Example-Based Subspace Stress Analysis for Interactive Shape Design

Xiang Chen, Changxi Zheng, and Kun Zhou, *Fellow, IEEE*

Abstract—Stress analysis is a crucial tool for designing structurally sound shapes. However, the expensive computational cost has hampered its use in interactive shape editing tasks. We augment the existing example-based shape editing tools, and propose a fast subspace stress analysis method to enable *stress-aware* shape editing. In particular, we construct a reduced stress basis from a small set of shape exemplars and possible external forces. This stress basis is automatically adapted to the current user edited shape on the fly, and thereby offers reliable stress estimation. We then introduce a new finite element discretization scheme to use the reduced basis for fast stress analysis. Our method runs up to two orders of magnitude faster than the full-space finite element analysis, with average L_2 estimation errors less than 2% and maximum L_2 errors less than 6%. Furthermore, we build an interactive stress-aware shape editing tool to demonstrate its performance in practice.

Index Terms—Shape editing, elastostatic stress analysis, reduced stress basis, finite element method, shape deformation.

1 Introduction

An old yet important tool in engineering design, structural stress analysis predicts stress responses of structures subjected to external forces. The use of stress analysis has also been popularized recently in computer graphics, owing to the rapid advances in 3D fabrication techniques. To ease novice users to create and fabricate structurally sound shapes, many tools have integrated structural analysis in the shape design process (e.g., see [1], [2], [3], [4], [5]).

However, structural stress analysis remains computationally expensive. When designing a shape for physical fabrication, one needs to prioritize the computational accuracy over its performance. The classic finite element method (FEM), when used for predictive stress analysis, often requires a high-resolution mesh or high-order elements, resulting in a large linear system generally expensive to solve. Given a typical 3D model for fabrication, hundreds of thousands of mesh vertices are commonly needed for a reasonably accurate analysis, which can take up to minutes. As an example, the linear stress analysis of the classic Armadillo model (Figure 1) with two million tetrahedra takes eight minutes on our desktop—even with the state-of-the-art industrial FEM software COMSOL [6].

The high computational cost places a hurdle for using stress analysis in interactive shape editing. While there exist many well-developed tools for interactive shape deformation, whenever a shape is changed, previous stress analysis results become invalid, and the user needs a complete recomputation of stress analysis to assess the structural soundness of the newly updated shape.

In this paper, we propose a fast stress analysis method. Combined with existing shape deformation tools, our method enables *stress-aware* shape editing, providing the user immediate stress analysis feedback in a shape design loop.

Our approach is motivated by the recent success of example-based shape editing methods (e.g. [7], [8], [9], [10], [11], [12]). These methods build a subspace of shapes using a number of example deformations. The resulting subspace model of shape deformation allows the user to explore at runtime a range of shapes interactively. Because the linear elastostatic stress depends on the geometry of an object (as well as the material and external forces), we hypothesize that the stresses of the subspace shape variants in those shape editing tools also lie in a low-dimensional stress space. Therefore, we reuse the same example shapes provided for those shape editing tools and build a *reduced stress* model.

Our method integrates naturally with existing example-based shape editing methods, and enables fast stress analysis with sufficient accuracy. At the precomputation step, when the existing tools build a subspace of deformations, our method computes a stress basis in parallel. During a runtime editing session, after the user changes the shape, our method updates the resulting shape's stress field under external forces immediately. It also allows the user to adjust forces (such as gravitational magnitude and direction) and update the stress instantly. To demonstrate the application of our fast stress analysis, we augment the existing shape editing tools to provide new stress-aware functionalities, including interactive feedback of the shape's structural soundness as well as automatic shape adjustment for satisfying their stress requirement (§6.4).

This work has two major technical contributions. First, we introduce a method to construct a reduced stress basis from a small set of shape deformation examples and possible external forces. Our stress subspace is not represented by merely a fixed stress basis. Through pull-back and push-forward operations, our stress basis adapts to the current user edited shape on the fly, and thus offers

- *X. Chen and K. Zhou are with the State Key Lab of CAD&CG, Zhejiang University, Mengminwei Building, Zijingang Campus, Hangzhou, Zhejiang, China 310058. E-mail: xchen.cs@gmail.com, kunzhou@acm.org.*
- *C. Zheng is with the Department of Computer Science, Columbia University, 616 Schapiro (CEPSR), New York, NY 10027. Email: cxz@cs.columbia.edu.*

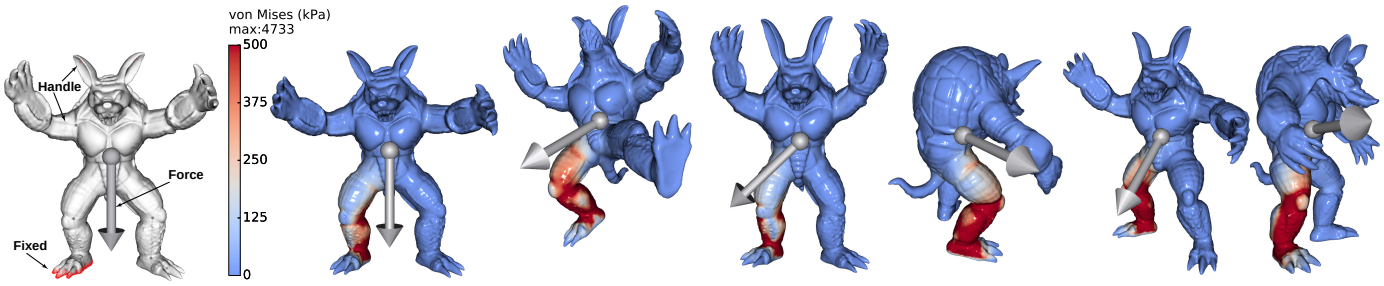


Fig. 1. **Armadillo under Stress.** Given an initial rest shape (591k tetrahedra) and its boundary condition setting (left), our subspace method enables interactive linear elastostatic stress analysis (0.4sec) when the user deforms the rest shape and modifies external forces (right). Our method in this example is **67**$\times$ faster than a full-space FEM analysis (28secs), while retaining comparable computation accuracy.

reliable stress estimation. Next, we introduce a new finite element discretization to use the reduced basis for fast stress analysis. We formulate a quadratic minimization problem directly on the stress field, in contrast to the formulation on displacement field as used in standard finite element practice. Minimizing the quadratic energy function amounts to solving a small linear system, thus facilitating interactive stress analysis at runtime.

Our method differs from the traditional engineering stress analysis in terms of its target applications. We do not aim for general design tasks. Instead, we focus our method in the context of shape editing applications, and aim for a fast and reliable stress analysis tool. Experiments show that our method runs up to two orders of magnitude faster than the standard finite element analysis, with average L_2 estimation errors less than 2% and maximum L_2 estimation errors less than 6%.

2 Related Work

Interactive shape deformation is a subject of substantial interest in computer graphics. There has been a large body of work on this topic that we can not exhaustively list here. Typical examples include free-form deformation [13], [14], multi-resolution deformation [15], [16], and differential domain methods [17], [18], [19] (also see [20] for a survey), just to name a few. All these methods focus on the geometric aspects of a shape, whereas our method is concerned with the shape's physical properties, the stress distribution, during an editing process. Our approach is independent of any particular shape editing tool and thus complementary to these methods.

One common strategy of interactive shape deformation is motivated by the observation that a user-edited shape is often in a subspace of all possible deformations. In many methods, building such a deformation subspace exploits user provided exemplars [7], [8], [9], [10]. For instance, Sumner et al. [8] extracted feature vectors of deformation gradients from example deformations and used them to interpolate new shapes at runtime.

In a larger context, shape exemplars have also been used to add dynamic effects [11], build a subspace of elastic strain [21], [22], synthesize animations [23], [24], and so forth. In physics-based simulation such as contact simulation, while there exist efficient full-space methods (e.g., [25]), example-based subspace construction has been studied to provide better performance, often at interactive rates. This line of research includes fast simulation of deformable bodies [26], [27], fluids [28], hair [29], animation control [30], [31], sound synthesis [32], interactive material design [33] and many others.

In this work, we exploit example-based model reduction for linear elastostatic stress analysis, particularly for fabrication-oriented shape editing tasks. This goal gives rise to unique challenges for subspace construction. The core question is what a proper basis of a stress tensor field across different shapes should be.

In parallel to the increasing popularity of additive manufacturing technology (i.e., "3D printing"), fabrication-aware design is of growing research interest. Under this umbrella, plenty of prior work has incorporated a physical simulation into the design of fabricated geometries. For example, Umetani et al. [34] proposed an interactive system for furniture design, in which the structural stability is simulated and corrected using sensitivity analysis. Skouras et al. [35] employed an optimization process to design actuation forces and locations as well as material distributions to match input shapes. Chen et al. [36] solved a nonlinear elastostatic equation to design the shape of a soft object to match a desired shape subject to external forces.

Stress analysis has been serving as an important tool for many fabrication-aware design tasks. To design cost-effective printing shapes, stress analysis can help to ensure the structural soundness while reducing the material consumption using skin-frame structures [3] or honeycomb-like porous carving [4]. Stava et al. [1] introduced a systematic approach to evaluate the structural weakness of a shape and proposed a set of operations to strengthen the weak parts. More widely, in engineering design, there has been a long history of optimizing the shapes minimizing stress concentration [37], [38], [39]. These methods typically use elastostatic stress analysis in their system, but the stress analysis itself is performed using the standard finite element method. In contrast, our goal in this work is to expedite the stress analysis process while retaining sufficient accuracy.

More recently, Zhou et al. [2] proposed a novel constrained optimization method to analyze linear elastostatic stresses. It computes a weakness map to estimate how the object behaves in the worst case among various possibilities of external forces. Umetani and Schmidt [40] introduced a technique to analyze the cross-sections of printed objects and optimize the printing orientation for the purpose of increasing mechanical strength. While the methods are ef-

fective for identifying fragile parts and unwanted forces loaded on a design shape, the designed shape itself is required to be unchanged during the analysis. In contrast, we focus on the fast stress analysis technique during a typical design session, in which the design shape and forces are both interactively varying. Additionally, our approach is an example-based reduced model, which we exploit to achieve fast computational performance.

Sharing a similar goal, a recent work by Xie et al. [5] presented a shape editing system while providing the user structural analysis feedback. Their method decomposes the shape into domains and reuses the computation of stiffness matrix for unedited domains. Thereby, it reduces the matrix assembling time, but their stress computation still needs to solve the full-space linear FEM system, which is often the most expensive step. Instead, we exploit a stress subspace and thereby gain orders of magnitude speedups.

There have been many works on reduced simulation of elasticity in computer graphics, e.g., using vibrational modes and modal derivatives for simulating deformable objects [41], [42]. Another strategy is to use numerical coarsening to preserve large-scale deformable behaviors while using a low-resolution mesh for finite element analysis [43], [44], [45]. These methods mainly focus on elastic animations. Instead, we focus on fast and accurate elastostatic stress analysis for fabrication-oriented shape editing.

In summary, compared with all these methods, our work provides a unique feature for many engineering and graphics applications that require frequent and accurate stress analysis in an interactive shape editing process.

3 Background

We start by introducing notations used throughout this paper, and then briefly review the standard linear elastostatic stress analysis.

3.1 Notation

We build our method on the theory of continuum mechanics, and thus follow the notations widely used therein. We use a bold letter (e.g., $\boldsymbol{x}$) to denote a vector, a Greek letter (e.g., $\boldsymbol{\sigma}$) to denote a 2nd-order tensor, and a sans-serif upper letter (e.g., C) to denote a matrix or higher-order tensor. We index elements in vectors and tensors using subscripts. For instance, C_{ijmn} is an element in a 4th-order tensor C. When dealing with vector-valued functions $\boldsymbol{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, f_j refers to the j-th component of $\boldsymbol{f}(\boldsymbol{x})$. Further, we write the derivatives of a vector variable function as

$$f_{i,j} = \frac{\partial f_i}{\partial x_j} \text{ and } f_{i,jk} = \frac{\partial^2 f_i}{\partial x_j \partial x_k}.$$

Here the comma on the left-hand side is used to separate component indices from differentiation indices.

We use Einstein summation convention to avoid the summation signs which occur frequently when manipulating indexed quantities in our derivation. When we write a product of indexed quantities, a subscript is a *free* index when it occurs only once in the product, and a *summation* index otherwise. For instance, a matrix-vector multiplication, $c_i = \sum_{j=1}^{n} \mathsf{A}_{ij} b_j$ can be written as $c_i = \mathsf{A}_{ij} b_j$, where i is a free index and j is the summation index.

3.2 Linear elastostatic analysis

Many materials used in 3D fabrication, including strong plastic, metal, ceramic and porcelain, are stiff, undergoing little deformation before reaching yield conditions. Therefore, we use the linear elasticity model for stress analysis, because of its computational efficiency and its successful use in many 3D fabrication design work [1], [2], [3], [4], [5].

