\boldsymbol{x}] = [\boldsymbol{x}_1 - \boldsymbol{x}_0 \;\; \boldsymbol{x}_2 - \boldsymbol{x}_0 \;\; \boldsymbol{x}_3 - \boldsymbol{x}_0],$$
$$[\boldsymbol{X}] = [\boldsymbol{X}_1 - \boldsymbol{X}_0 \;\; \boldsymbol{X}_2 - \boldsymbol{X}_0 \;\; \boldsymbol{X}_3 - \boldsymbol{X}_0].$$

Here $\boldsymbol{x}_i$ and $\boldsymbol{X}_i, i = 0...3$ are vertex positions of a tetrahedron in the deformed and material space.

Based on these relationships, the recurrence formula for computing k-th order polynomial coefficients has the following expansions:

$$\begin{cases} \boldsymbol{\sigma}_k = \mu\left(s_k^{(5)}\boldsymbol{b}_0 + s_0^{(5)}\boldsymbol{b}_k + \sum_{r=1}^{k-1} s_r^{(5)}\boldsymbol{b}_{k-r}\right) \\ \qquad -\dfrac{\mu}{3}\boldsymbol{I}\left(s_k^{(5)}(I_c)_0 + s_0^{(5)}(I_c)_k + \sum_{r=1}^{k-1} s_r^{(5)}(I_c)_{k-r}\right) + \kappa J_k\boldsymbol{I} \\ s_k^{(5)} = s_0^{(2)}J_k^{inv} + s_k^{(2)}J_0^{inv} + \sum_{r=1}^{k-1} s_r^{(2)}J_{k-r}^{inv} \\ s_k^{(2)} = s_0^{(1)}s_k^{(1)} + s_k^{(1)}s_0^{(1)} + \sum_{r=1}^{k-1} s_r^{(1)}s_{k-r}^{(1)} = 2s_0^{(1)}s_k^{(1)} + \sum_{r=1}^{k-1} s_r^{(1)}s_{k-r}^{(1)} \\ J_k^{inv} = s_0^{(2)}s_k^{(1)} + s_k^{(2)}s_0^{(1)} + \sum_{r=1}^{k-1} s_r^{(2)}s_{k-r}^{(1)} \\ J_k^{inv} = -J_0^{inv}J_kJ_0^{inv} - J_0^{inv}\sum_{r=1}^{k-1} J_rJ_{k-r}^{inv} \\ J_k = \epsilon_{lmn}\left[(\tilde{\boldsymbol{F}}_{lm})_k(\boldsymbol{F}_{n3})_0 + (\tilde{\boldsymbol{F}}_{lm})_0(\boldsymbol{F}_{n3})_k + \sum_{r=1}^{k-1}(\tilde{\boldsymbol{F}}_{lm})_r(\boldsymbol{F}_{n3})_{k-r}\right] \\ (\tilde{\boldsymbol{F}}_{lm})_k = (\boldsymbol{F}_{l1})_k(\boldsymbol{F}_{m2})_0 + (\boldsymbol{F}_{l1})_0(\boldsymbol{F}_{m2})_k + \sum_{r=1}^{k-1}(\boldsymbol{F}_{l1})_r(\boldsymbol{F}_{m2})_{k-r} \\ (I_c)_k = \boldsymbol{I} : \boldsymbol{b}_k \\ \boldsymbol{b}_k = \boldsymbol{F}_k\boldsymbol{F}_0^T + \boldsymbol{F}_0\boldsymbol{F}_k^T + \sum_{r=1}^{k-1}\boldsymbol{F}_r\boldsymbol{F}_{k-r}^T \\ \boldsymbol{F}_k = [\boldsymbol{x}][\boldsymbol{X}]_k^{inv} \\ [\boldsymbol{X}]_k^{inv} = -[\boldsymbol{X}]_0^{inv}[\boldsymbol{X}]_k[\boldsymbol{X}]_0^{inv} - [\boldsymbol{X}]_0^{inv}\left(\sum_{r=1}^{k-1}[\boldsymbol{X}]_r[\boldsymbol{X}]_{k-r}^{inv}\right) \end{cases}$$

Here $[\boldsymbol{X}]_k$ is the expansion coefficient of $[\boldsymbol{X}]$. Because all elements of $[\boldsymbol{X}]$ are linear with respect to the vector $\boldsymbol{X}$, $[\boldsymbol{X}]_k$ is also linear with respect to $\boldsymbol{X}_k$ in Eq. (6). Using these expansions, one can see that $\boldsymbol{\sigma}_k$ is linear with respect to $\boldsymbol{X}_k$. Therefore, the computation of matrix A in (18) merely depends on the zeroth-order terms of the auxiliary variables, and the vector $\boldsymbol{f}_{(k)}^{nl}$ can be assembled from all r-th order terms where $1 \le r < k$. After computing A and $\boldsymbol{f}_{(k)}^{nl}$, we solve for $(\boldsymbol{X}_k, \lambda_k)$ using Eq. (12) and Eq. (18) as follows:

