Analytic Eigensystems for Isotropic Distortion Energies

BREANNAN SMITH, FERNANDO DE GOES, and THEODORE KIM, Pixar Animation Studios

Fig. 1. We present closed-form expressions for the eigensystems of isotropic distortion energies suited to geometry processing and physical simulation. We use the analytic eigensystem of the Symmetric Dirichlet energy to optimize a 2D parameterization (left). Our analytic eigensystem for the ARAP energy was used to simulate the over-inflated tires in this scene from Cars 3 (right). ©Disney/Pixar

Many strategies exist for optimizing non-linear distortion energies in geometry and physics applications, but devising an approach that achieves the convergence promised by Newton-type methods remains challenging. In order to guarantee the positive semi-definiteness required by these methods, a numerical eigendecomposition or approximate regularization is usually needed. In this paper, we present analytic expressions for the eigensystems at each quadrature point of a wide range of isotropic distortion energies. These systems can then be used to project energy Hessians to positive semi-definiteness analytically. Unlike previous attempts, our formulation provides compact expressions that are valid both in 2D and 3D, and does not introduce spurious degeneracies. At its core, our approach utilizes the invariants of the stretch tensor that arises from the polar decomposition of the deformation gradient. We provide closed-form expressions for the eigensystems for all these invariants, and use them to systematically derive the eigensystems of any isotropic energy. Our results are suitable for geometry optimization over flat surfaces or volumes, and agnostic to both the choice of discretization and basis function. To demonstrate the efficiency of our approach, we include comparisons against existing methods on common graphics tasks such as surface parameterization and volume deformation.

CCS Concepts: • Computing methodologies → Mesh geometry models:

Additional Key Words and Phrases: second-order methods, geometry optimization, distortion energy.

ACM Reference Format:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Association for Computing Machinery.
0730-0301/2018/8-ART $15.00
https://doi.org/10.1145/nmnnmn.nmnnnn

1 INTRODUCTION
The minimization of distortion energies is at the core of many graphics applications, such as surface parameterization and physically-based deformation. Unfortunately, most energies are non-convex, which make them challenging to optimize numerically. In particular, the classic second-order Newton algorithm (see, e.g., [Nocedal and Wright 2006]) tends to stall and even diverge in the presence of an indefinite energy Hessian. To overcome this issue, many strategies have been developed for modifying the Hessian matrix using a variety of convex proxies. For instance, some techniques replace the Hessian with a Laplacian-like preconditioning matrix at the cost of degrading convergence to first-order [Claici et al. 2017; Kovalsky et al. 2016]. Other methods alter the Hessian by removing negative eigenvalues numerically [Stomakhin et al. 2012; Teran et al. 2005] or via 2D composite majorization [Shtengel et al. 2017].

In this paper, we resolve Hessian indefiniteness by investigating eigenstructures analytically. A similar approach was considered previously by Chen and Weber [2017] for the restricted case of 2D isotropic energies, but their formulation is based on complex analysis and thus does not generalize to 3D. Conversely, some attempts have sacrificed generality to derive fast Hessian projections for specific energies. In particular, McAdams et al. [2011] obtained analytic eigenvalues for Co-rotational elasticity, while Smith et al. [2018] addressed a custom Neo-Hookean energy. Our method automatically produces these results as special cases, but goes further and establishes a general method for revealing compact eigensystems for a variety of isotropic distortion energies in both 2D and 3D.

Our key contribution is a list of analytic expressions that can be used to efficiently project the Hessian of isotropic distortion energies to positive semi-definiteness. These expressions can be easily incorporated into a projected Newton solver, thus providing second-order convergence with existing distortion optimization techniques. To achieve this, we depart from previous methods by addressing isotropic energies using the invariants of the stretch tensor that arises from the polar decomposition of the deformation gradient. We present closed-form expressions for the eigenvalues and eigenvectors of all these tensor invariants, and use these results to arrive at a systematic scheme for constructing the eigenstructure...
of the Hessians for any isotropic energy. Our approach leads to analytical expressions for \( \frac{1}{3} \) of the eigenpairs of any isotropic distortion energy in 3D, and for \( \frac{1}{2} \) of the eigenpairs in 2D. We also show that these closed-form eigenvectors are invariant to the distortion model both in 2D and 3D. The remaining eigenvalues and eigenvectors can often be found analytically or, in the worst case, by solving a small \( d \times d \) eigenproblem \((d = 2 \text{ or } 3)\). We demonstrate the effectiveness of our method by providing fully analytic expressions for the eigensystems of many popular distortion energies, including ARAP, Symmetric ARAP, Co-rotational, Symmetric Dirichlet, and MIPS.

2 RELATED WORK

Many graphics applications are based on the optimization of isotropic distortion energies, so for brevity we will focus on previous methods that have addressed the numerics of these optimizations.

2.1 First-Order Methods

The local-global algorithm is arguably the most popular optimization technique in geometry processing. It consists of an expectation-maximization scheme that alternates local and global steps. Alexa et al. [2000], for instance, employed this method to minimize the As-Rigid-As-Possible (ARAP) energy in 2D. The local step projects each deformation gradient to a rotation, while the global step uses a Laplacian to fuse the rotated elements using a linear solve. This approach has been applied to many applications, such as surface modeling [Sorkine and Alexa 2007], mesh parameterization [Liu et al. 2008], and volume deformation [Jacobson et al. 2012].

Bouaziz et al. [2012] extended the local-global method to a broad class of distortion energies using proximal operators. The Laplacian in the global step was replaced by the Jacobian matrix of the proximal operators, which resembles the Hessian approximation of a Gauss-Newton method, while element-wise local steps were retained. This technique was later employed in physical simulation [Bouaziz et al. 2014] and improved using ADMM [Narain et al. 2016], Chebyshev extrapolation [Wang 2015], and L-BFGS updates [Liu et al. 2017].

Other distortion energies were addressed using block coordinate descent methods [Fu et al. 2015; Hornan and Greiner 1999] and Gauss-Newton solvers [Eigensatz and Pauly 2009]. More recently, Kovalsky et al. [2016] showed that the Laplacian matrix is a reliable quadratic proxy for many distortion energies, while Cläi et al. [2017] advocated the discrete Killing operator as an isometry-aware proxy. Alternatively, Rabinovich et al. [2017] proposed an iterative reweighting scheme of the Laplacian based on gradient residuals. Pighin and Lewis [2007] instead used an iterative reweighted least-squares technique, while Bommes et al. [2009] adopted a greedy progressive stiffening of the Laplacian.

All of these techniques are solely contingent on the gradient of the distortion energy, and are therefore inherently first-order. In contrast, our work investigates the use of second-order methods.

2.2 Second-Order Methods

Second-order methods are preferred in numerical optimization due to their superior convergence. However, special care must be taken in order to handle indefinite Hessians, otherwise the computations can stall or diverge. For example, Chao et al. [2010] used a trust-region solver for the optimization of ARAP and Co-rotational energies, which is essentially an adaptive regularization of the Hessian. Shitengel et al. [2017] advocated a convex-concave decomposition of the distortion energies combined with a convex majorizer solver. However, finding an effective convex-concave decomposition is an opaque process, and the approach of Shitengel et al. [2017] is limited to 2D. We instead present a systematic, second-order approach for distortion energies in both 2D and 3D.

An alternative is to use a projected Newton technique, which projects the Hessian to a positive semi-definite matrix by clamping its negative eigenvalues at every solver iteration. Since extracting the eigenvalues and eigenvectors of the global Hessian is computationally prohibitive, several authors have projected the eigensystem of a local Hessian evaluated at quadrature points (see, e.g., [Fu and Liu 2016]). Consequently, the eigendecomposition of each element-wise matrix becomes the computational bottleneck. Our results accelerate this stage by presenting closed-form expressions for the eigenvalues and eigenvectors.

By viewing the distortion energies in terms of the invariants of the 3D Cauchy-Green strain tensor, the work of Teran et al. [2005] was able to probe the indefiniteness of the Hessian using one \( 3 \times 3 \) and three \( 2 \times 2 \) eigenproblems. While this approach reduces the problem size, it does not yield the closed-form expressions for the underlying eigenvalues and eigenvectors, so the construction of the projected Hessian is performed numerically. The use of tensor invariants with respect to the Cauchy-Green strain tensor is also limiting, because it is insufficient to express stretch-based energies such as ARAP or the Co-rotational model. This limitation was addressed in Stomakhin et al. [2012] by replacing the Cauchy-Green invariants with the singular values of the deformation gradient, but the inefficient numerical eigensolves for Hessian blocks remain. Xu et al. [2015] improved these computations further by restricting the analysis to energies that satisfy the Valanis-Landel hypothesis. Unfortunately, many geometric energies, such as MIPS, Symmetric Dirichlet, and Symmetric ARAP do not fall into this category. Our results instead lead to faster, more compact, and more general code.