In linear elastostatic analysis, the stress $\boldsymbol{\sigma}$, strain $\boldsymbol{\epsilon}$, and displacement $\boldsymbol{u}$ are linearly related to each other:

$$\sigma_{ij} = \mathsf{C}_{ijmn}\epsilon_{mn} \text{ and } \epsilon_{mn} = \frac{1}{2}(u_{m,n} + u_{n,m}), \quad (1)$$

where $\boldsymbol{\sigma}$ and $\boldsymbol{\epsilon}$ are both 2nd-order stress tensors (i.e., 3×3 matrices), and C is a 4th-order stiffness tensor whose coefficients depend on specific materials [46]. $u_{m,n}$ is the displacement gradient (i.e., $u_{m,n} = \frac{\partial u_m}{\partial x_n}$) using the indexed notation. Combining both expressions, we obtain the relationship between $\boldsymbol{\sigma}$ and $\boldsymbol{u}$:

$$\sigma_{ij} = \mathsf{E}_{ijmn} u_{m,n}, \text{ where } \mathsf{E}_{ijmn} = \frac{1}{2}(\mathsf{C}_{ijmn} + \mathsf{C}_{ijnm}). \quad (2)$$

The central equation in elastostatic stress analysis is

$$(\sigma_{ij})_{,j} + b_i = (\mathsf{E}_{ijmn} u_{m,n})_{,j} + b_i = 0, \quad (3)$$

where $\boldsymbol{b}$ is the external force at a position of the solid body (i.e., $\boldsymbol{b}$ is a spatially varying function), and $(\sigma_{ij})_{,j}$ is the divergence of the stress tensor, describing the internal strain force. Again, as an example of the index notation, $(\sigma_{ij})_{,j} = \sum_{j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j}$, where the subscript j is a summation index, and i is a free index. Eq. (3) states that the internal strain force produced by the material deformation should balance against the external force $\boldsymbol{b}$.

To indicate the occurrence of material failure or yielding (e.g., the plastic deformation), a stress-related scalar is used to check if a yielding threshold is reached. Depending on specific applications, commonly used quantities include the *von Mises* stress [47] and the maximum principal stress [48], both are scalar functions of the stress tensor $\boldsymbol{\sigma}$. In this work, we evaluate our method using both of these measures (§6).

The linear elasticity equation Eq. (3) is often solved using a finite element method (FEM) [49]. Typically, one first discretizes the displacement field $\boldsymbol{u}$ over the entire solid body represented by a tetrahedral mesh and obtains a linear system, in which the unknowns are $\boldsymbol{u}$ values at mesh vertices. Lastly, $\boldsymbol{u}$ is substituted into the relationships in Eq. (1) to evaluate stress tensors. When the mesh resolution is high, the linear system is of a large size, resulting in a high computational cost.

4 Reduced Stress Analysis

We now present our runtime algorithm of reduced stress analysis, aiming to shape editing tasks for 3D fabrication. The edited shape at runtime is a rest shape, which may change largely during the user editing. Our goal is to estimate the stress of the edited shape under external forces.

Our reduced stress analysis takes the input of a 3D shape and user-specified external forces—both are interactively editable—and outputs the stress on the 3D shape produced by the external forces. We assume that the shape is always fixed at some part of its surface and the material parameters

are given. We also note that our reduced model does not depend on any specific shape editing model; it can work in tandem with any shape deformation tool chosen by the user.

Linear and Higher-Order Finite Elements. Computational design of 3D shapes for fabrication has seen the use of both linear [2] and higher-order [1], [4] finite elements for stress analysis. Our reduced model supports finite elements of any order in a unified framework. For simplicity, here we depict our method using linear elements, but highlight the differences for using higher-order elements. We describe the detailed numerical formulas in the supplementary document (§1 for linear elements and §2 for quadratic elements).

4.1 Elastostatic Analysis using Stress Elements

Our subspace stress analysis is built on the standard finite element discretization, starting from a weak form of Eq. (3),

$$\int_V [(\sigma_{ij})_{,j} + b_i] u_i^* \, dV = 0,$$

where V indicates the entire volume of the solid body, and u_i^* is the i-th component of a displacement test function (or virtual displacement). The finite element discretization represents the displacement field $\boldsymbol{u}$ by interpolating nodal displacement variables with a set of nodal basis functions B_k (see [49] for a detailed exposition). This leads to a discretized elastostatic stress equation

$$\sum_e \int_{V_e} \sigma_{ij} B_{k,j} \, dV = \sum_e \left(\int_{V_e} b_i B_k \, dV + \int_{\partial V_e} t_i B_k \, dS \right), \quad (4)$$

where the summation iterates over every tetrahedron e, whose volume is denoted by V_e; $\boldsymbol{t}$ is the traction applied on the object surface. In a standard FEM, σ_{ij} is substituted using the expression (2), and the equation is expressed with respect to the displacement $\boldsymbol{u}$. Using different nodal basis functions B_k, one obtains a full-rank linear system with respect to the displacement $\boldsymbol{u}$ defined on the mesh nodes.

Our method differs from the standard finite element discretization, since the stress tensor, rather than the displacement, is of our major interest. Thus, by discretizing Eq. (4) directly without using the displacement $\boldsymbol{u}$, we obtain a linear system with respect to the stress tensor

$$\mathsf{A}_\sigma [\boldsymbol{\sigma}] = \boldsymbol{b}_\sigma, \quad (5)$$

where $[\boldsymbol{\sigma}]$ denotes a long vector that concatenates flattened versions of the stress tensors (i.e., a 9×1 vector) of all tetrahedra. A_σ is a sparse matrix with a size of $3N_V \times 9N_T$, where N_V is the number of vertices and N_T is the number of tetrahedra. We defer the details of constructing A_σ and $\boldsymbol{b}_\sigma$ until the supplementary document (§1), but note that because stress tensor is a differentiation of displacement, the finite element basis functions used for discretizing $\boldsymbol{\sigma}$ is one order lower than those for discretizing $\boldsymbol{u}$. When linear elements are used for discretizing displacement, the stress tensor $\boldsymbol{\sigma}$ is piecewise constant over the tetrahedral mesh.

The same derivation can be applied with higher-order finite elements, wherein the stress tensor is no longer piecewise constant, but we can still obtain a linear system (5), in which $[\boldsymbol{\sigma}]$ is a concatenation of stress tensors defined at the Gaussian quadrature points in each tetrahedron. We describe the details in §2 of the supplementary document.

Unfortunately, Eq. (5) is not directly solvable, as A_σ is not a square matrix and not even full-rank in most cases—perhaps this is why Eq. (5) has not been used in stress analysis to our knowledge [1]. Nevertheless, it provides a starting point for us to develop a reduced stress solver.

4.2 Reduced Stress Solver

During shape editing, the user explores a space of rest shapes that are continuously varying. We expect that the resulting stresses of those shapes also vary continuously, and the stress fields can be well-approximated using a stress subspace. Suppose that such a stress subspace is expanded by a linear basis, S of size N_r (i.e., the number of columns of S is N_r). The stress vector $[\boldsymbol{\sigma}]$ can be approximated as

$$[\boldsymbol{\sigma}] = \mathsf{S}\boldsymbol{r}, \quad (6)$$

where $\boldsymbol{r}$ is a vector representing the reduced coordinates in the stress subspace. We emphasize that the stress basis S is shape-dependent. Given a user edited shape, S is a constant matrix, but it varies across different shapes. In §5, we will construct S on the fly during a shape editing process. For now, we assume that S of an edited shape is given.

Substituting Eq. (6) into the discretized stress Eq. (5), we obtain a reduced linear system with respect to $\boldsymbol{r}$,

$$(\mathsf{A}_\sigma \mathsf{S}) \boldsymbol{r} = \boldsymbol{b}_\sigma. \quad (7)$$

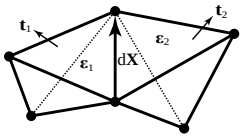
When N_r is small and $\mathsf{A}_\sigma \mathsf{S}$ is a thin matrix. This system can be solved in the least-squares sense.

Unfortunately, directly solving this least-squares problem produces a stress estimation that is very off from the true stress. This is because an arbitrary stress field might not be physically meaningful. In other words, there may not exist such a displacement field $\boldsymbol{u}$ that realizes a given stress field. In a discretized setting, this is reflected by the fact that the mapping between a stress vector of a length $9N_T$ and a displacement vector of a length $3N_V$ is not bijective.

Regularization. We, therefore, propose to regularize the stress in the subspace so that it can be physically realized through a displacement field. In the regime of linear elasticity, the infinitesimal strain $\boldsymbol{\epsilon}$ defined in Eq. (1) is linearly related to the stress tensor $\boldsymbol{\sigma}$. Moreover, $\boldsymbol{\epsilon}$ is to measure the length change of an infinitesimal segment. Specifically, the following relationship holds [51],

$$\frac{\|\mathrm{d}\boldsymbol{x}\|_2 - \|\mathrm{d}\boldsymbol{X}\|_2}{\|\mathrm{d}\boldsymbol{X}\|_2} = \boldsymbol{N}^T \boldsymbol{\epsilon} \boldsymbol{N}, \quad (8)$$

where $\mathrm{d}\boldsymbol{X}$ is an infinitesimal directional segment in the rest pose; under external forces, $\mathrm{d}\boldsymbol{X}$ is deformed into $\mathrm{d}\boldsymbol{x}$; and $\boldsymbol{N} = \frac{\mathrm{d}\boldsymbol{X}}{\|\mathrm{d}\boldsymbol{X}\|_2}$ is the normalized undeformed direction of $\mathrm{d}\boldsymbol{X}$. In a discretized setting, when two tetrahedra share an edge $\mathrm{d}\boldsymbol{X}$ (see the adjacent figure), the length changes of $\mathrm{d}\boldsymbol{X}$ measured in both tetrahedra ought to agree with each other. In other words, we expect $\boldsymbol{N}^T \boldsymbol{\epsilon}_1 \boldsymbol{N} = \boldsymbol{N}^T \boldsymbol{\epsilon}_2 \boldsymbol{N}$.



1. We note that the Hu-Washizu variational principle, which mixes different discretizations for displacement and internal pressure, has been used to prevent locking of incompressible materials (see [50]).

To incorporate this requirement into our reduced stress solver, we introduce a measure of stress realizability,

$$E_r(\boldsymbol{r}) = \sum_e \sum_{t \in \mathcal{T}(e)} l_e (\boldsymbol{N}_e^T (\mathsf{E}^t : \boldsymbol{r}) \boldsymbol{N}_e - C_e)^2. \quad (9)$$

The outer summation here iterates over all tetrahedral edges, weighted by the edge length l_e. The inner summation iterates over every tetrahedron t that is adjacent to e (denoted by the set $\mathcal{T}(e)$). $\boldsymbol{N}_e$ is the normalized direction of the edge e. $(\mathsf{E}^t : \boldsymbol{r})$ is the strain tensor of t corresponding to the reduced stress field $\mathsf{S}\boldsymbol{r}$. Because the stress and strain are linearly related (recall Eq. (1)), the strain of t is also linearly related to $\boldsymbol{r}$, and E^t is a 3rd-order tensor (see §3 of the supplementary document). Lastly, C_e is the average of edge length changes measured from all adjacent tetrahedra,

$$C_e = \frac{1}{|\mathcal{T}(e)|} \sum_{t \in \mathcal{T}(e)} \boldsymbol{N}_e^T (\mathsf{E}_t : \boldsymbol{r}) \boldsymbol{N}_e.$$

Quadratic Minimization. We now combine the least-squares problem (7) with the stress regularization (9), and introduce a minimization problem of $\boldsymbol{r}$,

$$\min_{\boldsymbol{r}} \|(\mathsf{A}_\sigma \mathsf{S})\boldsymbol{r} - \boldsymbol{b}_\sigma\|_2^2 + \alpha E_r(\boldsymbol{r}). \quad (10)$$

Both terms here are quadratic forms of $\boldsymbol{r}$. α is a scalar to balance their weights. This problem can be solved by a linear system of $\boldsymbol{r}$ with a size of $N_r \times N_r$. Because the reduced basis size, N_r, is small, we are able to solve for $\boldsymbol{r}$ interactively while the user is editing the shape. Afterward, we evaluate the stress tensor (using Eq. (6)) and the equivalent stress (e.g., von Mises or maximal principal stress).

We note that the use of the regularization term (9) as a soft constraint in (10) may not guarantee that the resulting stress field is realizable by a displacement field. It merely regularizes the subspace stress representation to better approximate a physically meaningful stress field, and thereby estimates stress much more accurately (§5.4 and §6).

