Geometric Optimization via Composite Majorization

ANNA SHTENGENL, Weizmann Institute of Science
ROI PORANNE and OLGA SORKINE-HORNUNG, ETH Zurich
SHAHAR Z. KOVALSKY, Duke University
YARON LIPMAN, Weizmann Institute of Science

Many algorithms on meshes require the minimization of composite objectives, i.e., energies that are compositions of simpler parts. Canonical examples include mesh parameterization and deformation. We propose a second order optimization approach that exploits this composite structure to efficiently converge to a local minimum.

Our main observation is that a convex-concave decomposition of the energy constituents is simple and readily available in many cases of practical relevance in graphics. We utilize such convex-concave decompositions to define a tight convex majorizer of the energy, which we employ as a convex second order approximation of the objective function. In contrast to existing approaches that largely use only local convexification, our method is able to take advantage of a more global view on the energy landscape. Our experiments on triangular meshes demonstrate that our approach outperforms the state of the art on standard problems in geometry processing, and potentially provide a unified framework for developing efficient geometric optimization algorithms.

CCS Concepts: • Computing methodologies → Computer graphics; • Theory of computation → Nonconvex optimization;

Additional Key Words and Phrases: second order optimization, majorization, convex-concave, distortion, geometry processing

ACM Reference format:

1 INTRODUCTION

Mesh processing tasks in computer graphics, including deformation and parameterization, are often cast as nonlinear minimization problems of the general form:

$$\min_x f(x)$$

(1)

Active research effort is dedicated to develop numerical optimization methods for the solution of such problems, taking advantage of the known context, e.g., the geometric structure of the problem.

Existing optimization algorithms typically produce a sequence of approximations, \(x_n\), designed to converge to a solution of (1). To this end, most approaches use, either explicitly or implicitly, a local quadratic approximation of the objective function: they construct an osculating convex quadric to \(f\) at \(x_n\), whose minimization determines the next approximation \(x_{n+1}\). From this point of view, the difference between the various methods lies in the particular choice of an osculating quadric, or more precisely, the choice of its Hessian.

The archetypal Newton algorithm [Lange 2013; Nocedal and Wright 2006] uses the Hessian of \(f\) itself to define the osculating quadric. For strictly convex functions, this leads to a well-defined algorithm with quadratic order of convergence. For non-convex functions, however, the Hessian is often indefinite and thus Newton’s algorithm is no longer guaranteed to be a descent algorithm. Several general heuristics exist on how to modify Newton’s Hessian so as to force it into being positive semidefinite. Unfortunately, there is no clear generic strategy for such modification that works well in all, or even most cases. Consequently, generic algorithms are usually satisfied with just finding some decent direction, and Newton’s desirable convergence rates are often not attained in practice.

In computer graphics, a specific effort is dedicated to developing efficient optimization algorithms for objective functions defined on meshes. These algorithms take advantage of the particular structure of the energies used in geometry processing and introduce, albeit sometimes implicitly, a convex osculating quadric used to determine their iterations. For instance, [Kovalsky et al. 2016; Liu et al. 2008; Sorkine and Alexa 2007] essentially replace the Hessian with the mesh Laplacian, [Liu et al. 2016] further use low-rank quasi-Newton updates to better approximate the Hessian, and [Rabinovich et al. 2017] reweigh the Laplacian to improve the effectiveness of their iterations. These are all first order methods, meaning they do not directly use second order derivatives of the energy, and therefore generally fail to achieve high convergence rate, particularly as they approach convergence.

Our goal is to devise a second order optimization approach applicable to a generic class of composite nonlinear energies in computer graphics, namely, we are concerned with objective functions that can be represented as the composition of simpler functions. Our

strategy for picking a convex osculating quadric at $x_n$ is based on: (i) exploiting the composite structure for constructing a convex majorizer to $f$ centered at $x_n$, and (ii) computing its Hessian at $x_n$. The majorizer provides a tight convex upper bound to $f$ in a well understood, nontrivial neighborhood of $x_n$ and therefore provides a well justified choice of a positive semidefinite Hessian at $x_n$. This eliminates the need to heuristically enforce positive semidefiniteness.

Although a tight convex majorizer for a general non-linear function $f$ is often difficult to obtain, we make the observation that it can be derived analytically from convex-concave decompositions of the functions composing $f$. Consequently, the resulting Hessian formula at $x_n$ takes a simple analytic form and is efficient to compute in practice.