The work of McAdams et al. [2011] presented an analytic solution for the indefiniteness of the Co-rotational energy. To do so, they decomposed the 4th order tensor defined by the energy Hessian into symmetric and skew-symmetric parts. This yields the eigenstructure of the Co-rotational model, albeit embedded inside 4th order tensors. We propose a more general derivation that produces their eigenvalue expressions as a special case, and additionally reveals the structure of the underlying eigenvectors.

Chen and Weber [2017] considered analytic eigensystems for 2D isotropic energies. However, their derivation relies heavily on complex derivatives and thus does not extend to 3D. Their generic expressions also contain degeneracies that invalidate the eigenanalysis for a variety of energies and deformation configurations. For example, at uniform scaling configurations such as the rest state, their formulas return a null vector as an eigenvector (see §1.3 in the supplemental material). In sharp contrast, our formulation is well-defined for any deformation gradient configuration and for any isotropic energy.
More recently, Smith et al. [2018] took an energy-specific approach and resolved the indefiniteness of a custom Neo-Hookean model. In doing so, they derived the analytic eigensystems for the Frobenius squared norm and the determinant of the deformation gradient in 3D. However, their approach lacks generality as they did not derive the eigensystem of a third invariant (i.e., in §4.1) that is not present in the Neo-Hookean model. This additional term appears in almost all geometric distortion energies, and precludes the use of their results in this context. In contrast, our formulation handles this term in closed-form. By addressing this invariant, we also provide the eigenstructure for the rotation gradient, a term that has been known to introduce numerical difficulties in the past [Twigg and Kačić-Alesić 2010], and is often computed by performing a set of matrix inverses [Barbić and Zhao 2011]. Our analysis reveals that there is in fact a simple, direct expression for this term that bypasses the eigenstructure for the rotation gradient and resolves the indefiniteness of a custom Neo-Hookean approach and resolved the indefiniteness of a custom Neo-Hookean model. In doing so, they derived the analytic eigensystems for the Frobenius squared norm and the determinant of the deformation gradient in 3D. However, their approach lacks generality as they did not derive the eigensystem of a third invariant (i.e., in §4.1) that is not present in the Neo-Hookean model. This additional term appears in almost all geometric distortion energies, and precludes the use of their results in this context. In contrast, our formulation handles this term in closed-form. By addressing this invariant, we also provide the eigenstructure for the rotation gradient, a term that has been known to introduce numerical difficulties in the past [Twigg and Kačić-Alesić 2010], and is often computed by performing a set of matrix inverses [Barbić and Zhao 2011]. Our analysis reveals that there is in fact a simple, direct expression for this term that bypasses the eigenstructure for the rotation gradient and resolves the indefiniteness of a custom Neo-Hookean model.

We begin by establishing several basic definitions and concepts. Scalars will be denoted with unbolded lowercase (a), vectors as bolded lowercase (a), matrices (a.k.a. 2nd order tensors) as bolded uppercase (A), and 4th order tensors using blackboard bold (A).

Tensor Notation: We will be analyzing many 2nd and 4th order tensors, so we adopt a notation similar to Golub and Van Loan [2012] that flattens 2nd and 4th order tensors to vectors and matrices respectively. Concretely, given a \(2 \times 2\) matrix \(A\), we define the vectorization operator "vec" as the column-wise flattening:

\[
A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \Leftrightarrow \quad \text{vec}(A) = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = a.
\] 

(1)

We can arrange a 4th order tensor as a matrix of matrices:

\[
\mathbf{A} = \begin{bmatrix} a & c & i & k \\ b & d & j & l \\ e & g & m & o \\ f & h & n & p \end{bmatrix} = \begin{bmatrix} [A_{11}] & [A_{12}] \\ [A_{21}] & [A_{22}] \end{bmatrix},
\] 

(2)

and define its vectorization operator "vec" as a double unfolding that yields the matrix:

\[
\text{vec}(\mathbf{A}) = \begin{bmatrix} \text{vec}(A_{11}) & \text{vec}(A_{12}) & \text{vec}(A_{21}) & \text{vec}(A_{22}) \end{bmatrix} = \begin{bmatrix} a & e & i & m \\ b & f & j & n \\ c & g & k & o \\ d & h & l & p \end{bmatrix} = A.
\] 

(3)

This flattening convention has several properties that we leverage in our derivations. The double contraction of two 2nd order tensors becomes a dot product, i.e. \(A \cdot B = \text{tr}(A^T B) = a^T b\), and the eigenvector of a flattened tensor \(B a = \lambda a\) is equivalent to the "eigenvector" of the corresponding higher-order tensor, i.e., \(B \in \mathbb{R}^n \Lambda\).

Discretization: We consider the distortion optimization of a d-dimensional mesh \(M\), where \(d = 2\) or 3. We use \(\mathbf{x}\) to denote the vertex positions of \(M\) in the rest pose, and \(x\) to indicate the new positions computed by the optimization. These positions can be interpolated using any choice of basis functions associated with quadrature points \(\{q\}\) distributed over \(M\). For instance, the typical case of simplicial meshes sets the quadrature point to the center of simplices (i.e., triangles in 2D and tetrahedra in 3D) with piecewise linear bases. The number of degrees of freedom in a mesh is denoted \(n\) and it is equal to the dimension \(d\) of the mesh embedding multiplied by the the number of vertices in \(M\).

Deformation Gradient: The distortion of \(M\) from a rest configuration \(\mathbf{x}\) to a posed configuration \(\mathbf{x}\) can be quantified by computing the deformation gradient \(\mathbf{F}\) at each quadrature point \(q\). The deformation gradient at \(q\) indicates the linear term of the affine transformation that maps \(x_q = F_q x_q + y\), where \(y\) is a translation vector. Note that \(F_q\) depends on the basis functions used to interpolate \(\mathbf{x}\) at the quadrature point \(q\). We denote the SVD of the deformation gradient as \(F = U \Sigma V^T\), where \(U, V \in \mathbb{SO}(d)\) and \(\Sigma\) is a diagonal matrix containing the principal stretches \(\{\sigma_1, \ldots, \sigma_d\}\). The principal stretches may have negative values in order to accommodate reflections (see, e.g., [Irving et al. 2004; Twigg and Kačić-Alesić 2010]). This SVD can be used to define the polar decomposition \(F = RS\), where \(R\) is a rotation and \(S = V \Sigma V^T\) is a (symmetric) stretch tensor.

Distortion Energy: The distortion quantified by the deformation gradient at each quadrature point can be aggregated into a scalar distortion energy of the form:

\[
\Psi(\mathbf{x}) = \sum_q \Psi_q(\mathbf{F}_q)|q|,
\] 

(4)

where \(\Psi(\mathbf{x})\) is the energy over the entire mesh, \(\Psi_q\) is the energy at a quadrature point, and \(|q|\) is a volume weight associated with quadrature point \(q\) at rest. We can further apply the chain rule to express the first and second derivatives of \(\Psi\) in terms of \(\mathbf{F} = \text{vec}(\mathbf{F})\):

\[
\frac{\partial \Psi}{\partial \mathbf{x}} = \sum_q |q| \frac{\partial \mathbf{F}_q}{\partial \mathbf{x}} \frac{\partial \Psi_q}{\partial \mathbf{F}_q}
\] 

(5a)

\[
\frac{\partial^2 \Psi}{\partial \mathbf{x}^2} = \sum_q |q| \frac{\partial^2 \mathbf{F}_q}{\partial \mathbf{x}^2} \frac{\partial \Psi_q}{\partial \mathbf{F}_q} + \frac{\partial \Psi_q}{\partial \mathbf{x}} \frac{\partial^2 \mathbf{F}_q}{\partial \mathbf{x}^2}
\] 

(5b)

Here, \(\frac{\partial \mathbf{F}_q}{\partial \mathbf{x}}\) is a \(d^2 \times n\) matrix, \(\frac{\partial \Psi_q}{\partial \mathbf{F}_q}\) is a vector of size \(d^2\), and \(\frac{\partial^2 \Psi_q}{\partial \mathbf{F}_q^2}\) is a \(d^2 \times d^2\) matrix, where \(n\) is the number of degrees of freedom in \(M\) and \(d\) is the embedding dimension. While this change of variables is not a similarity transform and can modify the spectrum, the positive-semi definiteness of \(\frac{\partial^2 \Psi_q}{\partial \mathbf{F}_q^2}\) guarantees the positive-semi definiteness of the resulting product in Eq. (5b). Therefore, we can construct a numerical solver that is agnostic to the discretization by analyzing the derivatives with respect to \(\mathbf{F}\) (see, e.g., [Sifakis and Barbic 2012]). We will henceforth refer to direct modifications of \(\frac{\partial^2 \Psi}{\partial \mathbf{x}^2}\) as \(x\)-based, and direct modifications of \(\frac{\partial^2 \Psi}{\partial \mathbf{F}_q^2}\) as \(F\)-based.
We now present our core contribution. First, we demonstrate that we use the eigensystems of the invariants to arrive at a procedure for deriving the analytic eigensystems of isotropic distortion energies in both 2D and 3D. Since our analysis is performed per quadrature point, we drop the subscript $q$ for conciseness.