Lastly, we need to choose an α value in Eq. (10) in order to obtain an accurate stress estimation. A proper choice of α depends on the specific shapes. Our algorithm automatically chooses an α value that adapts to the specific edited shape in the step of example-based stress basis construction. We will describe the algorithm of choosing α in §5.4 after presenting the construction of reduced stress basis.

Weighted Least-Squares for improving accuracy. Eq. (10) is a least-squares system, where the first term sums up the squared force differences at every vertex. In Eq. (10), every three rows with indices $3i, 3i+1, 3i+2$ in $(\mathsf{A}_\sigma \mathsf{S})$ and $\boldsymbol{b}_\sigma$ correspond to the force equilibrium at vertex i. We found that the least-squares formulation with equal weights over all vertices is not always the best choice, as the vertices who have higher stress ought to be more important in the stress analysis. This suggests that a weighted least-squares formulation can perform better.

In light of this, we first solve the least-squares problem (10) and evaluate the vector $\boldsymbol{v}$ that stacks von Mises stress (or maximal principal stress) at every vertex. We then compute a per-vertex weight vector $\boldsymbol{\omega} = \boldsymbol{v}/\|\boldsymbol{v}\|_\infty$, and use it to re-weight the rows in $(\mathsf{A}_\sigma \mathsf{S}) - \boldsymbol{b}_\sigma$ in the least-squares formulation (10). In this way, the vertices with higher stress values now contribute more to the force equilibrium term. We solve the re-weighted minimization problem once again to estimate the final stress values. This approach requires solving the least-square problem (10) twice. Fortunately, with the sparsification introduced in §4.3, the least-squares problem (10) is of a small size and no longer the computational bottleneck. In practice, we found that this double-solving strategy imposes little computational overhead, but produces higher stress estimation accuracy.

The above re-weight process can be further executed iteratively to update least-squares weights and stress. In our tests, the resulting stress usually converges after about 4-5 iterations. We found that the stresses after the second and third iterations often have relative difference less than 1%, which indicates that the two-step solution is a reasonable performance-accuracy balance.

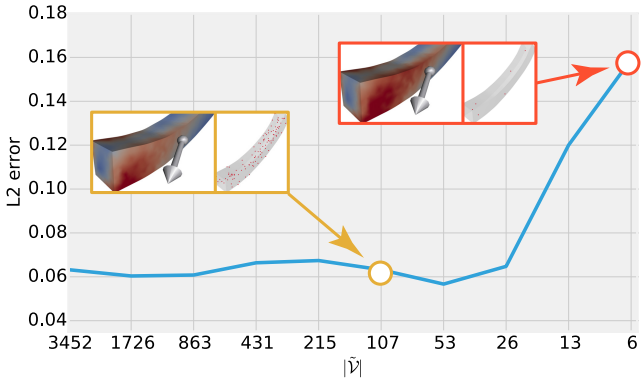


Fig. 2. **Sparsification.** We uniformly sample a subset of mesh vertices (insets) to be included in the stress estimation problem (10). The resulting stress estimation error changes from 0.06 to 0.16 when the number of sampled points decreases from 107 to 6. However, when the number of samples is larger than 26, the estimation accuracy can be well-retained.

Material. Lastly, we note that throughout the development of our reduced stress analysis method, we do not assume the homogeneity of the object material. Our method is also applicable to heterogeneous and composite materials, as long as the structures of material composition are given.

4.3 Sparsification

While we can solve Eq. (10) in real-time, constructing the $N_r \times N_r$ linear system with respect to $\boldsymbol{r}$ is still bound to the number of vertices, N_V. This is because the derivative of (10) with respect to $\boldsymbol{r}$ involves the computation of $\mathsf{S}^T \mathsf{A}_\sigma^T \mathsf{A}_\sigma \mathsf{S}$, whose complexity is proportional to N_V. For a high-resolution tetrahedral mesh, this linear system construction becomes a performance bottleneck.

We notice that every three rows of A_σ corresponds to a single node of the tetrahedral mesh (in x-, y- and z- component). Computing one matrix element of $\mathsf{A}_\sigma^T \mathsf{A}_\sigma$ requires an iteration through all rows. In the continuous setting, this is equivalent to a volume integral, which can be approximated using summations at a small set of Gaussian quadrature points. This observation inspires the simple idea of sparsifying the optimization problem (10). In particular, we select a small set of vertices $\tilde{\mathcal{V}}$ on the tetrahedral mesh and only consider the static force equilibrium at those vertices (similar to cubature schemes [26], [52]). That is, for

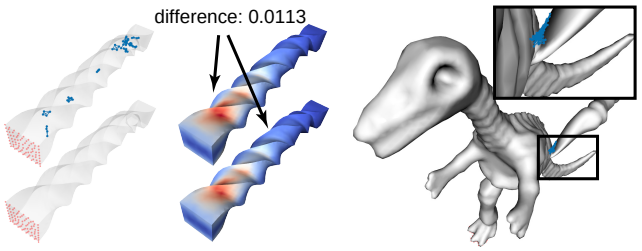


Fig. 3. **Tetrahedra inversion.** (left and middle) Our method is robust when a relatively small number of tetrahedra are inverted. Here the difference is measured in a L_2 sense (§6.2). (right) Inverted tetrahedra result in a low-quality finite element mesh, which can also show visual artifacts (inset).

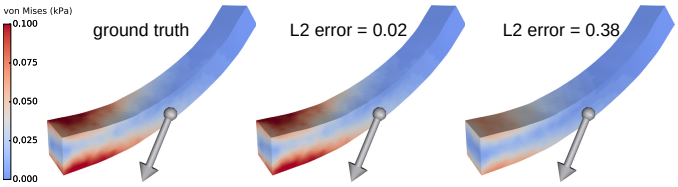


Fig. 4. **Comparison** of stress estimation with and without stress pull-back/push-forward. (left) ground-truth stress field; (middle) stress estimated using our algorithm; (right) stress estimated without pull-back/push-forward when we construct the basis.

the first term of (10), we only count the rows of $(\mathsf{A}_\sigma\mathsf{A}\boldsymbol{r} - \boldsymbol{b})$ that correspond to the vertices in $\tilde{\mathcal{V}}$. Similarly, for the second term of (10), instead of iterating through all edges, we only consider the edges that are incident to vertices in $\tilde{\mathcal{V}}$. When $|\tilde{\mathcal{V}}| \ll N_V$, the construction of the linear system becomes much faster while the accuracy is still retained.

In practice, we found that uniformly sampling the mesh vertices suffices. In all our examples, we use $|\tilde{\mathcal{V}}| \geq 0.01N_V$, which guarantees sufficient estimation fidelity. Figure 2 illustrates the change of estimation error with respect to $|\tilde{\mathcal{V}}|$. Here the error is computed as the L_2 relative error of the von Mises stress. We also examined other error metrics such as the L_∞ error (reported in §6.2).

We note that the method of sparsification is able to estimate the stress field more accurately, in comparison to the use of a coarse finite element mesh. Because the low-resolution finite element basis can not express the stress field accurately in the first place (see §6.3 and Figure 13).

4.4 Improving Mesh Quality

Our reduced stress analysis is designed to be independent of any specific mesh deformation method. However, the accuracy of a finite element stress analysis depends on the tetrahedral mesh, which may degrade or even invert during user edits (Figure 3-right).

Developing an interactive remeshing algorithm is out of the scope of this paper. Instead, building on the method of Tertois et al. [53], we adopt a lightweight adaptation method to improve mesh quality at runtime, while retaining the mesh topology. First, we check if a mesh vertex is too close to its opposite face or if there are already inverted tetrahedra. We add all those candidate vertices into a set $\hat{\mathcal{V}}$. Next, we adjust their positions to improve the tetrahedral quality. Let $\hat{\mathcal{T}}$ denote the set of tetrahedra that are incident to the vertices in $\hat{\mathcal{V}}$. We use $\boldsymbol{v}^*$ and $\boldsymbol{v}$ to denote two vectors that stack the *current* and *adjusted* positions of vertices in $\hat{\mathcal{V}}$, respectively. To find the adjusted positions, we formulate a constrained minimization problem with respect to $\boldsymbol{v}$,

$$\min_{\boldsymbol{v}} \|\boldsymbol{v} - \boldsymbol{v}^*\|_2^2, \quad \text{s.t. } V_t(\boldsymbol{v}) \geq |V_t(\boldsymbol{v}^*)| \cdot (1+\delta), \ \forall t \in \hat{\mathcal{T}},$$

where $V_t(\boldsymbol{v})$ is the volume of the tetrahedron t. The (positive) threshold δ is to force the adjusted vertices to form well-shaped tetrahedra. To ensure interactive solving, the cubic constraints with respect to $\boldsymbol{v}$ are simply linearized by discarding higher-order terms (see Tertois et al. [53] for details). In practice, we use MOSEK [54] to solve the resulting quadratic programming problem.

Discussion. Thanks to the least-squares formulation (10), our method is generally robust when a small number of tetrahedra are inverted, since the well-shaped tetrahedra will contribute predominately to the objective function. As shown in Figure 3-left, when the inverted tetrahedra (shown in blue) are sparse over the mesh, the resulting stress shows only 1.2% relative L_2 error (Figure 3-middle). Nevertheless, when more inverted tetrahedra emerge, mesh improvement is needed to maintain the FEM accuracy.

5 Construction of Stress Basis

We now describe the construction of efficient stress basis for our subspace stress analysis. Taking as input a set of example deformations, we first perform the standard finite element stress analysis on every example shape and obtain a set of stress fields. We then extract a stress basis by analyzing the principal components of those stress fields.

At first glance, we can create stress basis by applying a principal component analysis (PCA) over the example stress fields. Unfortunately, this algorithm fails to produce efficient basis, often resulting in unreliable stress estimation (Figure 4). This is because the input shape exemplars can be quite distinct from each other and the stress fields are thus computed with respect to very different rest poses. Consequently, these stress fields contain no clear "principal components" to represent them. In light of this, we seek to transform the stress fields in order to unify their rest poses before analyzing their principal components.

5.1 Algorithm

Input. The input of our basis construction algorithm consists of N_S example shapes, each represented by a mesh of N_T tetrahedra. In practice, we choose the example shapes by deforming a rest-pose mesh, so we ensure all the example shapes have different geometry but the same topology. The example shapes can be user-supplied to reflect the intended subspace of shapes that will be explored; they can also be automatically sampled in the deformation space, depending on specific editing applications. For each shape exemplar, the boundary conditions (such as the fixed vertices) and external forces (such as gravity) are given. When one designs shapes for fabrication, the fixed regions of the shape are often known (e.g., red vertices in Figure 1-left). One

Algorithm 1 Basis precomputation

Require: shape exemplars $S_i, i = 1...N_S$
ground-truth stress fields $\boldsymbol{\sigma}^i$ computed on S_i
1: Select a reference shape S_0
2: **for** $i = 1 : N_S$ **do** ▷ iterate over all example shapes
3: **for** $t = 1 : N_T$ **do** ▷ iterate over all tetrahedra
4: Evaluate deformation gradient $\boldsymbol{F}_{it}$ of S_i w.r.t. S_0
5: $\tilde{\boldsymbol{\sigma}}_t^i \leftarrow$ pull_back$(\boldsymbol{\sigma}_t^i, \boldsymbol{F}_{it})$ ▷ §5.2
6: **end for**
7: Concatenate fattened $\tilde{\boldsymbol{\sigma}}_t^i$ for all t into $\boldsymbol{p}^i$
8: Attach $\boldsymbol{p}^i$ into a data matrix B
9: **end for**
10: Extract basis S using SVD(B)

might also explore different external forces (e.g., by setting different gravity directions). In that case, each set of external forces together with the corresponding shapes are used as individual training examples.

Computation. As outlined in Algorithm 1, we first compute the stress field for every example shape. When linear finite elements are used, the stress is piecewise constant. Let $\boldsymbol{\sigma}_t^i$ denote the stress tensor of the t-th tetrahedron in the i-th training example ($t \in \{1...N_T\}$ and $i \in \{1...N_S\}$). Instead of using $\boldsymbol{\sigma}_t^i$ directly for basis construction, we first "pull back" all the stress tensors to express them with respect to a single fixed rest pose (line 5 of Algorithm 1). We denote the pull-back of $\boldsymbol{\sigma}_t^i$ as $\tilde{\boldsymbol{\sigma}}_t^i$, a 3×3 matrix. We then reshape every (unsymmetric) $\tilde{\boldsymbol{\sigma}}_t^i$ into a 9×1 vector for all the tetrahedra, and concatenate them into a single vector $\boldsymbol{p}^i$ (line 7 of Algorithm 1). We then perform PCA over these vectors to obtain a set of stress basis. In practice, we choose to run the singular value decomposition (SVD) over the matrix that stacks all $\boldsymbol{p}^i$ as column vectors.