For the first order, we have

$$\boldsymbol{X}_1^* = \mathsf{A}^{-1}\boldsymbol{g}, \;\; \lambda_1 = \left(\|\boldsymbol{X}_1^*\|^2 + 1\right)^{-\frac{1}{2}}, \;\; \boldsymbol{X}_1 = \left(\|\boldsymbol{X}_1^*\|^2 + 1\right)^{-\frac{1}{2}}\boldsymbol{X}_1^*.$$

For order $k > 1$, we solve the linear equation for $(\boldsymbol{X}_{(k)}^{nl}, \lambda_k)$ using

$$\begin{cases} \mathsf{A}\boldsymbol{X}_{(k)}^{nl} = \boldsymbol{f}_{(k)}^{nl} \\ \lambda_k = -\lambda_1\boldsymbol{X}_1^T\boldsymbol{X}_{(k)}^{nl} \end{cases},$$

followed by $\boldsymbol{X}_k = \dfrac{\lambda_k}{\lambda_1}\boldsymbol{X}_1 + \boldsymbol{X}_{(k)}^{nl}$.

Finally, we use a simple example to illustrate why A is equivalent to $\left.\dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{X}}\right|_{\boldsymbol{X}_0}$. Consider a quadratic form $f(x) = a(x)b(x)$, suppose

the precondition $\left.\frac{\partial a}{\partial x}\right|_{x_0} = \alpha$, $\left.\frac{\partial b}{\partial x}\right|_{x_0} = \beta$ and $a_k = \alpha x_k + a^{nl}_{(k)}$, $b_k = \beta x_k + b^{nl}_{(k)}$ is true, we can simply get $f_k = a_0 b_k + a_k b_0 + \sum_{r=1}^{k-1} a_r b_{k-r} = (a_0\beta + b_0\alpha)x_k + f^{nl}_{(k)}$, in which $a_0\beta + b_0\alpha = a(x_0)\left.\frac{\partial b}{\partial x}\right|_{x_0} + b(x_0)\left.\frac{\partial a}{\partial x}\right|_{x_0} = \left.\frac{\partial f}{\partial x}\right|_{x_0}$. The precondition is always true for linear functions of x.

References

ALLGOWER, E. L., AND GEORG, K. 1990. *Numerical continuation methods*, vol. 13. Springer-Verlag Berlin.

BÄCHER, M., BICKEL, B., JAMES, D. L., AND PFISTER, H. 2012. Fabricating articulated characters from skinned meshes. *ACM Trans. Graph. 31*, 4, 47.

BARBIČ, J., DA SILVA, M., AND POPOVIĆ, J. 2009. Deformable object animation using reduced optimal control. *ACM Trans. Graph. 28*, 3, 53.

BARBIČ, J., SIN, F. S., AND SCHROEDER, D., 2012. Vega FEM Library. http://www.jernejbarbic.com/vega.

BARBIČ, J., SIN, F., AND GRINSPUN, E. 2012. Interactive Editing of Deformable Simulations. *ACM Trans. Graph. 31*, 4, 70.

BICKEL, B., BÄCHER, M., OTADUY, M. A., LEE, H. R., PFISTER, H., GROSS, M., AND MATUSIK, W. 2010. Design and fabrication of materials with desired deformation behavior. *ACM Trans. Graph. 29*, 4, 63.

BICKEL, B., KAUFMANN, P., SKOURAS, M., THOMASZEWSKI, B., BRADLEY, D., BEELER, T., JACKSON, P., MARSCHNER, S., MATUSIK, W., AND GROSS, M. 2012. Physical face cloning. *ACM Trans. Graph. 31*, 4, 118.

BONET, J., AND WOOD, R. D. 1997. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University.

BRENT, R. P. 2013. *Algorithms for minimization without derivatives*. Courier Dover Publications.

CALÌ, J., CALIAN, D. A., AMATI, C., KLEINBERGER, R., STEED, A., KAUTZ, J., AND WEYRICH, T. 2012. 3d-printing of non-assembly, articulated models. *ACM Trans. Graph. 31*, 6, 130.

CEYLAN, D., LI, W., MITRA, N. J., AGRAWALA, M., AND PAULY, M. 2013. Designing and fabricating mechanical automata from mocap sequences. *ACM Trans. Graph. 32*, 6, 186.