### 4 OUR APPROACH

We now present our core contribution. First, we demonstrate that the use of scalar invariants based on $S$, the stretch tensor of $F$, is the correct perspective for the analysis. Then, we show that the eigensystem of each invariant can be stated in closed-form. Finally, we use the eigensystems of the invariants to arrive at a procedure for deriving the linear distortion terms that appear in many popular models, such as the Co-rotational energy. Instead, these methods expanded distortion energies in terms of principal stretches, at the cost of verbose expressions and slower computations. Alternatively, Smith et al. [2018] used the 3D versions of the $I_2$ and $I_3$ we describe here, but their analysis is incomplete because it misses the existence of $I_1$ entirely. Shtengel et al. [2017] and Chen and Weber [2017] proposed to use the similarity and anti-similarity parts of $F$, but this decomposition is valid only in 2D.

By using the $S$ invariants, we are able to perform a complete eigenanalysis that has none of the shortcomings of these previous approaches. This substitution can be employed without loss of generality, because we show in Appendix A that the 3D Cauchy-Green invariants can be written in terms of our invariants. The characteristic polynomial of $F$ can also be expressed with our invariants via:

\[
\begin{align*}
I_1 &= \text{tr}(S) = \sum_i \sigma_i, \\
I_2 &= \|S\|^2 = \sum_i \sigma_i^2, \\
I_3 &= \det(S) = \prod_i \sigma_i.
\end{align*}
\]

The first invariant $I_1$ sums one-dimensional stretches, and allows the linear terms from distortion energies such as ARAP and Co-rotational elasticity to be represented. The second invariant $I_2$ is the Frobenius squared norm of the deformation gradient (i.e. $\|S\|^2 = \|F\|^2$) and is used to measure the least-squares distortion. The third invariant $I_3$ encodes the volume change associated with $F$.

Other choices of invariants have previously been investigated. Physics-driven methods [Irving et al. 2004; Teran et al. 2005], and mechanics approaches in general [Bonet and Wood 2008; Marsden and Hughes 1994], traditionally use the invariants of the Cauchy-Green strain tensor $C = F^T F$ to obtain the alternate invariants $I_2 = \text{tr}(C)$, $I_3 = \|C\|^2$, and $I_3 = \det(C)$. However, Stomakhin et al. [2012] and Xu et al. [2015] observed that these invariants are insufficient to express the linear distortion terms that appear in many popular models, such as the Co-rotational energy. Instead, these methods expanded distortion energies in terms of principal stretches, at the cost of verbose expressions and slower computations. Alternatively, Smith et al. [2018] used the 3D versions of the $I_2$ and $I_3$ we describe here, but their analysis is incomplete because it misses the existence of $I_1$ entirely. Shtengel et al. [2017] and Chen and Weber [2017] proposed to use the similarity and anti-similarity parts of $F$, but this decomposition is valid only in 2D.

Note that one invariant is redundant. In 2D since $I_2 = I_1^2 - 2I_3$, but we still use all three to maintain consistency between 2D and 3D.

To better understand the relationship between invariants and Hessians, we expand the quadrature-point-wise derivatives in Eqn. (5) in terms of \{I_1, I_2, I_3\}. The gradient of $\Psi$ with respect to the flattened deformation gradient $f = \text{vec}(F)$ yields:

\[
\frac{\partial \Psi}{\partial \vec{\alpha}} = \sum_i \frac{\partial \Psi}{\partial \alpha_i} \frac{\partial \alpha_i}{\partial \vec{\alpha}}.
\]
Fig. 3. Surface parameterizations computed with our algorithm applied to the Symmetric Dirichlet energy. The parameterizations are visualized in the plane (top row), and with a texture map on the 3D meshes (bottom row). From left to right, the meshes have 80k, 47k, 296k, 190k, 27k, and 1M triangles. Leftmost image ©Disney/Pixar.

The energy Hessian in terms of $f$ is then:

$$
\frac{\partial^2 \Psi}{\partial f^2} = \sum_i \frac{\partial \Psi}{\partial f_i} \frac{\partial^2 f_i}{\partial f^2} + \sum_i \frac{\partial^2 \Psi}{\partial f_i^2} \left( \frac{\partial f_i}{\partial f_i} \right) \left( \frac{\partial f_i}{\partial f_i} \right)^T + \sum_{i<j} \frac{\partial^2 \Psi}{\partial f_i \partial f_j} \left( \frac{\partial f_i}{\partial f_i} \right) \left( \frac{\partial f_j}{\partial f_j} \right)^T \left( \frac{\partial f_i}{\partial f_i} \right) \left( \frac{\partial f_j}{\partial f_j} \right)^T.
$$

(9)

Note that all the 2nd order tensors (highlighted in red overbrace and blue underbrace) are composed of first and second derivatives of the invariants, so $\{I_1, I_2, I_3\}$ can be used as proxies for analyzing the eigenstructure of the Hessian. We perform this analysis next.

4.2 Eigensystems of Invariants

The eigenvalues and eigenvectors of our $S$-based invariants can be stated in closed-form. In deriving these expressions, we will make heavy use of the tensor notation defined in §3. We note that two of the following eigensystems, the ones for $I_2$ and the 3D version of $I_3$, were also shown in Smith et al. [2018].

**Eigensystem of $I_1$:** The gradient of $I_1$ with respect to $F$ is the rotation $R$ from the polar decomposition of $F$:

$$
\frac{\partial I_1}{\partial F} = \text{vec} \left( \frac{\partial I_1}{\partial f} \right) = \text{vec}(R) = r.
$$

(10)

The Hessian of $I_1$ is then the gradient of $R$. So-called "rotation gradients" appear in many contexts, such as articulated dynamics [Twigg and Kačić-Alesić 2010] and computer vision [Papadopoulo and Lourakis 2000], and are usually computed by numerically differentiating the SVD. This operation tends to be fairly opaque, as it requires a matrix inverse [Barbić and Zhao 2011] that does not reveal any underlying structure. However, we have found that rotation gradients have simple closed-form expressions in 2D and 3D. We first consider the 2D case and introduce the $F$-based twist matrix:

$$
T = \frac{1}{\sqrt{2}} U \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} V^T.
$$

(11)

The Hessian of $I_1$ in 2D can then be written in closed-form as:

$$
\frac{\partial^2 I_1}{\partial F^2} = \frac{\partial r}{\partial f} = \frac{2}{\sigma_1 + \sigma_2} tt^T,
$$

(12)

where $t = \text{vec}(T)$ is a unit norm vector. Thus, the Hessian $\frac{\partial^2 I_1}{\partial F^2}$ is a rank-one matrix with eigenvector $t$ and eigenvalue inversely proportional to the average principal stretch. The 3D case involves three F-based twists $(T_1, T_2, T_3)$, one for each axis. For example, the x-axis twist is:

$$
T_1 = \frac{1}{\sqrt{2}} U \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} V^T.
$$

(13)

The 3D Hessian of $I_1$ is then:

$$
\frac{\partial^2 I_1}{\partial F^2} = \frac{2}{\sigma_2 + \sigma_3} t_1 t_1^T + \frac{2}{\sigma_3 + \sigma_1} t_2 t_1^T + \frac{2}{\sigma_1 + \sigma_2} t_3 t_1^T,
$$

(14)

which is a rank-three matrix with eigenvectors $t_i$ and eigenvalues inversely proportional to the average orthogonal principal stretches.

**Eigensystem of $I_2$:** We compute the derivatives of $I_2$ by noting that $I_2 = \|S\|^2 = \|F\|^2 = \|f\|^2$, which yields

$$
\frac{\partial I_2}{\partial F} = 2f \quad \text{and} \quad \frac{\partial^2 I_2}{\partial F^2} = 2I.
$$

(15)

Here, $I$ is a $d^2 \times d^2$ identity matrix ($d = 2$ or 3). The Hessian of $I_2$ is full-rank with a repeated eigenvalue equal to two, and the eigenvectors
can be set to any orthonormal bases. This invariant thus serves as a regularizer for distortion energies expressed in terms of $F$.