When higher-order finite elements are used, the stress is not constant in every tetrahedra anymore. Thus, when assembling a stress vector $\boldsymbol{p}^i$ in Algorithm 1, we sample quadrature points in each tetrahedron. As mentioned earlier, the finite element basis functions of stress tensor are one order lower than the displacement basis functions. Therefore, for quadratic elements, we sample 4 points. We pull back the stress tensors defined at those quadrature points using the deformation gradients of the tetrahedron in which the points are located. The rest of the computation remains the same as for the case of linear finite element meshes.

5.2 Stress Tensor Pull-Back

With the example stress fields computed from the linear elastostatic analysis on different rest poses (i.e., shape exemplars), we now describe the stress pull-back, in order to reconcile those stress fields such that they are all expressed with respect to a common rest pose. We first choose an arbitrary shape S_0 from the N_S example shapes as the reference shape. Consider one example shape S_i, $i \in \{1...N_S\}$. The stress tensor $\boldsymbol{\sigma}_t^i$ of a tetrahedron t describes the force acting on an infinitesimal area $\mathrm{d}\boldsymbol{a}$ in S_i. If we treat the shape S_i as a deformation from S_0, then S_0 has an infinitesimal area $\mathrm{d}\boldsymbol{A}$ corresponding to $\mathrm{d}\boldsymbol{a}$; and they have a relationship, $\mathrm{d}\boldsymbol{a} = J_{it}\boldsymbol{F}_{it}^{-T}\mathrm{d}\boldsymbol{A}$, where $\boldsymbol{F}_{it}$ is the piecewise constant deformation gradient of S_i with respect to S_0 in a tetrahedron

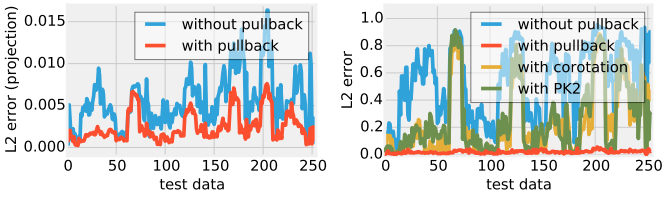


Fig. 5. **Efficacy of pull-back/push-forward.** (left) With the pull-back/push-forward operations, the difference between a stress vector and its subspace projection decreases about 60%. (right) The estimation error with the pull-back/push-forward operations is significantly lower than that without the operations, and is also much lower than that with corotational and with second Piola-Kirchhoff stress tensors. Both plots are tested with a bar model. See more explanation in §6.2.

t, and $J_{it} = \det\boldsymbol{F}_{it}$ is its determinant (see [50]). This allows to express the stress field of S_i with respect to S_0,

$$\tilde{\boldsymbol{\sigma}}_t^i = J_{it}\boldsymbol{\sigma}_t^i\boldsymbol{F}_{it}^{-T}, \tag{11}$$

which is precisely our pull-back operator. We note that while this operator looks almost identical to the first Piola-Kirchhoff stress tensor [50] in continuum mechanics, they have different physical interpretations: in (11), $\boldsymbol{\sigma}_t^i$ is computed by solving the linear elasticity problem using S_i as the rest pose, while $\boldsymbol{F}_{it}$ is computed based on the tetrahedrons of S_0 and S_i, describing the local geometric deformation from S_0 to S_i. In contrast, for the first Piola-Kirchhoff stress tensor, $\boldsymbol{F}$ describes the physical deformation that leads to the internal stress $\boldsymbol{\sigma}$.

5.3 Stress Basis Push-Forward

We construct the stress basis following Algorithm 1 as a preprocessing. The output of the preprocessing is a set of stress basis, denoted by a set of vectors $\hat{\boldsymbol{p}}_i, i = 1...N_r$, where N_r is the size of the basis ($N_r \leq N_s$). Every $\hat{\boldsymbol{p}}_i$ stacks the basis stresses on all the tetrahedron, but they are constructed with respect to a fixed rest pose, namely the shape S_0 in §5.2. At runtime, when the user updates a shape $\hat{S}$, it becomes the rest pose for estimating the stress. We, therefore, need to "push forward" the stress basis to express it with respect to $\hat{S}$ rather than S_0.

Our runtime push-forward operator is an inverse of the pull-back operator. Concretely, for every precomputed stress basis $\hat{\boldsymbol{p}}_i, i = 1...N_r$ and every tetrahedron t, we compute

$$\boldsymbol{s}_t^i = \hat{J}_t^{-1}\hat{\boldsymbol{\sigma}}_t^i\hat{\boldsymbol{F}}_t^T, \tag{12}$$

where $\hat{\boldsymbol{\sigma}}_t^i$ is the i-th basis stress tensor for tetrahedron t, unstacked from $\hat{\boldsymbol{p}}_i$. $\hat{\boldsymbol{F}}_t$ is the deformation gradient of tetrahedron t that deforms the shape S_0 into the current shape $\hat{S}$, and $\hat{J}_t$ is its determinant. The resulting stress tensor $\boldsymbol{s}_t^i$ is the i-th basis stress for the tetrahedra t of the current shape $\hat{S}$. We stack them into a vector and further put the basis vectors into the matrix S as used in §4.2.

We note that the pull-back and push-forward operators are of great importance in our algorithm. As shown in Figure 5-right, they improve the stress estimation accuracy tremendously. We present detailed validation in §6.2.

5.4 Precomputation of Regularization Weight α

With the set of training poses and their ground-truth stresses computed, we need to choose a weight α to balance the

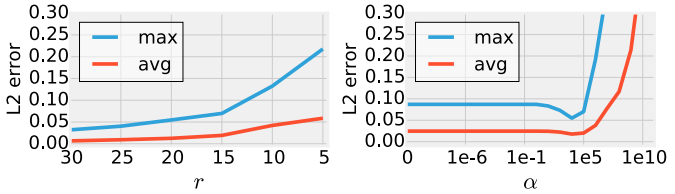


Fig. 6. **Stress error affected by parameters.** (left) Estimation error changes with the size of stress basis, tested with the dinosaur model. (right) Estimation error changes with different α values used in Eq. (10), tested with the bar model. In both plots, the errors are evaluated using all test poses; both the averaged and maximum L_2 error are shown (see §6.2 for error computation).

terms in the objective function (10). A common practice of balancing the energy terms in Eq. (10) is through normalizing both terms. However, the normalization requires an estimation of the varying ranges of both terms, and for an arbitrary shape, this is nontrivial.

Instead, we automatically choose the best weight α using cross-validation. In particular, we define 25 different α values, $10^{-12}, 10^{-11}, ..., 10^{11}, 10^{12}$. For every training pose S_i, we remove it from the training data set and use the rest of training poses to construct the stress basis. We then repeatedly use the basis to estimate the stress of S_i, each time with different α values. For every estimation, we compare it with the ground-truth stress of S_i and compute an L_2 error (§6.2), which measures the quality of using that particular α. For a single α value, we repeat this error measurement for every training pose S_i, $i = 1...N_s$, and average the error values across all poses. Lastly, we choose the α value with the lowest average error to use at runtime.

As shown in Figure 6-right, the value of α indeed affects the estimation accuracy. Our algorithm automatically adapts α to specific shapes (Table 2). We observed that the α values vary significantly across different shapes, because the scales of the two energy terms in Eq. (10) can differ substantially.

Discussion. Our reduced model is directly built on the stress field without the use of displacements. Thereby, the force equilibrium in Eq. (10) is computationally more efficient. Using displacement to construct the subspace can be another choice, with a potential advantage that the resulting stress is guaranteed to be realizable. However, it is unclear how to define the pull-back/push-forward operations for displacement. As described in §5.3, these operations have proven essential to the expressiveness of the subspace representation, for which we leave as an interesting future work.

6 Validation and Results

Hardware and software. We run our experiments on a desktop PC equipped with an Intel i7-3770K CPU. Our program runs in 8 threads using OpenMP [55]. We built a prototyping shape deformation tool using the libigl library [56]. In addition, we use COMSOL [6], a widely used commercial finite element analysis software, to compute the stresses of our training shapes as well as the ground-truth stress values for evaluating stress estimation error.

6.1 Preparing training and test data

In our experiments, we implement shape deformation using linear blend skinning (LBS) with the bounded biharmonic weights [57]. For every shape in the training examples, body forces such as gravity are also assigned by sampling different body force directions to add in the training data set. Because of the linear relationship between external force and deformation displacement, there is no need to sample multiple force magnitudes in the training data. Lastly, with the LBS model, we automatically generate a set of test poses by interpolating the configurations of handles (or bones) presented in the training examples. The test poses are used to validate the accuracy of our reduced stress estimation. They are uniformly sampled in the space of handle positions. We ensure that all the poses vary in a reasonably large range. For example, the range of peak von Mise stresses (indicating where a model is weakest) for the lamp dataset is $[5.5\,\mathrm{MPa}, 19.1\,\mathrm{MPa}]$.

TABLE 1
Statistics of experimental models.

Model	#Vertices	#Elements	#Order	#DoFs	#Train	#Test
bar	3,452	15,602	linear	10,356	39	254
horse	8,253	34,019	quadratic	167,034	21	399
bird	10,876	43,299	quadratic	216,072	27	525
dinosaur	14,029	57,830	linear	42,087	33	359
fertility	17,492	82,271	quadratic	377,667	18	136
lamp	20,428	94,184	linear	61,284	42	448
gargoyle	30,455	145,076	linear	91,365	33	457
armadillo	118,595	591,547	linear	355,785	33	387

From left to right: number of vertices, number of elements, order of finite elements, degrees of freedom in full-space finite element computation, number of training examples, and number of test data.

We use all the training and test shapes together with the sampled force settings to compute stresses using COMSOL, which performs the stress analysis using a standard FEM. In all stress computation, we use the physical parameters of a typical fabrication material (i.e., hard plastic) used in 3D printing: we set Young's modulus $E = 1.0\mathrm{e}9\,\mathrm{N/m^2}$, Poisson's ratio $\nu = 0.45$, and density $\rho = 958.1\,\mathrm{kg/m^3}$.

Table 1 lists the statistics of our experimental data. We choose test geometries representing different types of objects, such as creatures (dinosaur), artistic designs (fertility) and man-made models (lamp). Their numbers of DoFs range from tens of thousands to hundreds of thousands, as used in typical personal fabrications [2]. Among these models, we use quadratic finite elements in bird, horse and fertility examples; other examples use linear finite elements. When computing training and ground-truth stresses in COMSOL, we choose either type of finite elements correspondingly. We note that in Table 1 when quadratic elements are used, mesh vertices are just part of the finite element nodes, which also include the edge nodes. Thus, the numbers of DoFs are larger than the number of vertices. In the supplemental video, we also report the reduced model precomputation time for specific examples.

6.2 Accuracy

Metrics. For every test shape T_i, we compute its stresses using our reduced method with no more than 50 stress basis vectors. We also compute the ground-truth stresses using the standard FEM in COMSOL, and then compute the equivalent stress (both von Mises stress and the maximal principal stress as summarized in Table 2) in all tetrahedron.

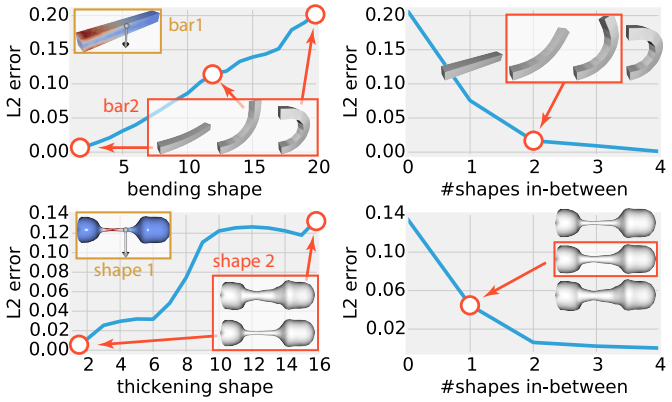


Fig. 7. **Validation of bending and thickening.** (top left) Estimation error changes w.r.t. the bending angle of the bar. (top right) Estimation error changes w.r.t. the number of in-between bending shapes used for subspace construction. (bottom left) Estimation error changes w.r.t. the thickness of the dumbbell. (bottom right) Estimation error changes w.r.t. the number of in-between thickening shapes used for training.

Stacking them together we get an estimated equivalent stress vector $\boldsymbol{v}_i$ for the shape T_i and the corresponding ground-truth stress vector $\boldsymbol{v}_i^*$. In our validation, we examine two types of errors, the L_2 error (e_2^i) and L_∞ error (e_∞^i) defined respectively as

$$e_2^i = \frac{\|\boldsymbol{v}_i - \boldsymbol{v}_i^*\|_2}{\|\boldsymbol{v}_i^*\|_2} \quad \text{and} \quad e_\infty^i = \frac{\text{abs}(\|\boldsymbol{v}_i\|_\infty - \|\boldsymbol{v}_i^*\|_\infty)}{\|\boldsymbol{v}_i^*\|_\infty}.$$

The L_2 error reflects the stress error averaged over an entire solid body, while the L_∞ error indicates the error of peak stress value on the object. Given a test shape, the errors are computed per test pose.