We argue that common problems in computer graphics are often compatible with our assumptions on the composite structure of the objective function. We demonstrate the utility of our approach in geometry processing of triangular surface meshes via instantiation of our Hessian formula for several popular energies, such as energies defined in terms of singular values. In general, applying our method requires convex-concave decompositions of the functions composing $f$. We evaluate the performance of the proposed approach and show that it outperforms state of the art algorithms on a variety of geometric optimization problems. Figure 1 shows deformations of a bar released from a bent initial configuration (left) computed using several state of the art methods, further details are provided in Section 5; our method, Composite Majorization (CM–ours), is the first to converge to the rest state of the bar (right).

2 BACKGROUND

A general meta-algorithm for the unconstrained minimization of an energy $f$ is summarized in Algorithm 1. It iteratively solves

$$Hp = -\nabla f(x),$$

(2)

where $H$ is a positive definite matrix and $x$ is the current iterate, to determine a search direction $p$, and takes a step in the direction $p$ that reduces the value of $f$. To simplify the exposition, we describe the unconstrained case; constraints, which are integral and important to many problems, can be handled via the solution of the corresponding KKT system in the linear case [Nocedal and Wright 2006] or by adding penalty to the objective function in the non-linear case, see Section 5.5 for an example.

Since $H$ is positive definite, the solution of $Hp = -\nabla f(x)$ corresponds to a minimum of the convex osculating quadric to $f$ at $x$, given by

$$g(z) = \frac{1}{2} (z-x)^T H (z-x) + \nabla f(x)^T (z-x).$$

(3)

Convexity is essential, as it guarantees that the search direction $p$ is indeed a descent direction, i.e., $p^T \nabla f(x) < 0$. Line search then ensures that each iteration of Algorithm 1 reduces the objective.

From this perspective, nonlinear optimization algorithms differ in the choice of Hessian $H$ of the osculating quadric they employ, as well as how they enforce its positive definiteness. We present our approach for choosing $H$ in Section 3. The rest of this section is devoted to an overview of related methods and puts our approach in a relevant context.

---

Algorithm 1: Meta-algorithm for nonlinear optimization

Input: initial guess $x$

repeat

$p \leftarrow -H^{-1}\nabla f(x)$; \hspace{2cm} (compute search direction)
$t \leftarrow \arg \min_{\varepsilon \in (0,1]} f(x + \tau p)$; \hspace{2cm} (perform line search)
$x \leftarrow x + t \ p$; \hspace{2cm} (make a step)

until convergence;

---

Newton’s algorithm. Newton’s algorithm makes the choice $H = \nabla^2 f$. This amounts to having the quadric $q$ coincide with $f$ up to second order. For convex functions this implies that $p$ is a descent direction. Furthermore, under additional mild assumptions, Algorithm 1 with this choice of osculating quadric converges quadratically—probably Newton algorithm’s most attractive property.

However, when optimizing a non-convex function, which is the typical case in geometry processing, $\nabla^2 f$ is not positive semidefinite (PSD) and $p$ is no longer guaranteed to be a descent direction. Generic techniques simply replace $\nabla^2 f$ with a positive definite approximating matrix $H$. The simplest techniques, such as modifying the true Hessian by adding a multiple of the identity, are usually suboptimal and sacrifice convergence rate. Other heuristics, such as modified factorizations (e.g., spectral, Cholesky) and projection may perform better, but they also introduce a significant computational overhead and may become computationally prohibitive. Unfortunately, there is no universal answer to what constitutes a good approximation. See Section 3.4 in [Nocedal and Wright 2006] for a comprehensive discussion on Newton’s method with Hessian modifications.

Hessian modifications have also been adapted to geometry processing in several works [Fu and Liu 2016; Teran et al. 2005]. These works observe that in the common case of separable energies, which decompose as a sum over the elements of the mesh, it is sufficient to modify the sub-Hessians corresponding to each mesh element. Teran et al. [2005] propose to individually project the Hessian of each element onto the PSD cone, thus alleviating the computational burden of projecting the entire, usually large, Hessian at once. The resulting approximate Hessian is PSD due to linearity, and has been observed to work better in practice than the generic Hessian modifications [Liu et al. 2016].