Eigensystem of $I_1$: The third invariant is more challenging since it is quadratic (cubic) in 2D (3D) with respect to $F$. The 2D gradient is:

$$\frac{\partial I_1}{\partial F} = g = \text{vec}(G) \quad \text{with} \quad G = U \begin{bmatrix} \sigma_2 & 0 \\ 0 & \sigma_1 \end{bmatrix} V^T. \quad (16)$$

For the Hessian, we again need the $F$-basis twist matrix $T$, but additionally introduce the following flip ($L$) and pinch ($P$) matrices:

$$L = \frac{1}{\sqrt{2}} U \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} V^T \quad \text{and} \quad P = \frac{1}{\sqrt{2}} U \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} V^T. \quad (17)$$

By denoting $l = \text{vec}(L)$ and $p = \text{vec}(P)$, we can verify that $(l, p, l, t)$ form an orthonormal bases, where $r$ denotes the normalized rotation vector, i.e., $\tilde{r} = r/\sqrt{2}$. The eigensystem of $I_1$ in 2D is then:

$$\frac{\partial^2 I_1}{\partial T^2} = \tilde{r}^T + tt^T - pp^T - I_1^T. \quad (18)$$

The eigenvalues are respectively 1 for the $\tilde{r}$ and $t$ eigenvectors, and $-1$ for $p$ and $I_1$. The rank-two subspaces spanned by $(l, t)$ and $(p, l)$ are thus arbitrary, but we have found this particular decomposition to be geometrically intuitive. Similar to 2D, the 3D gradient is:

$$\frac{\partial I_3}{\partial F} = g = \text{vec}(G) \quad \text{with} \quad G = U \begin{bmatrix} \sigma_2 \sigma_3 & 0 \\ 0 & \sigma_3 \sigma_1 \\ 0 & 0 & \sigma_1 \sigma_2 \end{bmatrix} V^T. \quad (19)$$

Similar to Eq. (13), the 3D versions of the flip matrices ($L_1, L_2, L_3$) now appear along each axis. For example, the $x$-axis flip is:

$$L_1 = \frac{1}{\sqrt{2}} U \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} V^T. \quad (20)$$

The arbitrary subspaces from the 2D case disappear, so no 3D versions of $R$ or $P$ are needed. Each twist $t_1$ is an eigenvector with corresponding eigenvalue $\sigma_1$, and $L_1$ is an eigenvector paired with $-\sigma_1$. The remaining eigenpairs are the solutions of the depressed cubic derived by Smith et al. [2018], which we list in the supplement.

4.3 Eigensystems of Arbitrary Isotropic Energies

We now have all the components necessary to revisit Eqn. (9). To begin, we prove in Appendix B that the twist $t$ and flip $I$ vectors are orthogonal to the gradient of any invariant in 2D and in 3D:

$$\left( \frac{\partial l_i}{\partial F} \right)^T t = \left( \frac{\partial l_i}{\partial F} \right)^T I = 0. \quad (21)$$

Consequently, they have zero projection onto the blue (underbraced) matrices in Eqn. (9). The remaining red (overbraced) terms are all Hessians of invariants $(\partial^2 I_1/\partial F^2)$, and we can establish via multiplication that $t$ and $I$ are eigenvectors of any 2D isotropic energy. The eigenvalues $\lambda$ then take the analytic form:

$$\frac{\partial^2 \Psi}{\partial T^2} t_1 = \frac{2}{\sigma_1 + \sigma_2} \frac{\partial \Psi}{\partial l_1} + \frac{2}{\sigma_2 + \sigma_3} \frac{\partial \Psi}{\partial l_2} + \frac{\partial \Psi}{\partial l_3} t_1 = \lambda_{\text{scalar}} t_1. \quad (22a)$$

$$\frac{\partial^2 \Psi}{\partial F^2} l_1 = \frac{2}{\sigma_2} \frac{\partial \Psi}{\partial l_2} - \frac{\partial \Psi}{\partial l_3} l_1 = \lambda_{\text{scalar}} l_1. \quad (22b)$$

Similarly, we can show that $\{t_1\}$ and $\{l_1\}$ are eigenvectors of any 3D isotropic energy, and have the analytic eigenvalues:

$$\frac{\partial^2 \Psi}{\partial T^2} t_1 = \left[ \frac{2}{\sigma_2 + \sigma_3} \frac{\partial \Psi}{\partial l_1} + \frac{2}{\sigma_3 + \sigma_1} \frac{\partial \Psi}{\partial l_2} + \frac{2}{\sigma_1 + \sigma_2} \frac{\partial \Psi}{\partial l_3} \right] t_1 = \lambda_{\text{scalar}} t_1. \quad (23a)$$

$$\frac{\partial^2 \Psi}{\partial F^2} l_1 = \frac{2}{\sigma_2} \frac{\partial \Psi}{\partial l_2} - \frac{\partial \Psi}{\partial l_3} l_1 = \lambda_{\text{scalar}} l_1. \quad (23b)$$

We have now unveiled the expressions for two (six) eigenpairs in 2D (3D). These expressions can be used to obtain the respective analytic eigenpairs for any isotropic energy. The remaining two (three) eigenpairs in 2D (3D) depend on the blue outer product terms in Eqn. (9). In the most general case, these terms result in a quadratic (cubic) system in 2D (3D), that is encoded in a $2\times2$ ($3\times3$) matrix $A$. We can construct this matrix $A$ by probing $\partial^2 \Psi/\partial F^2$ with scaling modes, e.g., the $x$-scaling vector in 2D is:

$$D_1 = U \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} V^T. \quad (24)$$

The entries of $A$ are obtained by computing the coefficients:

$$a_{ij} = d_i^T \left( \frac{\partial^2 \Psi}{\partial F^2} \right) d_j. \quad (25)$$

where $d_i = \text{vec}(D_i)$ is a unit vector orthogonal to the twist and flip vectors. The explicit expressions of $a_{ij}$ are listed in Appendix D. We have found that, for the most popular distortion energies in graphics applications, the off-diagonal entries $a_{ij}$ resolve to zero, and the scaling vectors are in fact the eigenvectors. In these cases, we can write simplified closed-form expressions for the last $d$ eigenpairs. In 2D, the scaling vector $d_1$ has the eigenvalue:

$$\frac{\partial^2 \Psi}{\partial T^2} d_1 = \left[ \frac{2}{\sigma_2} \frac{\partial \Psi}{\partial l_2} + \frac{2}{\sigma_3} \frac{\partial \Psi}{\partial l_3} + 4\sigma_1 \frac{\partial^2 \Psi}{\partial l_1^2} + \frac{2}{\sigma_2} \frac{\partial^2 \Psi}{\partial l_3^2} \right] d_1 = \lambda_{\text{scalar}} d_1. \quad (26)$$

and the eigenvalue for $d_2$ reverses the positions of $\sigma_1$ and $\sigma_2$ in the above equation. In 3D, the eigenvalue for $d_1$ becomes:

$$\frac{\partial^2 \Psi}{\partial T^2} d_1 = \left[ \frac{2}{\sigma_2} \frac{\partial \Psi}{\partial l_2} + \frac{2}{\sigma_3} \frac{\partial \Psi}{\partial l_3} + \frac{2}{\sigma_1} \frac{\partial^2 \Psi}{\partial l_1^2} + \frac{2}{\sigma_2} \frac{\partial^2 \Psi}{\partial l_3^2} + \frac{2}{\sigma_1} \frac{\partial^2 \Psi}{\partial l_2^2} \right] d_1 = \lambda_{\text{scalar}} d_1. \quad (27)$$

The eigenvalues for $d_2$ and $d_3$ respectively follow from $\sigma_2$ and $\sigma_3$, and are listed explicitly in Appendix E.

We now have a method for deriving analytic expressions for the complete eigensystem of any arbitrary isotropic energy. In all cases, simple expressions for $d^2 - d$ eigenpairs can be computed directly with Eqns. (22) and (23). As we will show in the next section, the off-diagonal entries in Eqn. (25) are often zero, and yield compact expressions for the last $d$ eigenpairs. Finally, even in the cases where a non-diagonal matrix $A$ is present, we will show in §5.3 that relatively simple expressions appear.
with ARAP, the arbitrary subspace does not need to be explicitly derived. An equivalent process yields the 3D eigenpairs:
\[\lambda^{3D}_1 = 1 + 3/\sigma_1 \quad e_1 = d_1 \]
\[\lambda^{3D}_2 = 1 + 3/\sigma_2 \quad e_2 = d_2 \]
\[\lambda^{3D}_3 = 1 + 1/I_2^2 + I_2/I_3 \quad e_3 = l \]
\[\lambda^{3D}_4 = 1 + 1/I_3^2 - I_2/I_3 \quad e_4 = t. \]

In 3D, the inverse term is more intricate and the energy can be written using the S invariants as:
\[\Psi_{SD}^3 = \frac{1}{2}I_2 + \frac{1}{8} \left( I_2^2 - I_2 \right)^2 - \frac{I_1}{I_3}. \]

The three scaling eigenpairs are
\[\lambda^{3D}_{i+3} = 1 + 3/\sigma_i^4 \quad e_i = d_i, \]
while the three twist eigenpairs are
\[\lambda^{3D}_{i+3} = 1 + 3/\sigma_i^4 \quad e_i = t_i. \]

Finally, the flip eigenpairs are
\[\lambda^{3D}_{i+6} = 1 + 3/\sigma_i^4 \quad e_i = l_i. \]

Our eigenanalysis reveals a singularity in Symmetric Dirichlet under degenerate configurations, and will generally reveal fundamental singularities in any distortion energy that it is applied to. Conversely, if a distortion energy is singularity-free, our analysis will produce singularity-free eigenvalue expressions. See Appendix F.3 for an example. Consistent with previous works, our Symmetric Dirichlet tests have found that this singularity requires no special treatment.