Validation on simple examples. We perform two simple experiments to validate the key idea of our reduced method. The first one is through bending a bar. We construct the stress basis with a training set consisting of only two bars; one is straight and the other is bent (Figure 7). To measure the accuracy, we generate 30 testing bars by uniformly interpolating the two training bars, compute their stresses, and evaluate the maximum L_2 error e_{max} over all test bars. Next, we repeat the above test incrementally: each time we bend the second training bar slightly more and regenerate the testing bars. Figure 7 (top left) shows the e_{max} curve for 20 increment steps. As the training bars become more distinct from each other, their stresses have a larger variance, and thus our stress subspace becomes less representative. In an extreme case (e.g., a C-shape bar), the error goes up to about 20%. We can reduce the estimation error by including more training shapes whose shapes are between the two training bars. Figure 7 (top right) shows that the error decreases as a few in-between shapes are added in the training set. With only two additional shapes, the error drops from 20% to 1.7%. The second experiment is through thickening a dumbbell. Figure 7 (bottom left) shows the e_{max} curve when the slim stem is thickened. Similarly, the error drops from 14% to 4.5% by adding only one in-between shape (Figure 7 (bottom right)) for training.

We observed that the stress fields of the blended shapes change smoothly as we vary the blending weights. For instance, as we progressively bend a straight bar, the estimated weight of the stress basis corresponding to the straight bar's stress decreases and the weight corresponding to the bent bar's stress increases. This confirms that the space of stress fields can be effectively reduced to enable fast stress estimation.

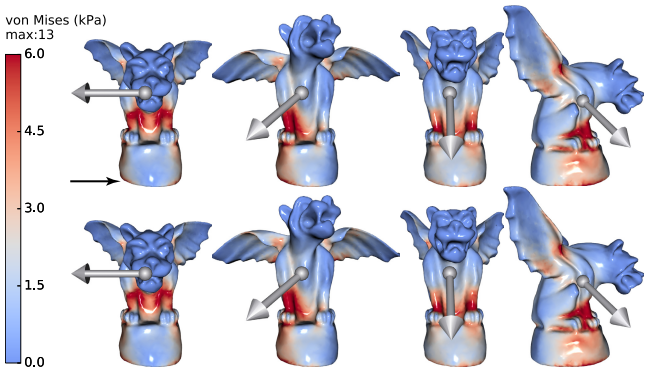


Fig. 8. **Gargoyle.** the von Mises stresses estimated by our method (top row), and the ground truth stresses computed by FEM (bottom row).

Validation on complex examples. We also validate over the data set described in §6.1. Figure 10 plots e_2^i (blue curves) of individual test shapes indexed by the abscissa of the plots. Those plots also indicate the averaged (yellow) and maximum (red) L_2 error over all test poses. In all tests, the L_2 errors are no more than 6%. While Figure 10 provides the plots of numerical errors, we also show some visual comparisons in Figure 8, 9 and 11. For example, the estimated stress field and the ground-truth stress on the Gargoyle are visually almost indistinguishable (Figure 8).

Table 2 reports the averaged (e_{avg}) and maximum (e_{max}) L_2 errors over all test shapes of individual examples. Further, we compute the worst-case L_∞ error and report in Table 2 as e_{max}^* (i.e., $e_{max}^* = \max_i e_\infty^i$). In addition, Table 2 also lists the maximum L_2 error ($\tilde{e}_{max}$) and L_∞ error ($\tilde{e}_{max}^*$) for the maximal principal stress. In general the errors of estimated maximal principal stress are similar to the errors of von Mises stress. All the error values are less than 6.8%, indicating that our reduced stress analysis is able to achieve reasonably accurate stress estimation. We also examined the error of stress tensor, measured in a Frobenius-norm sense. The error of stress tensor is slightly larger than that of the equivalent stress. For all our tests, the largest error of stress

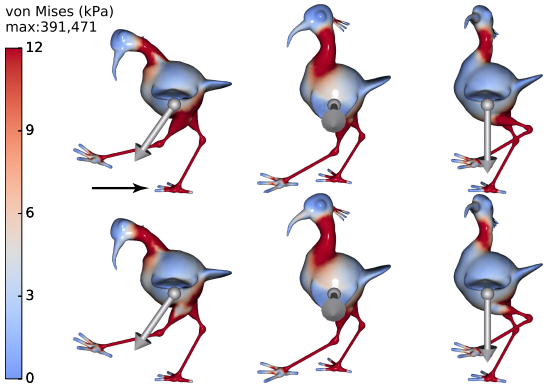


Fig. 9. **Bird.** The von Mises stresses estimated by our method (top row), and the ground truth stresses computed by FEM (bottom row). The black arrow (similarly for other figures) indicates the fixed part.

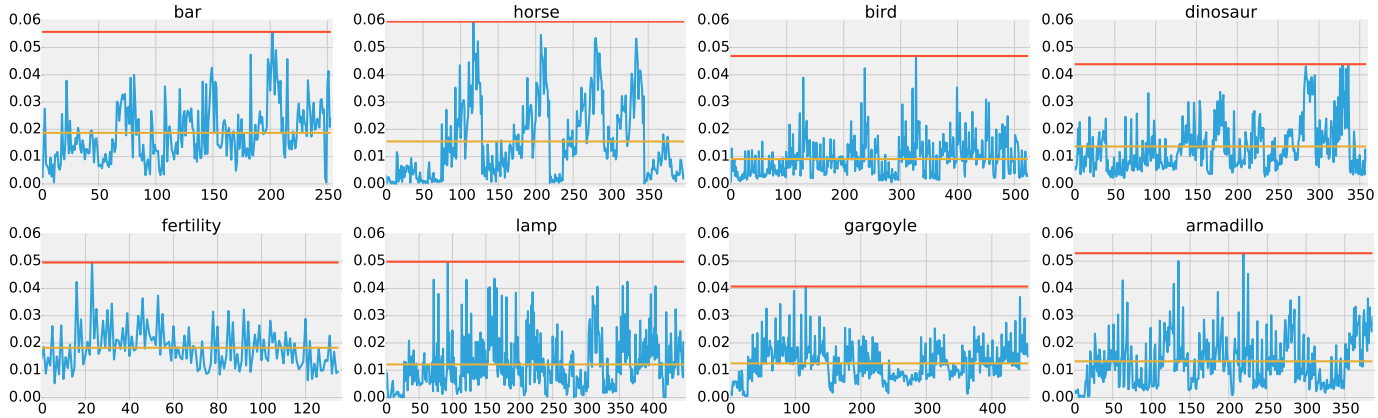


Fig. 10. **Errors of all test poses.** The abscissa represents the indices of test data, and the ordinate represents the relative errors between the estimated stresses and the ground truth stresses.

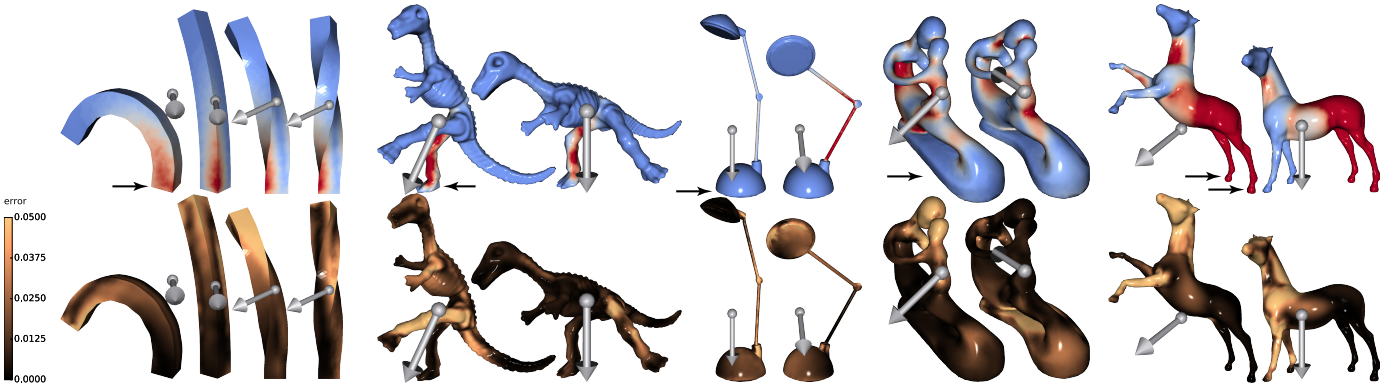


Fig. 11. **Error map.** Given the von Mises stresses (top row) estimated by our method, we compute maps of the relative errors (bottom row) by comparing with the ground-truth. In all the examples, the errors in the regions with large stress are always lower. In other words, the red regions in the top row always have relatively small estimation errors (black colors in the bottom row). This is advantageous in practice, as high-stress regions are more important in many structural design tasks.

TABLE 2
Statistics of stress estimations

Model	Estimation Parameters			Estimation Errors					Estimation Timings (sec)		
	Reg(α)	#Bases(r)	#Points($\lvert\bar{\mathcal{V}}\rvert$)	e_{max}	e_{avg}	e^*_{max}	$\tilde{e}_{max}$	$\tilde{e}^*_{max}$	Full	Est (SU)	SpEst (SU)
bar	10	30	53	0.056	0.018	0.035	0.046	0.029	1.9	0.36 (5×)	0.026 (73×)
horse	1e-12	20	100	0.059	0.016	0.059	0.059	0.059	12.4	0.89 (14×)	0.071 (175×)
bird	1e-12	20	100	0.047	0.009	0.026	0.048	0.026	16.6	1.17 (14×)	0.083 (200×)
dinosaur	10	18	100	0.044	0.014	0.041	0.046	0.041	3.8	0.65 (6×)	0.042 (90×)
fertility	1e7	16	200	0.050	0.018	0.063	0.048	0.068	22.6	1.73 (13×)	0.165 (137×)
lamp	1e-8	41	200	0.050	0.012	0.050	0.050	0.036	3.9	2.34 (2×)	0.158 (25×)
gargoyle	1	28	300	0.041	0.012	0.062	0.047	0.056	6.7	2.23 (3×)	0.153 (44×)
armadillo	1e-5	32	1000	0.052	0.013	0.052	0.053	0.055	27.9	11.04 (3×)	0.419 (67×)

From left to right: regularization parameter (Reg), number of reduced bases (#Bases), number of sampled points for sparse estimation (#Points), maximum L_2 estimation error for von Mises stress on test data (e_{max}), average L_2 estimation error for von Mises stress on test data (e_{avg}), maximum L_∞ estimation error for von Mises stress on test data (e^*_{max}), maximum L_2 estimation error for maximal principal stress on test data ($\tilde{e}_{max}$), maximum L_∞ estimation error for maximal principal stress on test data ($\tilde{e}^*_{max}$), time for full FEM computation (Full), time for estimation by using all points (Est), time for estimation by using only sparsely sampled points (SpEst). The speedups (SU) are compared with full FEM computation.

tensor is less than 11%.

Our method supports different types of shape deformations, such as bending/twisting (the Bar), thickening (leg of the Bird), and stretching (ear of the Armadillo). We found that to support complex deformation operations, many stress bases are often needed. In artistic design, shapes with non-zero genus are common, and editing the shape of its silhouette is often needed from the designer's point view, but this would change the volume of the model. The Fertility model serves as an example to test these cases and shows that our method can handle such editing well. We use the Lamp as an example of man-made shapes. The big lampshade generates stress concentrations near the slender struts and joints. Consequently, the pose modification leads to a fast change of the stress fields. The estimation on this model works fairly well using sufficient training data.

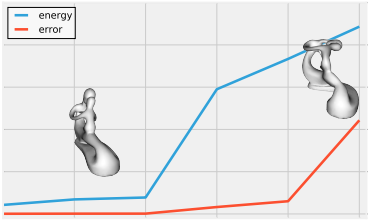


Fig. 12. The estimation errors (red) and energy function values (blue) of interpolated (left) and extrapolated (right) shapes. As the edited shape moves out of the subspace, both curves increase and share a similar trend.

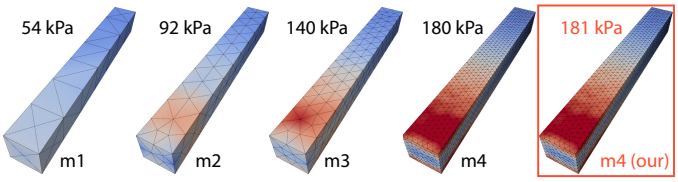


Fig. 13. Standard finite element stress analysis using meshes with four different resolutions (from coarse to dense). The rightmost shows the stress distribution estimated using our reduced method. The estimated peak stress values are also listed.