Fig. 2 compares different Hessian modification methods for minimizing the symmetric Dirichlet energy (see equation (24)) of a small-scale sphere mesh comprising $\approx 400$ vertices: Newton’s method (without any modification), our method (CM–ours), the per element Hessians projections (PN), and full Hessian projections (Full PN). For the Full PN we tested two variants: (i) replacing eigenvalues of the Hessian that are smaller than $\varepsilon$ to $\varepsilon$ (Full-PN(a)); or (ii) adding a multiple of the identity matrix to the Hessian (Full-PN(b)). As can be seen from the graph, the latter “off-the-shelf” projections lead to longer damped phases. Note that the standard Newton’s method fails to find a descent direction at some point and thus halts.

First order methods. Motivated by the particular structure of the energies commonly used in geometry processing, several works have exploited the mesh Laplacian [Pinkall and Polthier 1993] for approximating the Hessian. The Laplacian is implicitly used in the
local/global algorithm [Liu et al. 2008; Sorkine and Alexa 2007] for minimizing the as-rigid-as-possible energy. Kovalsky et al. [2016] advocate using the Laplacian instead of the true Hessian for general energies in geometry processing. Liu et al. [2016] suggest to refine the Hessian approximation for elastic simulations by applying low-rank quasi-Newton updates to the Laplacian. These works all benefit from the observation that the Laplacian is, by definition, positive semidefinite. Furthermore, by exploiting the fixed Hessian approximation (i.e., independent of $\mathbf{x}$) they are able to devise particularly efficient iterations.

Rabinovich et al. [2017] are inspired by the fast initial progress of global-local iterations and advocate the use of a reweighted Laplacian for the minimization of nonlinear energies on meshes. Their algorithm is shown to be very efficient in minimizing the energy from arbitrary feasible initializations, but it significantly slows down near an optimum, and so additional Newton iterations might be required if an accurate solution is pursued.

Generally, since first order methods do not fully exploit second order information, they usually do not achieve the convergence rates associated with Newton-type algorithms. This is particularly the case when approaching a minimum, where Newton methods enter their powerful, quadratically convergent phase [Boyd and Vandenberghe 2004].

**Gauss-Newton.** The Gauss-Newton algorithm is often discussed in the context of nonlinear least squares [Boyd and Vandenberghe 2004; Nocedal and Wright 2006]. It considers energies of the form $f = \frac{1}{2} \sum g_i^2$, for which the Hessian is given by

$$\nabla^2 f = \sum_i \nabla g_i^T \nabla g_i + \sum_i g_i \nabla^2 g_i.$$  

The Gauss-Newton method omits the last term of $\nabla^2 f$ and instead uses

$$H = \sum_i \nabla g_i^T \nabla g_i,$$

which is clearly PSD. The rationale is that when the $g_i$ values are small (e.g., close to a minimum), $H$ well approximates the Hessian $\nabla^2 f$. Gauss-Newton has been successfully used in computer graphics for problems of adequate form, e.g., [Huang et al. 2009; Tang et al. 2014].

Functions of the more general form $f = \sum_i h(g_i)$ have also been studied in [Martens 2010; Schraudolph 2002]. Their generalized Gauss-Newton Hessian approximation

$$H = \sum_i \nabla g_i^T (\nabla^2 h) \nabla g_i$$

is derived similarly to the Gauss-Newton approximation, and reduces to it for $h(u) = \frac{1}{2} u^2$. It is guaranteed to be PSD if $h$ is convex.

**Majorization-Minimization and Convex-concave.** Majorization Minimization (MM) is an optimization framework that works by fitting, at each iteration, a convex surrogate upper bound, called a majorizer, which is minimized to drive the objective function downhill. It thus replaces a difficult optimization with a sequence of simpler optimization problems. Simplicity is often attained by variable separation, linearization, convexification and other approaches that exploit the structure of specific problems, see [Hunter and Lange 2004] and [Lange 2013, chapter 8].

The use of MM is prevalent in various fields of numerical optimization, albeit often implicitly: in statistics, the popular Expectation Maximization (EM) is a special instance of MM [Lange 2013, chapter 9]; so are the classes of iteratively reweighted least squares (IRLS) and iterative shrinkage thresholding (IST) algorithms, often considered in the literature of sparse linear inverse problems [Daubechies et al. 2004, 2010], or Stress Majorization for multidimensional scaling [Gansner et al. 2004], to name just a few examples. Instances of MM have also been employed in computer graphics, for example in texture synthesis [kwatra et al. 2005] or feature matching [Lipman et al. 2014]. In fact, the popular local-global procedure for the minimization of the as-rigid-as-possible energy [Liu et al. 2008; Sorkine and Alexa 2007] is another example of a MM algorithm.