The Symmetric ARAP energy [Shtengel et al. 2017] adds several new terms to the Symmetric Dirichlet energy, particularly involving \(I_1\). We show in Appendix F.2 that the resulting eigensystem is similar to Symmetric Dirichlet.

### 5.3 MIPS energy

The Most Isometric Parameterization (MIPS) energy [Horman and Greiner 1999] is defined in 2D as \(\Psi_{MIPS} = I_2/I_3\). We examine it here as a case where the scaling modes remain coupled. The twist and flip eigenpairs are respectively:
\[\lambda_{i+3}^{MIPS} = 2/I_2 - I_2/I_3 \quad e_i = t_i \]
\[\lambda_{i+6}^{MIPS} = 2/I_3 + I_2/I_3 \quad e_i = l_i. \]
We conclude this section by discussing the relationship of our results. In §5.2, we showed in §5.2 that the second derivative of $I_2$ introduces an identity matrix at each quadrature point that contributes a Laplacian to the global Hessian matrix. Therefore, the method of Kovalsky et al. [2016] can be viewed as a looser approximation of the energy Hessian that assigns the F-based Hessian to identity at each quadrature point. In the case of the ARAP energy, for instance, this corresponds to the removal of part of the contribution of the twist eigenvectors from §5.1.

Comparison to Composite Majorization (CM). The CM method of Shtengel et al. [2017] considered Hessian projection to positive semi-definiteness computed per (simplicial) mesh element. However, the extension of their composite majorization to 3D appears to be difficult because it relies on the convex-concave decomposition of expressions for the singular values of the deformation gradient $F$. Such expressions exist in 2D and can be obtained using the norm of the similarity and anti-similarity parts of $F$, but this no longer holds in 3D because the problem becomes cubic. Moreover, the majorizer approximation in Shtengel et al. [2017] (Eqn. (9) in that paper) projects each term in the Hessian separately and then sums the result. Consequently, it can introduce approximations even when the Hessian was already positive-definite. By exploiting the eigenstructure of the $S$ invariants, our method takes a more holistic view and projects the Hessian after all of its terms have been gathered. If a quadrature point is already positive semi-definite, our approach keeps its Hessian untouched. An extensive case study of our approach compared to CM for the specific case of the ARAP energy is presented in the supplement.

Comparison to Chen and Weber [2017]. Closely related to our work, Chen and Weber [2017] considered analytic eigensystems for isotropic distortion energies. However, their results are limited to 2D because their analysis is dependent on complex derivatives. Moreover, their closed-form expressions exhibit a series of invalid states that need to be resolved on a case-by-case basis. In our supplement, we revisit the derivation of Chen and Weber [2017] and...
We exercise our method on a range of graphics applications that work of Teran et al. [2005] used the
As we will show in §6, our analytic expressions offer simpler code, from MKL [Intel 2018]. For AQP, the Laplacian decouples across
2D, we solve linear systems using the Cholesky decomposition
∞ parallel to a threshold: \( \| \nabla \Psi \|_{\infty} \leq 10^{-4} \). In 2D, we solve linear systems using the Cholesky decomposition from MKL [Intel 2018]. For AQP, the Laplacian decouples across
each dimension, so we pre-factorize and solve two independent

Comparison to Stomakhin et al. [2012]; Teran et al. [2005]. The work of Teran et al. [2005] used the C-centric invariants (see Appendix A) to show that the indefiniteness of 3D Hessians can be probed using three 2×2 eigenproblems and one 3×3 eigenproblem. Stomakhin et al. [2012] extended these results by replacing the C invariants with the 3D principal stretches. However, none of these analyses reveal the closed-form eigensystem expressions we found here. In particular, these previous techniques assembled the Hessian eigenvectors numerically via a series of tensor contractions. As we will show in §6, our analytic expressions offer simpler code, leading up to a 3.69× speedup in Hessian construction compared to Stomakhin et al. [2012].

Comparison to Smith et al. [2018]. The work of Smith et al. [2018] established the eigensystems of the \( I_2 \) invariant and \( J = \det F \), which correspond to our \( I_2 \) and \( I_3 \) respectively. However, they failed to find the eigensystem for the \( I_C \) invariant (see their §7), which prevented them from fully characterizing all Cauchy-Green-based energies. Most significantly, their formulation missed the existence of the \( I_1 \) invariant entirely, making it impossible to discover almost all of the closed-form eigensystems that we presented in §3. With the exception of the MIPS energy, all of the energies we address in this paper contain the \( I_1 \) invariant, which puts these energies beyond the reach of the analysis in Smith et al. [2018]. Conversely, our analysis yields the analytic eigensystems for both \( I_1 \) and \( I_C \). As shown in Appendix A, the \( I_C \) invariant can be written in terms of our \( S \) invariants, thus establishing its eigensystem as just a specific application of our derivation. We are thus able to explicitly list the closed-form eigensystem of \( I_C \) in §3 of the supplemental materials.

6 RESULTS
We exercise our method on a range of graphics applications that span geometry processing and physical simulation. On the geometry processing front, we apply our approach to surface parameterization (§6.1), volume deformation (§6.2), and interpolation (§6.3). Halfway between geometry processing and physical simulation, we show that our method can be used as a pre-processing procedure to obtain intersection-free input surfaces for a cloth simulator (§6.4). Finally, we show that our approach can be used in volume simulation with dynamics (§6.5). In these tasks, we employ triangle meshes with linear elements in 2D and tetrahedral meshes with linear elements in 3D. We use a standard projected Newton solver augmented with a line search for all tests. Alg. 1 shows pseudo-code of our projected Newton, and includes our analytical Hessian computation. To prevent elements from inverting with Symmetric Dirichlet and Symmetric ARAP, we use the backtracking line search of Smith and Schaefer [2015] to limit the step-size to the largest inversion-avoiding value. For all remaining energies, we use a simple backtracking line search. We terminate the minimization when the norm of the gradient falls below a threshold: \( \| \nabla \Psi \|_{\infty} \leq 10^{-4} \). systems. In 3D, computing this decomposition is more costly due to the increased bandwidth of the stiffness matrix. Instead, we found that Jacobi-preconditioned conjugate gradient provides adequate performance. All SVDs and numerical eigendecompositions were computed with Eigen [Guennebaud et al. 2010].

6.1 Surface Parameterization
Fig. 3 shows the parameterization of six meshes computed by minimizing the Symmetric Dirichlet energy with our approach, where each was initialized with a Tutte embedding of the 3D surface. In Fig. 6 (left), we compare the performance for these examples obtained with our analytic projected Newton solver, composite majorization [Shtengel et al. 2017] (using code provided by authors), SLIM [Rabinovich et al. 2017] (using the optimized parallel code provided by the authors), and a numerical projected Newton solver (see, e.g., [Fu and Liu 2016]). The latter performs the numerical eigendecomposition of 6×6 matrices corresponding to the \( x \)-based Hessian per simplicial element. In contrast, our approach computes the analytic eigensystems for \( F \)-based Hessians of size 4×4 evaluated per quadrature point. We also employed our closed-form expressions to assemble the uncoupled \( x \)-based Hessians in lieu of auto-differentiation, so the implementation of the numerical projected Newton solver was also improved by our results. For the Symmetric Dirichlet energy, composite majorization and our method gave nearly identical performance in all but one test, and yielded consistently faster solves than the numerical projected

```
Function Projected_Newton_Solver(x₀)
for i ← 0 to n do
  b_i ← ∇Ψ(x_i) // Eqn. (5a)
  if \( \| b_i \|_{\infty} \leq 10^{-4} \) then
    return x_i
  end
  H_i ← Project_Hessian(x_i)
  d_i ← -H_i⁻¹b_i
  a_i ← Line_Search(x_i, d_i)
  x_{i+1} ← x_i + a_i d_i
end
return x_{n+1}

Function Project_Hessian(x)
H ← 0
for every quadrature point q do
  F ← Compute_Deformation_Gradient(q, x)
  f ← vec(F)
  \{U, Σ, V\} ← Compute_SVD(F)
  \{λ_i, e_i\} ← Eval_Energy_EigenSystem(U, Σ, V)
  H_q ← \sum_i max(λ_i, 0) e_i e_iᵀ
  H ← H + |q| (\∂f/∂x)ᵀ H_q (\∂f/∂x) // Eqn. (5b)
end
return H
```

Algorithm 1: Projected Newton Pseudocode. Our approach allows \( \text{Eval}_e \text{nergy}_e \text{igenSystem}(U, Σ, V) \) to be implemented in closed-form.
Newton solver. While the latter can terminate in fewer iterations than the competing methods, the cost of the 6x6 per-element eigen-decompositions outweigh these savings across all tests. Fig. 9 plots the optimization progress for the bear model, both in terms of iterations and wall clock time (in seconds). In the supplemental material, we provide the plots for the other meshes, as well as the performance statistics of a larger dataset of 41 models.