We validate the efficacy of our pull-back and push-forward operators by comparing the L_2 stress errors produced with those operators enabled and disabled. We use the bar model and perform two experiments and show them in Figure 5. For every test pose (indexed by the abscissa in the plots), we construct two bases, with and without the pull-back/push-forward operations. In the first experiment, we use the ground-truth stress vector and project it to the subspaces spanned by both bases respectively. We compute the L_2 difference between the original and the projected stress vectors. As shown in Figure 5-left, with pull-back/push-forward, the resulting subspace can express the stress vector more closely. In the second experiment (Figure 5-right), we directly compare the stress errors using both bases and confirm that our pull-back/push-forward operations indeed improve the estimation accuracy greatly.

In addition, we test other possible pull-back/push-forward definitions. In particular, one can use corotational and second Piola-Kirchhoff stress tensors to pull-back/push-forward stress basis. However, we found that those alternatives are much less efficient in comparison to our pull-back/push-forward operations (Figure 5-right).

While we show in the above tests that our reduced stress analysis produces fairly accurate estimations, it is possible that the user deforms a shape completely out of the subspace spanned by the stress basis. In this case, we provide a metric to indicate the estimation quality, as the ground-truth stress would not be available in a practical application. As a preliminary test, we found the energy function value of Eq. (10) can serve as such an indicator. Figure 12 shows the change of energy function values along with the estimation errors with respect to the change of shapes. Both curves share roughly the same trend. Therefore, a large energy value of Eq. (10) can indicate that the current shape is probably out of the current stress subspace, suggesting a compromised estimation accuracy.

Table 2 also lists the estimation parameters (i.e., α, r, and $|\bar{\mathcal{V}}|$), whose values affect the estimation accuracy. For the relationships between different choices of the estimation parameters and the estimation errors, we refer to Figure 2, Figure 5, Figure 6, and the discussions therein.

6.3 Performance

We measured the computation time of our method and compared with the full-space FEM computation (COMSOL) (Table 2). COMSOL allows the user to choose different state-of-the-art linear solvers. In practice, we use the direct solver PARDISO for linear elements and the Multigrid solver with GMRES for quadratic elements, which has proven much faster than direct solvers on large-scale problems. Finite element solvers in COMSOL take full advantage of all available CPU processors.

Across all the experiments, our method gains 1-2 orders of magnitude speedups over the standard finite element analysis. Such performances allow real-time or interactive stress estimation during shape design, as demonstrated in the supplemental video. Even for complex models like the Armadillo, which has more than 300,000 DoFs, we can still maintain interactive performance. We also tested a tetrahedral mesh with more than 2 million elements: As mentioned in §1, COMSOL takes 8 minutes to perform the stress analysis, whereas our method finishes the computation in 2.3 seconds, resulting in a 208× speedup.

We also investigated the performance gain resulting from individual parts of our method. Without the sparsification (§4.3), already we are able to achieve 2-14× speedups compared to the standard FEM (second last column of Table 2). With the sparsification, the stress estimation becomes even faster (last column of Table 2).

A few parameters affect the performance of the method. One is the number of DoFs of the finite element mesh. In general, a larger speedup can be obtained from our reduced method when there is a large number of DoFs. For instance, as shown in Table 2, models using quadratic elements all have more than $100\times$ speedups. Other parameters that affect the performance-accuracy trade-offs include the size of the reduced stress basis (r) and the number of sampled vertices ($|\bar{\mathcal{V}}|$) in sparsification (§4.3). Figure 2 and 6 show their influences of the stress estimation's L_2 errors.

Low resolution finite element computation. We also compare the accuracy and cost of our method with the finite element stress analysis using different mesh resolutions. Mesh resolution can affect the stress analysis accuracy. As illustrated in Figure 13, the standard finite element analysis on a low-resolution mesh m2 (313 elements) has a comparable runtime cost with our reduced method on a high-resolution mesh m4 (15602 elements), but produces 49% error relative to the high-resolution finite element analysis (on m4). In contrast, our reduced method produces much more accurate stress estimation ($e^*_{max} = 3.5\%$, see Table 2) with a comparable time cost.

6.4 Application: Stress-Aware Design

We now augment the interactive shape editing tool with our reduced stress analysis method to enable stress-aware design. Enjoying the accuracy and high performance of our method, our stress-aware editing tool demonstrates the application of our method in the following three ways.

Function I: Overstress alert. Straightforwardly, we detect the maximum stress of the designed shape immediately

after the shape is changed, and check whether the stress is beyond the yielding threshold, σ_{yield}, of the material. A prompt check enables the shape editing tool to alert the user when a structurally weak shape is made (Figure 14 (b)). In addition, we highlight the overstressed regions of the shape, that is, the areas in which the equivalent stress is larger than σ_{yield} (see Figure 14(b) and video). This on-the-fly alert in the design loop provides users immediate visualization of the structural information of their designs and allows them to rectify whenever needed.

Function II: Shape rollback. In addition to the overstress alert, we allow the shape editing tool to automatically adjust shapes when the stress exceeds the yielding threshold. We provide two shape adjustment operations. The first one is shape rollback. For instance, suppose when the user changes the stem of the Lamp model from pose p (Figure 14 (a)) to pose p^* (Figure 14 (b)), the stress exceeds the yielding threshold. This indicates that the shape is changed from a structurally valid state to an invalid state. One option to avoid the invalid state is to rollback the shape changes toward p until the entire shape is within the yielding threshold, while it still respects the user's shape changes as much as possible. Therefore, we find the latest valid shape (Figure 14 (c)) on a path that changes from p to p^*. In particular, we record the shape change path $\{h_i | i = 1, 2, \ldots, K\}$ while the user deforms the shape. Here h_1 is the transformation states of all LBS bones and handles of p, h_K is for the ending shape p^*, and K is the number of poses in the path. We find the last shape on the path with valid stress values. This is efficiently implemented using a binary search on the path. Each search step requires a stress analysis of the corresponding shape on that path, resulting in $O(\log K)$ stress solvings. Because of the fast performance of each reduced stress analysis, this adjustment can be finished interactively.

Function III: Automatic thickening. Sometimes, it is hard, if not impossible, to fix the overstressed regions of a shape by merely bending and twisting the current shape. We found that in many of these cases, bone thickening operation can effectively resolve those overstressed regions. When the user enables this option in our shape editing tool, our system first detects the bones $B = \{b_i\}$ whose corresponding regions have excessive stresses, and then incrementally thickens B until all the regions have stresses within the yielding threshold (Figure 14 (e)). Particularly, we conduct a two-step iterative process: (1) applying anisotropic scaling $s = \text{diag}(1, r, r)$ [2] into the current transformation of B to obtain a new shape produced with the thickened bones B; (2) executing our reduced stress analysis on the new shape and checking whether the excessive stresses are resolved. We break the loop when no excessive stresses are found or the maximum number of iterations is reached. The thickening factor r is set to a constant value 1.1. This operation requires training exemplars with various thicknesses, such as the Bird legs shown in the video. After precomputing the reduced model, our editing tool is fast enough to maintain interactive performance (see video).

2. Here we suppose the major axis of the bone in a material space is along the x-direction, so that the expanding directions are along the y- and z- directions, normal to the x-axis.

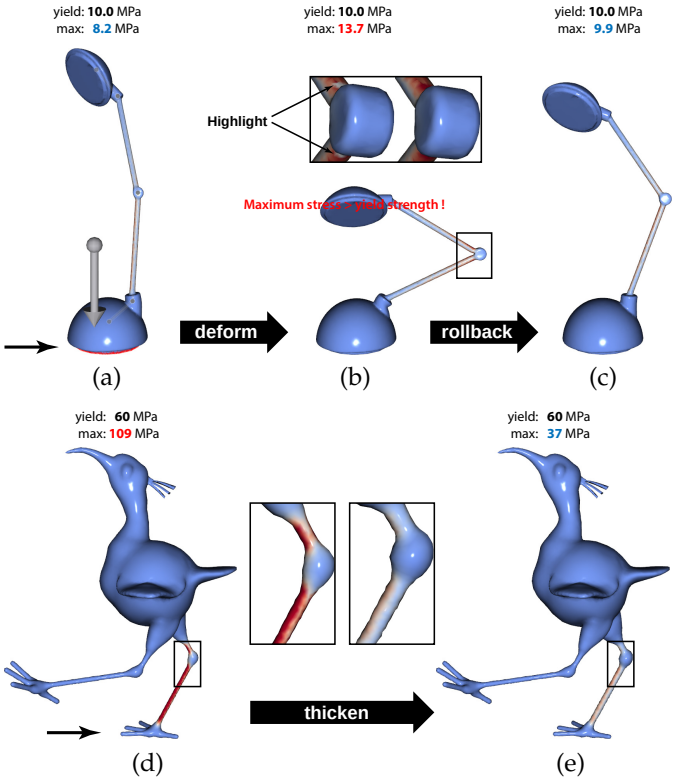


Fig. 14. **Stress-aware design.** (a) An input shape has a valid stress. (b) The shape is deformed by the user into an invalid pose (i.e., maximum equivalent stress becomes larger than a 10 MPa of yielding strength). (c) Shape rollback operation is triggered to pull the shape back to a valid pose with 9.9 MPa maximum stress. (d) The stress is till too large on the supporting leg. (e) Bone thickening operation is triggered to make the shape valid under a 60 MPa of yielding strength.

7 Limitations and Future Work

We have introduced an example-based subspace stress analysis method for stress-aware shape design. Compared with the full FEM computation, our method runs up to two orders of magnitudes faster, and thereby enables responsive and fluent design experiences. Meanwhile, our method retains the accuracy of estimated stresses comparable to ground truth stresses. We demonstrate the high accuracy and performance of our method on various types of models.

We envision our reduced stress analysis method as a key computation component useful in various design and analysis tasks. Harnessing the reduced stress analysis with little computational overhead, users could explore the shape space interactively using their favorite deformation tools, but with the resulting stress computed and visualized on the fly. Moreover, the reduced stress analysis opens up new opportunities for parametric shape optimization. Provided a space of possible shape designs, one can sample a set of shape parameters and precompute the stresses of the sampled shapes. Consequently, any design optimization that requires stress distribution information, e.g., metamaterial design, can be executed to search for the best parameters. Herein the reduced stress analysis takes a crucial role to expedite the expensive FEM computation.

In this work, we assume that the locations of surface tractions and fixed boundaries remain unchanged during

editing, while their magnitudes and directions are allowed to change. It is possible to extend our method for supporting limited changes of surface tractions and fixed boundaries: we can sample a set of surface tractions and fixed boundaries and construct a training set for the stress basis. Indeed, many products have limited locations to fix (or exert traction force). For instance, a mug handle has a specific way to be held in hand. But if there are many different locations for applying traction forces or fixed boundaries, it becomes impractical to sample and cover all the possibilities.

We are also interested in extending our method to handle periodic or probabilistic boundary conditions. These boundary conditions are useful when, for example, fatigue is taken into account—the stress range of a periodic motion is more informative than the peak stress [58]. In other tasks (like jewelry design), it is hard to predetermine external force locations or fixed boundaries. An interesting future work is to develop a reduced worst-case structural analysis method [2] that enables interactive computation of the weakness map during runtime shape editing.

While our method offers high performance for linear elastostatic stress analysis, the speedup decreases when we increase the size of the stress basis in order to obtain higher estimation accuracy. It is interesting to explore the use of modal analysis or other deformation basis to factorize large deformations at runtime. Thereby, we may build a reduced stress basis of smaller size for runtime estimation.

In industrial product design, different from bending and twisting operations, shape editing operations (e.g., sketch based extrusion) need to keep the semantic (parametric) design constraints [59], which usually leads to a large volume change, necessitating a complete remeshing. This problem poses new challenges for precomputing a reduced space.

Moreover, the decreased quality (or inversion) of the tetrahedral mesh is specific to the deformation method used. To fully remove its potential influence on the estimation accuracy, the boundary element method (BEM) is a possible solution. BEM computes the stress directly on the surface mesh, thus the tetrahedral mesh is no longer needed. It also works better on the stress singularity problem often encountered in engineering analysis, although it has limitations when handling heterogeneous materials.

Finally, while our runtime stress analysis is fast, the precomputation still takes time. Accelerating the subspace construction for stress analysis is also a direction worthy of future exploration [60].

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their constructive comments. This work is partially supported by the NSF of China (No.61303136 and No.61272305), the National Science Foundation of U.S. (CAREER-1453101), the National Program for Special Support of Eminent Professionals of China, Lenovo's Program for Young Scientists, and generous gifts from Intel and Adobe. Kun Zhou is the corresponding author of this paper.