CHEN, D., LEVIN, D. I. W., DIDYK, P., SITTHI-AMORN, P., AND MATUSIK, W. 2013. Spec2fab: A reducer-tuner model for translating specifications to 3d prints. *ACM Trans. Graph. 32*, 4.

COCHELIN, B. 1994. A path-following technique via an asymptotic-numerical method. *Computers & structures 53*, 5, 1181–1192.

COROS, S., MARTIN, S., THOMASZEWSKI, B., SCHUMACHER, C., SUMNER, R., AND GROSS, M. 2012. Deformable objects alive! *ACM Trans. Graph. 31*, 4, 69.

COROS, S., THOMASZEWSKI, B., NORIS, G., SUEDA, S., FORBERG, M., SUMNER, R. W., MATUSIK, W., AND BICKEL, B. 2013. Computational design of mechanical characters. *ACM Trans. Graph. 32*, 4, 83.

DAMIL, N., AND POTIER-FERRY, M. 1990. A new method to compute perturbed bifurcations: Application to the buckling of imperfect elastic structures. *International Journal of Engineering Science 28*, 9, 943–957.

DEROUET-JOURDAN, A., BERTAILS-DESCOUBES, F., AND THOLLOT, J. 2010. Stable inverse dynamic curves. *ACM Trans. Graph. 29*, 6, 137.

DEROUET-JOURDAN, A., BERTAILS-DESCOUBES, F., DAVIET, G., AND THOLLOT, J. 2013. Inverse dynamic hair modeling with frictional contact. *ACM Trans. Graph. 32*, 6, 159.

HADAP, S. 2006. Oriented strands: dynamics of stiff multi-body system. In *Proceedings of SCA*, 91–100.

HILDEBRANDT, K., SCHULZ, C., VON TYCOWICZ, C., AND POLTHIER, K. 2012. Interactive spacetime control of deformable objects. *ACM Trans. Graph. 31*, 4, 71.

LAZARUS, A., MILLER, J., AND REIS, P. 2013. Continuation of equilibria and stability of slender elastic rods using an asymptotic numerical method. *J. Mech. Phys. Solids 61*, 8, 1712–1736.

LEVENBERG, K. 1944. A method for the solution of certain non-linear problems in least squares. *Quart. J. Appl. Maths. II*, 2.

LOURAKIS, M. I. A., Jul. 2004. levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++. [web page] http://www.ics.forth.gr/~lourakis/levmar.

MARTIN, S., THOMASZEWSKI, B., GRINSPUN, E., AND GROSS, M. 2011. Example-based elastic materials. *ACM Trans. Graph. 30*, 4, 72.

MIGUEL, E., BRADLEY, D., THOMASZEWSKI, B., BICKEL, B., MATUSIK, W., OTADUY, M. A., AND MARSCHNER, S. 2012. Data-driven estimation of cloth simulation models. *Computer Graphics Forum 31*, 2, 519–528.

NEALEN, A., MÜLLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2006. Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4, 809–836.

OGDEN, R. W. 1997. *Non-linear elastic deformations*. Courier Dover Publications.

PRÉVOST, R., WHITING, E., LEFEBVRE, S., AND SORKINE-HORNUNG, O. 2013. Make it stand: balancing shapes for 3d fabrication. *ACM Trans. Graph. 32*, 4, 81.

SKOURAS, M., THOMASZEWSKI, B., BICKEL, B., AND GROSS, M. 2012. Computational design of rubber balloons. *Computer Graphics Forum 31*, 2, 835–844.

SKOURAS, M., THOMASZEWSKI, B., COROS, S., BICKEL, B., AND GROSS, M. 2013. Computational design of actuated deformable characters. *ACM Trans. Graph. 32*, 4, 82.

STAVA, O., VANEK, J., BENES, B., CARR, N., AND MĚCH, R. 2012. Stress relief: Improving structural strength of 3d printable objects. *ACM Trans. Graph. 31*, 4, 48.

SUNPE, P. Professional Rapid Prototyping & Manufacturing. http://www.sunpe.com/types-34.html, SunPe PROTOTYPE.

TWIGG, C. D., AND KAČIĆ-ALESIĆ, Z. 2011. Optimization for sag-free simulations. In *Proceedings of SCA*, 225–236.

UMETANI, N., KAUFMAN, D. M., IGARASHI, T., AND GRINSPUN, E. 2011. Sensitive couture for interactive garment modeling and editing. *ACM Trans. Graph. 30*, 4, 90.

ZAHROUNI, H., COCHELIN, B., AND POTIER-FERRY, M. 1999. Computing finite rotations of shells by an asymptotic-numerical method. *Comput. Methods Appl. Mech. Eng. 175*, 1, 71–85.