We also computed surface parameterizations using the ARAP energy. The composite majorization for ARAP was not addressed in Shtengel et al. [2017], so we considered two candidate derivations: one using the convex-concave decomposition obtained by their similarity and anti-similarity expansion of $F$, and another expressed in terms of our $S$ invariants (detailed derivations are in the supplemental material). In contrast to our formulation, we observed that both composite majorization schemes led to looser Hessian approximations even in the regime where the energy Hessian is already positive semi-definite. As shown in the supplement, the behavior of CM is highly dependent on the choice of decomposition. A naïve choice produces a solve that is equivalent to AQP [Kovalsky et al. 2016], degrading convergence to first-order. In Fig. 10, we show the number of iterations and the wall clock time (in seconds) for the bear model. We compared the performance of these methods to our solver over a dataset of 41 models, and complete results are in the supplemental material. Fig. 6 (right) shows a selection of these results. In a few instances, the $x$-based formulation completed sooner, but for the vast majority of cases our approach was clearly superior. In particular, for the low resolution car mesh (Car, Fig. 6), the $x$-based formulation had the best performance. On a higher resolution version of the same car mesh obtained through Catmull-Clark subdivision (Car 2, Fig. 6), however, we found that our method had the fastest performance.

6.2 Volume Deformation
We explored the effectiveness of our approach when computing a 3D deformation. Given a volumetric cow mesh containing 23k tetrahedra, we generated handles from the feet and head by hard-constraining and rigidly transforming groups of surface mesh vertices. The resulting position of the deformed mesh was computed by minimizing the distortion energy when a handle was transformed. Fig. 4 shows the results obtained after prescribing a transform to each handle. Our projected Newton solver successfully converged to a deformed shape, taking 2.46s with ARAP, 21.60s with Symmetric Dirichlet, and 23.19s with Symmetric ARAP. As composite majorization does not generalize to 3D, we instead compared to the per-element numerical projected Newton solver. Symmetric Dirichlet and Symmetric ARAP with per-quadrature point (9x9) numerical projection took a total of 40.91s and 53.97s, and Hessian projection was 5.49x slower than our method, respectively. Symmetric Dirichlet and Symmetric ARAP with per-element (12x12) numerical projection took a total of 44.12s and 56.64s, and Hessian projection was 6.54x and 8.67x slower than our method, respectively. Symmetric Dirichlet and Symmetric ARAP with [Stomakhin et al. 2012] projection took a total of 32.67s and 37.50s, and Hessian projection was 3.47x and 3.69x slower than our method, respectively. For comparisons with the 9 x 9 and 12 x 12 eigendecompositions, we also tested the LAPACK dsyevd and dsytrd/dorgtr/dsteqr routines as implemented in Intel’s MKL library, but found them to be roughly 30% slower than Eigen’s self-adjoint eigendecomposition. We further compare against the projection technique from [Teran et al. 2005], which requires a model that can be phrased in terms of the C invariants. Computing the deformation in Fig. 4 with the St. Venant-Kirchhoff (see Appendix F.3) material, which can be expressed in the C invariants, we found that our method took a total of 33.95s, while [Teran et al. 2005] took 47.12s. Per-quadrature point (9x9) numerical projection took 2.78x faster than [Teran et al. 2005], 3.70x faster than per-quadrature point numerical projection, and 4.89x faster than per-element numerical projection.

We further compare against the projection technique from [Teran et al. 2005], which requires a model that can be phrased in terms of the C invariants. Computing the deformation in Fig. 4 with the St. Venant-Kirchhoff (see Appendix F.3) material, which can be expressed in the C invariants, we found that our method took a total of 33.95s, while [Teran et al. 2005] took 47.12s. Per-quadrature point (9x9) numerical projection took 2.78x faster than [Teran et al. 2005], 3.70x faster than per-quadrature point numerical projection, and 4.89x faster than per-element numerical projection.

This example also demonstrates the generality of our approach, since we were able to extend Symmetric Dirichlet and Symmetric ARAP from 2D to 3D with no additional numerical machinery. Furthermore, we performed a live editing session with the Spot mesh using our solver (see our supplemental video). Fig. 2 depicts different Yoga poses generated by interactively manipulating handles and then minimizing the Symmetric Dirichlet energy with our method.
6.3 Shape Interpolation in 3D

The ability to robustly optimize 3D deformation energies also leads to an effective method for 3D shape interpolation. Following the framework of Chao et al. [2010], we defined an interpolated energy as the convex combination of two deformation energies, i.e., \( \Psi_t = (1 - t) \Psi_1 + t \Psi_2 \). Given two shapes, the deformation energy \( \Psi_1 \) was measured against the first shape, while the deformation \( \Psi_2 \) was measured against the second shape. The gradient and Hessian of this interpolated energy are simply respectively combinations of \( \Psi_1 \)'s and \( \Psi_2 \)'s gradient and Hessian. The sum of two positive-definite Hessians is still positive definite, so by projecting the individual Hessians we removed any indefiniteness from the sum. To obtain the interpolated shape at some value \( t \in [0, 1] \), we minimized \( \Psi_t \) for that value of \( t \) with our projected Newton solver. We tested this technique by minimizing the Symmetric ARAP energy with two examples. First, we interpolated between two poses of a bar (Fig. 7), demonstrating that the solver is robust under bending and twisting deformations. In this example, the bar model had 4,528 tetrahedra and performed at 0.029s per frame. Next, we interpolated between two poses of a hand (Fig. 5) in order to validate our method with more complex input geometry. In this case, the hand model had 23,692 tetrahedra and took 0.21s per frame. For both sequences, we initialized the solver using the result of the previous frame.

6.4 Intersection-Free Cloth Simulation Inputs

The minimization of volume distortion can also be used as a pre-process to clean up animation artifacts prior to cloth simulation. For the short film Bao, the input animations for a character often contained self-intersections that would cause non-physical tangling when a cloth simulation was later performed over its body (Fig. 8 (bottom-left)). These intersections were sufficiently severe that even the robust collision response [Bridson et al. 2002; Harmon et al. 2008] and untangling strategies [Baraff et al. 2003] in our production cloth simulator were unable to produce usable results. These challenging and non-physical configurations are quite common, as animators typically focus their efforts on geometry that is visible from the camera. Artifacts in non-visible regions can still destabilize a simulation, however, requiring time consuming and tedious manual cleanup. Previous attempts at Pixar to remove the self-intersections using surface-only techniques failed to give robust results, but our performance projective Newton solver enabled a volume-based workflow.

First, we generated a tetrahedral mesh for the T-pose of the body surface and warped its elements via a simple Poisson solve to match the surface of the input animation. These warped, intersecting volumetric meshes were then set as the rest configuration \( \mathbf{R} \) for our optimization at their respective frames. Starting from the tetrahedralized T-pose, we removed intersections by running a collision-aware ARAP volume solve for every frame (Fig. 8 (top-middle)). The intersection-free surfaces produced by these optimized volumes were then used as an input body collider for our cloth simulator (Fig. 8 (top-right)). The volumetric mesh contained 429,871 tetrahedra, and each frame of animation was processed in 4.59 seconds, on average. The workflow was sufficiently robust that it successfully produced artifact-free results for every clothed character in the short on the very first try. Our supplemental video displays an example of this application, and further details can be found in [Wong et al. 2018].

6.5 Adding Dynamics

Distortion energies can be incorporated into volume simulations with dynamics. In this case, a minor modification is applied to the projected Newton solver so that mass and damping matrices are added to the global Hessian matrix in order to control inertia and time stepping. Moreover, the solver must often resolve inverted elements induced by input animations. Our approach is especially suited to this scenario, since our closed-form expressions are valid for any value of deformation gradient, even under inversion. Fig. 1 shows a sequence from Cars 3 that was computed by integrating our ARAP minimization into a production volume simulator. Our expressions were also used to simulate the character Jack-Jack in Incredibles 2. Our production volume simulator is a semi-implicit solver [Baraff and Witkin 1998] that uses a BDF2 time discretization [Ascher and Petzold 1998], combined with continuous collision detection (e.g. [Tang et al. 2010]), and with impulse-based and penalty-based collision resolution [Bridson et al. 2002].

7 CONCLUSIONS AND FUTURE WORK

We have presented a systematic method for obtaining the analytic, per-quadrature point eigensystem of any isotropic distortion energy. The most immediate application of these results is to recent Domain Specific Languages (DSLs) for physical simulation, such as Ebb [Bernstein et al. 2016] and Simit [Kjolstad et al. 2016]. The hyperelastic strain energies used in physical simulations are functionally identical to geometric distortion energies. However, existing DSLs have only been able to support a limited number of these energies, as their gradients and Hessians are usually hand-derived and manually implemented on a case-by-case basis.