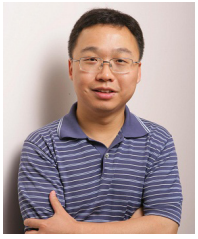
REFERENCES

[1] O. Stava, J. Vanek, B. Benes, N. Carr, and R. Měch, "Stress relief: improving structural strength of 3d printable objects," *ACM Trans. Graph.*, vol. 31, no. 4, p. 48, 2012.
[2] Q. Zhou, J. Panetta, and D. Zorin, "Worst-case structural analysis," *ACM Trans. Graph.*, vol. 32, no. 4, p. 137, 2013.
[3] W. Wang, T. Y. Wang, Z. Yang, L. Liu, X. Tong, W. Tong, J. Deng, F. Chen, and X. Liu, "Cost-effective printing of 3d objects with skin-frame structures," *ACM Trans. Graph.*, vol. 32, 2013.
[4] L. Lu, A. Sharf, H. Zhao, Y. Wei, Q. Fan, X. Chen, Y. Savoye, C. Tu, D. Cohen-Or, and B. Chen, "Build-to-last: Strength to weight 3d printed objects," *ACM Trans. Graph.*, vol. 33, 2014.
[5] Y. Xie, W. Xu, Y. Yang, X. Guo, and K. Zhou, "Agile structural analysis for fabrication-aware shape editing," *Computer Aided Geometric Design*, vol. 35, pp. 163–179, 2015.
[6] C. Multiphysics, "Comsol multiphysics user guide (version 4.3 a)," *COMSOL, AB*, 2012.
[7] J. Lewis, M. Cordner, and N. Fong, "Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation," in *SIGGRAPH '00*, 2000, pp. 165–172.
[8] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popović, "Mesh-based inverse kinematics," *ACM Trans. Graph.*, vol. 24, 2005.
[9] K. G. Der, R. W. Sumner, and J. Popović, "Inverse kinematics for reduced deformable models," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1174–1179, Jul. 2006.
[10] O. Weber, O. Sorkine, Y. Lipman, and C. Gotsman, "Context-aware skeletal shape deformation," *Computer Graphics Forum*, vol. 26, no. 3, pp. 265–274, 2007.
[11] X. Shi, K. Zhou, Y. Tong, M. Desbrun, H. Bao, and B. Guo, "Example-based dynamic skinning in real time," *ACM Trans. Graph.*, vol. 27, no. 3, p. 29, Aug. 2008.
[12] C. Von-Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt, "Real-time nonlinear shape interpolation," *ACM Trans. Graph.*, vol. 34, no. 3, p. 34, 2015.
[13] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," *ACM SIGGRAPH computer graphics*, vol. 20, no. 4, pp. 151–160, 1986.
[14] M. Botsch and L. Kobbelt, "An intuitive framework for real-time freeform modeling," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 630–634, 2004.
[15] D. Zorin, P. Schröder, and N. Fong, "Interactive multiresolution mesh editing," in *SIGGRAPH '97*, 1997, pp. 259–268.
[16] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel, "Interactive multi-resolution modeling on arbitrary meshes," in *SIGGRAPH '98*, 1998, pp. 105–114.
[17] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel, "Differential coordinates for interactive mesh editing," in *Shape Modeling International '04*, 2004, pp. 181–190.
[18] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *SGP'04*. New York, NY, USA: ACM, 2004, pp. 175–184.
[19] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum, "Mesh editing with poisson-based gradient field manipulation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 644–6351, 2004.
[20] M. Botsch and O. Sorkine, "On linear variational surface deformation methods," *IEEE Trans. Visualization and Computer Graphics*, vol. 14, no. 1, pp. 213–230, 2008.
[21] S. Martin, B. Thomaszewski, E. Grinspun, and M. Gross, "Example-based elastic materials," *ACM Trans. Graph.*, vol. 30, 2011.
[22] C. Schumacher, B. Thomaszewski, S. Coros, S. Martin, R. Sumner, and M. Gross, "Efficient simulation of example-based materials," in *Symp. on Computer Animation (SCA)*, 2012.
[23] D. L. James, C. D. Twigg, A. Cove, and R. Y. Wang, "Mesh ensemble motion graphs: Data-driven mesh animation with constraints," *ACM Trans. Graph.*, vol. 26, no. 4, Oct. 2007.
[24] C. Zheng, "One-to-many: Example-based mesh animation synthesis," in *Symp. on Computer Animation (SCA)*, 2013.
[25] A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis, "Efficient elasticity for character skinning with contact and collisions," *ACM Trans. Graph.*, vol. 30, no. 4, p. 37, 2011.
[26] S. S. An, T. Kim, and D. L. James, "Optimizing cubature for efficient integration of subspace deformations," *ACM Trans. Graph.*, vol. 27, no. 5, p. 165, Dec. 2008.
[27] Y. Teng, M. A. Otaduy, and T. Kim, "Simulating articulated subspace self-contact," *ACM Trans. Graph.*, vol. 33, no. 4, p. 106, 2014.

[28] A. Treuille, A. Lewis, and Z. Popović, "Model reduction for real-time fluids," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 826–834, Jul. 2006.
[29] M. Chai, C. Zheng, and K. Zhou, "A reduced model for interactive hairs," *ACM Trans. Graph.*, vol. 33, no. 4, 2014.
[30] S. Li, J. Huang, F. de Goes, X. Jin, H. Bao, and M. Desbrun, "Space-time editing of elastic motion through material optimization and reduction," *ACM Trans. Graph.*, vol. 33, no. 4, p. 108, 2014.
[31] C. Schulz, C. von Tycowicz, H.-P. Seidel, and K. Hildebrandt, "Animating deformable objects using sparse spacetime constraints," *ACM Trans. Graph.*, vol. 33, no. 4, p. 109, 2014.
[32] C. Zheng and D. L. James, "Rigid-body fracture sound with precomputed soundbanks," *ACM Trans. Graph.*, vol. 29, no. 3, 2010.
[33] H. Xu, Y. Li, Y. Chen, and J. Barbič, "Interactive material design using model reduction," *ACM Trans. Graph.*, vol. 34, no. 2, 2015.
[34] N. Umetani, T. Igarashi, and N. J. Mitra, "Guided exploration of physically valid shapes for furniture design." *ACM Trans. Graph.*, vol. 31, no. 4, p. 86, 2012.
[35] M. Skouras, B. Thomaszewski, S. Coros, B. Bickel, and M. Gross, "Computational design of actuated deformable characters," *ACM Trans. Graph.*, vol. 32, no. 4, Jul. 2013.
[36] X. Chen, C. Zheng, W. Xu, and K. Zhou, "An asymptotic numerical method for inverse elastic shape design," *ACM Trans. Graph.*, vol. 33, no. 4, 2014.
[37] A. Francavilla, C. Ramakrishnan, and O. Zienkiewicz, "Optimization of shape to minimize stress concentration," *The Journal of Strain Analysis for Engineering Design*, vol. 10, no. 2, pp. 63–70, 1975.
[38] P. Pedersen, "On optimal shapes in materials and structures," *Structural and Multidisciplinary Optimization*, vol. 19, no. 3, pp. 169–182, 2000.
[39] W. A. Wall, M. A. Frenzel, and C. Cyron, "Isogeometric structural shape optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 33, pp. 2976C–2988, 2008.
[40] N. Umetani and R. Schmidt, "Cross-sectional structural analysis for 3d printing optimization," in *SIGGRAPH Asia 2013 Technical Briefs*, ser. SA '13, 2013.
[41] J. Barbič and D. L. James, "Real-time subspace integration for st. venant-kirchhoff deformable models," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 982–990, 2005.
[42] K. Hildebrandt, C. Schulz, C. V. Tycowicz, and K. Polthier, "Interactive surface modeling using modal analysis," *ACM Trans. Graph.*, vol. 30, no. 5, p. 119, 2011.
[43] L. Kharevych, P. Mullen, H. Owhadi, and M. Desbrun, "Numerical coarsening of inhomogeneous elastic materials," *ACM Trans. Graph.*, vol. 28, no. 3, p. 51, 2009.
[44] M. Nesme, P. G. Kry, L. Jeřábková, and F. Faure, "Preserving topology and elasticity for embedded deformable models," *ACM Trans. Graph.*, vol. 28, no. 3, p. 52, 2009.
[45] D. Chen, D. I. Levin, S. Sueda, and W. Matusik, "Data-driven finite elements for geometry and material design," *ACM Trans. Graph.*, vol. 34, no. 4, p. 74, 2015.
[46] M. E. Gurtin, *An introduction to continuum mechanics*. Academic press, 1982.
[47] R. v. Mises, "Mechanik der festen körper im plastisch-deformablen zustand," *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, vol. 1913, pp. 582–592, 1913.
[48] J. Gere and B. Goodno, *Mechanics of materials*. Nelson Education, 2012.
[49] O. Zienkiewicz, R. Taylor, and J. Zhu, *The finite element method: its basis and fundamentals*. Butterworth-Heinemann, 2005.
[50] J. Bonet, *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press, 1997.
[51] J. N. Reddy, *An introduction to continuum mechanics*. Cambridge University Press, 2013.
[52] C. von Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt, "An efficient construction of reduced deformable objects," *ACM Trans. Graph.*, vol. 32, no. 6, p. 213, 2013.
[53] A.-L. Tertois, T. Frank, and J.-L. Mallet, "Real-time tetrahedral volume editing accounting for discontinuities," in *IEEE Computer Aided Design and Computer Graphics*, 2005.
[54] E. D. Andersen and K. D. Andersen, "The mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm," in *High performance optimization*. Springer, 2000, pp. 197–232.
[55] OpenMP, "OpenMP application program interface version 3.0," May 2008. [Online]. Available: http://www.openmp.org/mp-documents/spec30.pdf
[56] A. Jacobson, D. Panozzo *et al.*, "libigl: A simple C++ geometry processing library," 2015. [Online]. Available: http://libigl.github.io/libigl/
[57] A. Jacobson, I. Baran, J. Popovic, and O. Sorkine, "Bounded biharmonic weights for real-time deformation." *ACM Trans. Graph.*, vol. 30, no. 4, p. 78, 2011.
[58] W. Schütz, "A history of fatigue," *Engineering fracture mechanics*, vol. 54, no. 2, pp. 263–300, 1996.
[59] R. Gal, O. Sorkine, N. J. Mitra, and D. Cohen-Or, "iwires: An analyze-and-edit approach to shape manipulation," *ACM Trans. Graph.*, vol. 28, no. 3, p. 33, 2009.
[60] Y. Yang, D. Li, W. Xu, Y. Tian, and C. Zheng, "Expediting precomputation for reduced deformable simulation," *ACM Trans. Graph.*, vol. 34, no. 6, p. 243, 2015.

Xiang Chen is an Assistant Professor in the State Key Lab of CAD&CG, Zhejiang University. He received His Ph.D. in Computer Science from Zhejiang University in 2012. His current research interests mainly include fabrication-aware design, image analysis/editing, shape modeling/retrieval and computer-aided design.

Changxi Zheng is an Assistant Professor in the Computer Science Department at Columbia University. Prior to joining Columbia, he received his M.S. and Ph.D. from Cornell University and his B.S. from Shanghai Jiaotong University. His research spans computer graphics, physically-based simulation, computational design, computational acoustics, scientific computing and robotics. He has been serving as an Associated Editor of ACM Transactions on Graphics, and won the NSF CAREER Award and the Cornell CS Best Dissertation award in 2012.



Kun Zhou is a Cheung Kong Professor in the Computer Science Department of Zhejiang University, and the Director of the State Key Lab of CAD&CG. Prior to joining Zhejiang University in 2008, Dr. Zhou was a Leader Researcher of the Internet Graphics Group at Microsoft Research Asia. He received his B.S. degree and Ph.D. degree in computer science from Zhejiang University in 1997 and 2002, respectively. His research interests are in visual computing, parallel computing, human computer interaction, and virtual reality. He currently serves on the editorial/advisory boards of ACM Transactions on Graphics and IEEE Spectrum. He is a Fellow of IEEE.

Example-based Subspace Stress Analysis for Interactive Shape Design

SUPPLEMENTAL MATERIAL

Xiang Chen* Changxi Zheng† Kun Zhou*

* Zhejiang University † Columbia University

1 Linear Tetrahedral Element

To aid explanation, we reproduce the discretized elastostatic stress equation from the main paper

$$\sum_e \int_{V_e} \sigma_{ij} B_{k,j}\, dV = \sum_e \left(\int_{V_e} b_i B_k\, dV + \int_{\partial V_e} t_i B_k\, dS \right). \tag{1}$$

Each linear element has 4 nodes, one for each vertex of the tetrahedron (solid points in Figure 1). The 4 nodal basis functions B_k are linear. First, the RHS of Equation (1) is transformed by using the change of variables formula

$$\int_{V_e} b_i B_k\, dV = \iiint b_i B_k \det \boldsymbol{J}_e\, d\xi_1\, d\xi_2\, d\xi_3,$$

where $\boldsymbol{\xi}$ are local coordinates and $\boldsymbol{J}_e = \nabla_{\boldsymbol{\xi}} \boldsymbol{X}$ is the Jacobian matrix. Here b_i and $\det \boldsymbol{J}_e$ are constant with respect to $\boldsymbol{\xi}$, so only B_k is integrated (refer to the supplemental material for the details). We get the same result $\frac{1}{4} b_i V_e$, or in its matrix form $\frac{1}{4} \boldsymbol{b} V_e$ for every k, where V_e is the volume of element e. In other words, the whole body force of element e is equally lumped into its 4 nodes. For compactness, the traction term is omitted.

Similarly, the LHS of Equation (1) is transformed into

$$\int_{V_e} \sigma_{ij} B_{k,j}\, dV = \iiint \sigma_{ij} B_{k,j} \det \boldsymbol{J}_e\, d\xi_1\, d\xi_2\, d\xi_3,$$

where both σ_{ij} and $B_{k,j}$ are constant. The result is $\sigma_{ij}(\bar{n}_j)_k$, or in its matrix form $\boldsymbol{\sigma} \bar{\boldsymbol{n}}_k$, where $\bar{\boldsymbol{n}}_k$ is the outward normal of node k (see Section 4.1).

Given the above integration results, we have the element equation $\boldsymbol{\sigma}^e \bar{\boldsymbol{n}}_k^e = \frac{1}{4} \boldsymbol{b} V_e$ for each element e. We then assemble all the element equations to get the per-node static equilibrium equation

$$\boldsymbol{f}^i = \sum_{e \in \mathsf{adj}(i)} \boldsymbol{\sigma}^e \bar{\boldsymbol{n}}_i^e = \frac{1}{4} \boldsymbol{b} \sum_{e \in \mathsf{adj}(i)} V_e = \boldsymbol{g}^i,$$

where $e \in \mathsf{adj}(i)$ indicates an element incident to node i. By stacking the above equation for each node, we get the Equation (5) in paper, where A_σ represents the stacked sparse matrix from $\bar{\boldsymbol{n}}_i^e$, and $\boldsymbol{b}_\sigma$ is the stacked vector of $\boldsymbol{g}^i$.

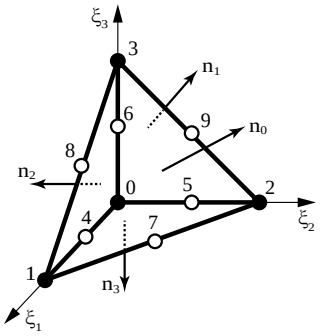


Figure 1: The finite tetrahedral element: linear (4 nodes) and quadratic (10 nodes).

2 Quadratic Tetrahedral Element

For quadratic elements, besides the 4 nodes located at mesh vertices, there are 6 more nodes positioned at the midpoints of the edges (see figure above). The 10 basis functions for these nodes are quadratic.

The body force term on the RHS of Equation (1) is integrated analytically as $c_k b_i V_e$, where $c_k = -\frac{1}{20}$ for $k = 1,2,3,4$, and $c_k = \frac{1}{5}$ for $k = 5,6,\ldots,10$. Refer to the supplemental material for a detailed derivation.

For the LHS of Equation (1), both σ_{ij} and $B_{k,j}$ change linearly in each element, leading to a complicated analytical integration. A cheaper way is using numerical tetrahedral integration. Hence we choose 4 quadrature points, with local coordinates $\boldsymbol{\xi}^q$ $(q = 1,2,3,4)$, inside each element, which is provably accurate enough for integration of quadratic functions. In matrix form, the integration is computed as (details in Section 4.2)

$$\int_{V_e} \boldsymbol{\sigma}\,(\nabla_{\boldsymbol{X}} B_m)\,dV \approx \sum_{q=1}^{4} w_q \boldsymbol{\sigma}(\boldsymbol{\xi}^q)\boldsymbol{T}\left(\nabla_{\boldsymbol{\xi}} B_m(\boldsymbol{\xi}^q)\right),$$

where w_q is the integration weight for quadrature point $\boldsymbol{\xi}^q$, $\boldsymbol{T} = \frac{-[a_1\boldsymbol{n}_1\ a_2\boldsymbol{n}_2\ a_3\boldsymbol{n}_3]}{3}$ is a constant matrix for each element, a_i and $\boldsymbol{n}_i$ (see the inset figure) are the area and normal vector of the triangular face opposite to vertex i in the tetrahedron. Similar to the per-node equation stacking in Section 1, we get the Equation (5) in paper for quadratic elements.

3 Construction of E^t

Similar to Equation (1) in paper, the strain ϵ_{mn} is linearly related to σ_{ij}:

$$\epsilon_{ij} = \mathsf{D}_{ijmn}\sigma_{mn},$$

where D is a 4th-order tensor whose coefficients depend on material parameters. Given the stress subspace S, we have $\boldsymbol{s}_k^t$, which is the stress field of tetrahedron t corresponding to the kth basis vector in S. Then we can write the reduced stress field $\sigma_{mn}^t = (s_k^t)_{mn} r_k$ according to the reduced coordinates $\boldsymbol{r}$. Therefore, by using the above linear relation between ϵ_{mn} and σ_{ij}, we can write the reduced strain field of tetrahedron t as

$$\epsilon_{ij}^t = \mathsf{D}_{ijmn}\sigma_{mn}^t = \mathsf{D}_{ijmn}(s_k^t)_{mn} r_k.$$

Here we define $E_{ijk}^t = \mathsf{D}_{ijmn}(s_k^t)_{mn}$, which is exactly the 3rd-order tensor E^t. Note that only linear operations are involved.

4 Integration of FEM Equation (1)

4.1 Linear basis functions

In case of linear tetrahedral element (4-nodes), we have the basis functions

$$B_0 = \lambda,\ B_1 = \xi_1,\ B_2 = \xi_2,\ B_3 = \xi_3,$$

where $\lambda = 1 - \xi_1 - \xi_2 - \xi_3$. Thus the jacobian matrix $\boldsymbol{J}_e$ is

$$\boldsymbol{J}_e = \nabla_{\boldsymbol{\xi}}\boldsymbol{X} = \sum_{k=0}^{3} \boldsymbol{X}_k \otimes \nabla_{\boldsymbol{\xi}} B_k = [\boldsymbol{X}_1 - \boldsymbol{X}_0\ \boldsymbol{X}_2 - \boldsymbol{X}_0\ \boldsymbol{X}_3 - \boldsymbol{X}_0],$$

and $\det \boldsymbol{J}_e = 6V_e$, where V_e is the volume of tetrahedron e.

Taking B_0 as an integration example

$$\begin{aligned}
&\iiint b_i B_0 \det \boldsymbol{J}_e \, d\xi_1 \, d\xi_2 \, d\xi_3 \\
&= 6 b_i V_e \int_0^1 \int_0^{1-\xi_1} \int_0^{1-\xi_1-\xi_2} (1-\xi_1-\xi_2-\xi_3) \, d\xi_1 \, d\xi_2 \, d\xi_3 \\
&= \frac{1}{4} b_i V_e,
\end{aligned}$$

while integration of B_1, B_2 and B_3 are similar.

For the LHS of Equation (1), we use matrix form here for derivation. In linear case, both $\boldsymbol{\sigma}$ and $\nabla_X B_k$ are constant and thus independent of $\boldsymbol{\xi}$. For instance

$$\begin{aligned}
\nabla_X B_0 &= \boldsymbol{J}_e^{-T} \nabla_\xi B_0 \\
&= \boldsymbol{J}_e^{-T} [-1 \; -1 \; -1]^T \\
&= \frac{1}{\det \boldsymbol{J}_e} [\boldsymbol{c}_1 \, \boldsymbol{c}_2 \, \boldsymbol{c}_3] [-1 \; -1 \; -1]^T
\end{aligned}$$

where $\boldsymbol{c}_1 = (\boldsymbol{X}_2 - \boldsymbol{X}_0) \times (\boldsymbol{X}_3 - \boldsymbol{X}_0) = -2a_1 \boldsymbol{n}_1$, $\boldsymbol{c}_2 = -2a_2 \boldsymbol{n}_2$, $\boldsymbol{c}_3 = -2a_3 \boldsymbol{n}_3$, a_i and $\boldsymbol{n}_i$ are the area and outward normal of the triangle opposite to node i in the tetrahedron. Based on the above derivation, and define $\bar{\boldsymbol{n}}_k = \frac{1}{3} \sum_{i \neq k}^4 a_i \boldsymbol{n}_i$ (i.e., the outward normal of node k), we have $\nabla_X B_k = \frac{6 \bar{\boldsymbol{n}}_k}{\det \boldsymbol{J}_e}$. Thus the integration is

$$\iiint \boldsymbol{\sigma} \, (\nabla_X B_k) \det \boldsymbol{J}_e \, d\xi_1 \, d\xi_2 \, d\xi_3 = \boldsymbol{\sigma} \bar{\boldsymbol{n}}_k.$$

4.2 Quadratic basis functions

For quadratic tetrahedral element (10-nodes, 4 on vertices and 6 on edges), we have the following basis functions

$$\begin{aligned}
&B_0 = \lambda(2\lambda - 1), \; B_1 = \xi_1(2\xi_1 - 1), \\
&B_2 = \xi_2(2\xi_2 - 1), \; B_3 = \xi_3(2\xi_3 - 1), \\
&B_4 = 4\xi_1\lambda, \; B_5 = 4\xi_2\lambda, \; B_6 = 4\xi_3\lambda, \\
&B_7 = 4\xi_1\xi_2, \; B_8 = 4\xi_1\xi_3, \; B_9 = 4\xi_2\xi_3.
\end{aligned}$$

When the 6 nodes are put right at the midpoints of edges, the jacobian matrix is $\boldsymbol{J}_e = \sum_{k=0}^9 \boldsymbol{X}_k \otimes \nabla_\xi B_k = [\boldsymbol{X}_1 - \boldsymbol{X}_0 \; \boldsymbol{X}_2 - \boldsymbol{X}_0 \; \boldsymbol{X}_3 - \boldsymbol{X}_0]$, exactly the same as it is in the case of linear basis functions.

Again, taking B_0 as an integration example

$$\begin{aligned}
&\iiint b_i B_0 \det \boldsymbol{J}_e \, d\xi_1 \, d\xi_2 \, d\xi_3 \\
&= 6 b_i V_e \int_0^1 \int_0^{1-\xi_1} \int_0^{1-\xi_1-\xi_2} \lambda(2\lambda - 1) \, d\xi_1 \, d\xi_2 \, d\xi_3 \\
&= -\frac{1}{20} b_i V_e,
\end{aligned}$$

and $B_1, B_2, \ldots, B_9$ are handled similarly.

For the LHS of Equation (1), we have

$$\nabla_X B_k = \boldsymbol{J}_e^{-T} \nabla_\xi B_k = \frac{1}{\det \boldsymbol{J}_e} [\boldsymbol{c}_1 \, \boldsymbol{c}_2 \, \boldsymbol{c}_3] \nabla_\xi B_k.$$

Since $\boldsymbol{\sigma}$ and $\nabla_{\xi} B_k$ changes with $\boldsymbol{\xi}$, we use numerical quadrature to compute the integral

$$
\begin{aligned}
&\int_{V_e} \boldsymbol{\sigma}\,(\nabla_{\boldsymbol{X}} B_m)\,dV \\
&\approx \sum_{q=1}^{n_q} w_q * \left(\boldsymbol{\sigma}(\boldsymbol{\xi}_q) \frac{1}{\det \boldsymbol{J}_e} [\boldsymbol{c}_1\, \boldsymbol{c}_2\, \boldsymbol{c}_3]\, \nabla_{\xi} B_k(\boldsymbol{\xi}_q) \right) * V_e \\
&= \sum_{q=1}^{n_q} w_q * \left(\boldsymbol{\sigma}(\boldsymbol{\xi}_q) \frac{[-2a_1\boldsymbol{n}_1\ -2a_2\boldsymbol{n}_2\ -2a_3\boldsymbol{n}_3]}{6V_e}\, \nabla_{\xi} B_k(\boldsymbol{\xi}_q) \right) * V_e \\
&= \sum_{q=1}^{n_q} w_q * \left(\boldsymbol{\sigma}(\boldsymbol{\xi}_q) \frac{[-a_1\boldsymbol{n}_1\ -a_2\boldsymbol{n}_2\ -a_3\boldsymbol{n}_3]}{3}\, \nabla_{\xi} B_k(\boldsymbol{\xi}_q) \right),
\end{aligned}
$$

where n_q is the number of quadrature points.