Although automatic differentiation methods are available, they result in slow and tortuous codes that are unable to discover structures like our closed-form rotation gradients (Eqn. (12)). Thus, no existing method has been able to automatically convert an arbitrary isotropic energy into efficient, parsimonious, projected Newton code. The expressions we have presented accomplish exactly this task. A user should now be able to type in any arbitrary isotropic

Fig. 7. We interpolate between an undeformed, tetrahedralized bar (left) and a deformed bar (right) by minimizing the 3D Symmetric ARAP energy.
For the short film Bao, intersection-free surfaces were obtained by running a collision-aware volume simulation with our ARAP expressions (§6.4). These were used as collision surfaces in a cloth simulator, and produced untangled clothing. Top row: In the original animation, an arm that is not visible from the camera intersects the torso (left). Volume simulation finds the closest intersection-free state (middle). This state is then sent to the cloth simulator (right). Bottom row: Cloth simulation using the original character animation (left). The character’s arm penetrates her shirt and artifacts appear along the hip. Simulation results obtained using the intersection-free volume solution are artifact-free (right). © Disney/Pixar

Fig. 9. Solver iterations (left) and wall clock time (right) to compute a parameterization with the Symmetric Dirichlet energy over the bear mesh using our method (pink circles), comp. major. (blue squares), per-element numerical projection (green upright triangles), SLIM (yellow diamonds), and Newton’s method without Hessian projection (purple inverted triangle). The termination threshold is denoted by a dashed line.

strain energy, and moments later experiment with the behavior of that energy. This will enable qualitative exploration of the space of hyperelastic and geometric distortion energies in a way that was not previously possible, and enable experimentation with more sophisticated energies.

A better understanding of the underlying eigensystems also have potential implications in model reduction. Many such reduction methods use the intrinsic eigensystems of the underlying models to both construct reduced-order basis functions [An et al. 2008; von Tycowicz et al. 2013] and compute reduced-order gradients and Hessians [Barbić and James 2005]. In particular, exact reduced-order derivatives have only ever been discovered for the St. Venant-Kirchhoff energy [Barbić and James 2005], which is a 4th order polynomial. However, many of the distortion energies we examined are lower-order polynomials, so with our S-centric invariants and their eigensystems in hand, it may be possible to derive other closed-form, reduced-order expressions.

Finally, the eigensystems we derived have implications for multigrid solvers [McAdams et al. 2011]. Our improved understanding of the spectral behavior at each quadrature point may be able to inform the design of better restriction and prolongation operators.

ACKNOWLEDGEMENTS

We would like to thank David Eberle and Audrey Wong for the intersection-free simulation tests and Henry Garcia for running the inflating tire test.

REFERENCES


A GENERALITY OF S INVARIANTS

We show that tensor invariants can be written in terms of $S$ instead of $C$ without loss of generality. Denoting the $C$ invariants as $I_C = \text{tr}(C)$,
$H_C = \|C\|^2$, and $H_C = \text{det}(C)$, it is straightforward to verify that:

\[ H_C = I_2, \]
\[ H_{C, 2D} = I_2^2 - 2I_2^2 \]
\[ H_{C, 3D} = \frac{1}{2}I_2^2 - \frac{1}{2}I_2^4 + I_2^2I_2 + 4I_1I_3 \]
\[ H_{C} = I_2^2. \]

Any isotropic energy written in terms of the C invariants can be re-written with the above substitutions. The converse does not hold, because the $C = F^TF$ squaring discards the sign information from F, and precludes negative values of $I_1$ and $I_3$.

**B ORTHOGONALITY OF TWIST AND FLIP MATRICES**

**Lemma B.1.** The twist t and flip f vectors are orthogonal to the gradients of any S invariant in 2D and 3D, i.e., $(\partial I_1/\partial \theta)^T t = (\partial I_3/\partial \theta)^T 1 = 0$.

**Proof:** We first note that the matrix versions of the invariant gradients in Eqns. (10), (15), and (16) correspond to matrices of the form $UXV^T$ with diagonal matrices $X_i$. Since the middle term of the twist matrix in 2D has a zero diagonal (Eqns. (11)), we have:

\[
(\partial I_1/\partial \theta)^T t = r^T = R : T = \frac{1}{\sqrt{2}} \text{tr}
\begin{pmatrix}
1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0
\end{pmatrix}
= 0,
\]
\[
(\partial I_2/\partial \theta)^T t = 2f^T = 2F : T = \frac{2}{\sqrt{2}} \text{tr}
\begin{pmatrix}
\sigma_1 & 0 & 0 & -1 \\
0 & \sigma_2 & 1 & 0
\end{pmatrix}
= 0,
\]
\[
(\partial I_3/\partial \theta)^T t = g^T = G : T = \frac{1}{\sqrt{2}} \text{tr}
\begin{pmatrix}
\sigma_2 & 0 & 0 & -1 \\
0 & \sigma_1 & 1 & 0
\end{pmatrix}
= 0.
\]

Similarly, the flip matrix has a zero diagonal and thus it is orthogonal to the invariant gradients. The same proof holds in 3D trivially. □

**C 2D ARAP HESSIAN IMPLEMENTATION**

The following 2D Hessian code is implemented using Eigen [Guennebaud et al. 2010]. We assume that a SVD implementation that moves all reflections into $\Sigma$ and vec function are both available.

using namespace Eigen;
Matrix4d ARAP::Hessian(const Matrix2d& F) {
  // Get the SVD of F
  Matrix2d U, Sigma, V;
  SVD(F, U, Sigma, V);
  // Build the eigenvector
  Matrix2d twist;
  twist << 0, 1, -1, 0;
  twist *= 1.0 / sqrt(2.0);
  const Vector4d e = vec(U * twist * V.transpose());
  // Filter the eigenvector to guarantee SPD-ness
  const double L_1 = Sigma.trace();
  const double filtered = (L_1 >= 2.0) ? 2.0 / L_1 : 1.0;
  // Build the final Hessian
  Matrix4d H;
  H.setIdentity();
  H *= filtered * (e * e.transpose());
  H *= 2.0;
  return H;
}

**D SCALING MODE MATRIX**

The general matrix $A$ that must be solved in 2D to obtain the final two eigenpairs is given by the entries:

\[
\begin{align*}
a_{ij}^{2D} &= \frac{\partial^2 \Psi_q}{\partial I_2^2} + \frac{\partial^2 \Psi_q}{\partial I_1^2} + 4\sigma_1^2 \frac{\partial^2 \Psi_q}{\partial I_2^2} + \sigma_1^2 \frac{\partial^2 \Psi_q}{\partial I_1^2} \\
&+ 2I_1 \frac{\partial^2 \Psi_q}{\partial I_1 \partial I_2} + 2I_2 \frac{\partial^2 \Psi_q}{\partial I_2 \partial I_3} + I_1 \frac{\partial^2 \Psi_q}{\partial I_3 \partial I_1} \\
&+ 4\sigma_1 \frac{\partial^2 \Psi_q}{\partial I_1 \partial I_2} + 4\sigma_2 \frac{\partial^2 \Psi_q}{\partial I_2 \partial I_3} + 2\sigma_1 \frac{\partial^2 \Psi_q}{\partial I_3 \partial I_1}.
\end{align*}
\]

In 3D, the matrix entries are:

\[
\begin{align*}
a_{ij}^{3D} &= \frac{\partial^2 \Psi_q}{\partial I_3^2} + \frac{\partial^2 \Psi_q}{\partial I_1^2} + 4\sigma_1 \frac{\partial^2 \Psi_q}{\partial I_3^2} + \sigma_1 \frac{\partial^2 \Psi_q}{\partial I_1^2} \\
&+ 2\left(I_1 - \sigma_k\right) \frac{\partial^2 \Psi_q}{\partial I_1 \partial I_3} + 2\sigma_k \left(I_3 - \sigma_k\right) \frac{\partial^2 \Psi_q}{\partial I_2 \partial I_3} + \sigma_k \left(I_3 - \sigma_k\right) \frac{\partial^2 \Psi_q}{\partial I_3 \partial I_1} \\
&+ 4\sigma_1 \frac{\partial^2 \Psi_q}{\partial I_1 \partial I_3} + 4\sigma_2 \frac{\partial^2 \Psi_q}{\partial I_2 \partial I_3} + 2\sigma_1 \frac{\partial^2 \Psi_q}{\partial I_3 \partial I_1}.
\end{align*}
\]

The two equations above can be further simplified using $\sigma_1 \sigma_j = \delta_k$. We found that this substitution can sometimes help during implementation, as each entry then only depends on one singular value in addition to the three invariants.

**E DECOUPLED 3D SCALING EIGENVALUES**

In many cases, the 3D scaling modes become decoupled, in which case their eigenvalues can be written down explicitly. The expression for $\lambda_{\text{scale}}$ is given in Eqn. (27), and the other two eigenvalues are:

\[
\begin{align*}
\frac{\partial^2 \Psi_q}{\partial I_2^2} d_2 &= \frac{1}{4} \left( \frac{\partial^2 \Psi_q}{I_2} + \frac{\partial^2 \Psi_q}{I_3^2} + 4\sigma_1^2 \frac{\partial^2 \Psi_q}{\partial I_1^2} + 4\sigma_2^2 \frac{\partial^2 \Psi_q}{\partial I_1 \partial I_2} + 4\sigma_3^2 \frac{\partial^2 \Psi_q}{\partial I_1 \partial I_3} \right) \\
&+ \frac{1}{4} \left( \frac{\partial^2 \Psi_q}{I_1} + \frac{\partial^2 \Psi_q}{I_3^2} + 4\sigma_1 \frac{\partial^2 \Psi_q}{\partial I_2 \partial I_3} + 4\sigma_3 \frac{\partial^2 \Psi_q}{\partial I_2 \partial I_1} + 4\sigma_2 \frac{\partial^2 \Psi_q}{\partial I_1 \partial I_3} \right) \\
&+ \frac{1}{4} \left( \frac{\partial^2 \Psi_q}{I_1} + \frac{\partial^2 \Psi_q}{I_2^2} + 4\sigma_2 \frac{\partial^2 \Psi_q}{\partial I_2 \partial I_3} + 4\sigma_3 \frac{\partial^2 \Psi_q}{\partial I_1 \partial I_3} + 4\sigma_1 \frac{\partial^2 \Psi_q}{\partial I_2 \partial I_1} \right)
\end{align*}
\]

**F ADDITIONAL ENERGY EIGENSYSTEMS**

We provide the closed-form expressions for the eigensystems of extended versions of the distortion energies from §5.
F.1 Co-rotational Energy

The Co-rotational energy (see, e.g., [McAdams et al. 2011]) is defined as \( \Psi_{CR} = \mu \| F - R \|^2 + \frac{\kappa}{2} \| t \|^2 / (S - I) \), where \( \kappa \) is a volume penalty. It can be written using the S invariants as \( \Psi_{CR} = \mu (l_2 - 2l_1 + d^2) + \frac{\kappa}{2} (l_1^2 - 2l_1 + d^2) \). In 2D, the eigensystem becomes:

\[
\begin{align*}
\lambda_1^{\text{CR}} &= 2\mu + 2\kappa (l_1 - 2) / l_1^2 & e_1 &= t \\
\lambda_2^{\text{CR}} &= 2\mu + 2\kappa & e_2 &= r / \sqrt{2} \\
\lambda_3^{\text{CR}} &= 2\mu & e_{3,4} &= \{p, l\}.
\end{align*}
\]

In 3D, these expressions become:

\[
\begin{align*}
\lambda_1^{\text{CR}} &= 2\mu + 2\kappa (l_1 - 3 - 2\mu) / (l_1 - 3) & e_1 &= t_1 \\
\lambda_2^{\text{CR}} &= 2\mu + 2\kappa (l_1 - 3 - 2\mu) / (l_1 - 3) & e_2 &= t_2 \\
\lambda_3^{\text{CR}} &= 2\mu + 2\kappa (l_1 - 3 - 2\mu) / (l_1 - 3) & e_3 &= t_3 \\
\lambda_4^{\text{CR}} &= 2\mu + 3\kappa & e_4 &= r / \sqrt{3} \\
\lambda_5^{\text{CR}} &= 2\mu & e_{5 \ldots 9} &= \text{5D subspace}.
\end{align*}
\]

Aside from the scaling by \( \mu \), the additions to the ARAP energy are most visible in the \( t_1 \) eigenvalues, where a new \( \kappa (l_1 - 3) \) term appears. Rotation also becomes a unique eigenvector. Similar to ARAP, there is no need to explicitly construct Co-rotational’s 5D subspace in practice. See Appendix C.

F.2 Symmetric ARAP

The Symmetric ARAP energy [Shtengel et al. 2017] is defined as \( \Psi_{\text{ARAP}} = \mu / 2 \left( \| F - R \|^2 + \| F^{-1} - R^{-1} \|^2 \right) \). Using the S invariants, this energy can be written in 2D as:

\[
\Psi_{\text{ARAP}} = \mu / 2 (l_2 - \mu l_1 + \mu l_2 / l_3^2 - \mu l_1 / l_3 + \mu d^2).
\]

The 2\( l_1 \) and 2\( l_1 / l_3 \) terms are new relative to the 2D Symmetric Dirichlet. The 2D eigensystem is:

\[
\begin{align*}
\lambda_1^{\text{ARAP}} &= \mu \left( 1 - 2 / \sigma_1^3 + 3 / \sigma_1^4 \right) & e_i &= d_i \\
\lambda_2^{\text{ARAP}} &= \mu + \mu (l_2 / l_3 - l_1 + 1) / l_3^2 & e_3 &= l \\
\lambda_3^{\text{ARAP}} &= \mu + \mu \left( (l_2 + l_1) / l_3^2 - l_2 / l_3 - 2 \right) / l_1 & e_4 &= t.
\end{align*}
\]

The 3D version expression is expressed in terms of \( S \) invariants as:

\[
\Psi_{\text{ARAP}} = \mu l_2 / 2 (l_2 - \mu l_1 + \mu (l_2 / l_3 - l_1 + 1) / l_3^2 - \mu l_1 / l_3 + \mu d^2).
\]

The eigensystem contains equivalent scaling modes:

\[
\lambda_1^{\text{ARAP}} = \mu \left( 1 - 2 / \sigma_1^3 + 3 / \sigma_1^4 \right) & e_i = d_i.
\]

Its twist eigenpairs are:

\[
\lambda_1^{\text{ARAP}} = \mu \left[ \frac{\mu}{\sigma_j + \sigma_k} \left( \frac{1}{\sigma_j^3} + \frac{1}{\sigma_k^3} - \frac{1}{\sigma_j^3} - \frac{1}{\sigma_k^3} - 2 \right) \right] e_{i+3} = t_i,
\]

which are defined over the \((i, j, k)\) triplets \((1, 2, 3), (2, 3, 1)\) and \((3, 1, 2)\).

Finally, the flip eigenpairs are:

\[
\lambda_1^{\text{ARAP}} = \mu \left[ 1 + \frac{1}{(\sigma_j \sigma_k)^2} + \frac{\sigma_j^2 + \sigma_k^2}{(\sigma_j \sigma_k)^2} - \frac{\sigma_j^2}{(\sigma_j \sigma_k)^2} - \frac{\sigma_k^2}{(\sigma_j \sigma_k)^2} - 2 \right] e_{i+6} = l_i.
\]

F.3 St. Venant-Kirchhoff

The St. Venant-Kirchhoff energy is defined as \( \Psi_{\text{SVK}} = \mu \| E \| + \frac{\kappa}{2} t r^2 \). Using the \( S \) invariants, this energy can be written in 3D as:

\[
\Psi_{\text{SVK}} = \frac{\kappa}{8} (l_2 - 3)^2 + \frac{\mu}{8} (8 l_1 l_3 + l_2^2 + 2 l_1^2 l_2 - 4 l_2 l_1^3 - l_1^4 + 6).
\]

The 3D twist and flip components of the eigensystem are:

\[
\lambda_i^{\text{SVK}} = \mu + \frac{\kappa}{2} (l_2 - 3) + \mu \left( \sigma_j^2 + \sigma_k^2 - \sigma_j \sigma_k \right) & e_i = t_i,
\]

\[
\lambda_{i+3}^{\text{SVK}} = \mu + \frac{\kappa}{2} (l_2 - 3) + \mu \left( \sigma_j^2 + \sigma_k^2 + \sigma_j \sigma_k \right) & e_{i+3} = l_i.
\]

The remaining three components are given by the eigensystem of the \( 3 \times 3 \) matrix \( A = QAQ^T \) with entries:

\[
\begin{align*}
\lambda_1^{\text{SVK}} &= \mu + \frac{\kappa}{2} (l_2 - 3) + \sigma_j^2 + \sigma_k^2 + \sigma_j \sigma_k \\
\lambda_2^{\text{SVK}} &= \kappa \sigma_j \sigma_k.
\end{align*}
\]

Given the eigenvalues \( \Lambda_i \) and eigenvectors \( Q_i \) of \( A \), we complete St. Venant-Kirchhoff’s eigensystem with:

\[
\lambda_i^{\text{SVK}} = \Lambda_i & e_{i+6} = \text{vec} \left( U \text{ diag} \left( Q_i \right) V^T \right).
\]

Observe that, like the energy itself, St. Venant-Kirchhoff’s eigensystem is singularity free.