Most related to this work is the convex-concave procedure [Lipp and Boyd 2014; Yuille and Rangarajan 2003]. It is a subfamily of MM algorithms concerned with objective functions $f$ that decompose as the sum $f = f^++f^−$ of a convex and a concave function. Given such decomposition, a global convex majorizer of $f$ is easily constructed by replacing the concave $f^−$ with its linear approximation at $\mathbf{x}_0$. As with the more general MM, this approach has also (implicitly) found applications in computer graphics; for example, Yuille and Rangarajan [2003] show that Sinkhorn’s algorithm, used by Solomon et al., [2015] for the computation of geometric optimal transportation, is an instance of the convex-concave procedure.

In the context of Algorithm 1 and equation (2), it is thus natural to set the approximate Hessian to be the Hessian of the convex majorizer,

$$H = \nabla^2 f^+,$$  

which is PSD by construction; this approach is also known as the MM-Gradient algorithm, see [Lange 2013, chapter 10.4]. Nevertheless, while the majority of functions admit a convex-concave decomposition [Hartman et al. 1959], it is not unique, and the specific chosen decomposition directly affects $H$ and thus the convergence. Finding efficient convex-concave decompositions is typically difficult for complicated functions and, in general, remains a challenge.

3 APPROACH

This paper is concerned with the choice of a convex approximate Hessian $H$ to be used in the generic nonlinear optimization Algorithm 1. We propose a general methodology for choosing $H$ in a broad class of non-convex optimization problems, demonstrated to be particularly suitable for geometry processing tasks.
Our approach is based on a novel convex majorization technique for composition of functions for which convex-concave decompositions are known. In turn, we use the majorizer’s Hessian as $H$ in Algorithm 1. Our choice of $H$ is therefore: (i) guaranteed to be PSD; (ii) corresponds to a non-trivial local convex majorizer; (iii) often takes simple analytic form and thus is simple to evaluate; and (iv) empirically outperforms alternative choices.

We begin by highlighting our main results and subsequently present their rigorous definitions and detailed derivations.

3.1 Summary of main results
We consider energies of the form

$$f(x) = \sum_i h_i(g_i(x)) = \sum_i (h_i \circ g_i)(x),$$

(6)

where $h_i : \mathbb{R}^k \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}^k$ are $C^2$ functions with convex-concave decompositions. That is, they each decompose as

$$h_i = h_i^+ + h_i^-,$$

and the vector-valued functions $g_i^+$ convex in each of their coordinates and, respectively, $h_i^-$ concave.

We will first show how to use these decompositions to construct a convex majorizer to $f$ centered at a general point $x_0$ that takes the form

$$\tilde{f} = \sum_i h_i \circ g_i,$$

(8)

where $\tilde{h}_i$ is a majorizer of $h_i$ centered at $g_i(x_0)$, and $[g_i]$ is a vector function whose entries are either majorizers or minorizers of $g_i$ centered at $x_0$.

We further prove that $\tilde{f}$ is a non-trivial local majorizer, in the sense that we define the non-trivial set $\Omega (x_0) \subseteq \mathbb{R}^n$ for which $\tilde{f}$ majorizes $f$. This comes in contrast to an arbitrarily defined local majorizer, that may majorize $f$ in only an arbitrarily small neighborhood of $x_0$. The inset illustrates our majorizer (green) for a simple 2-dimensional composed function (yellow) centered at the black disk; the region $\Omega (x_0)$, over which majorization is guaranteed, is colored red.

Lastly, since $\tilde{f}$ is a convex majorizer, it is natural to employ its minimization as a proxy for the minimization of $f$ itself. Therefore, we propose to set $H$ in Algorithm 1 to be the positive semidefinite Hessian of $\tilde{f}$ at $x_0$, given by

$$H = \nabla^2 \tilde{f} = \sum_i \left[ \begin{array}{c} \partial h_i \\
\partial g_i \end{array} \right]^T \nabla^2 h_i^+ \left[ \begin{array}{c} \partial g_i \\
\partial x \end{array} \right] + \sum_j \left( \frac{\partial h_i}{\partial u_{ij}} \right) + \nabla^2 g_i^+ + \sum_{ij} \left( \frac{\partial h_i}{\partial u_{ij}} \right) \nabla^2 g_i^-$$

(9)

where $(\cdot)$ keeps only positive numbers (linear rectifier), $(\cdot)_-$ only negative numbers, and $g_i^+$ and $g_i^-$ are the $j$-th coordinates of the vector-valued functions $g_i^+$ and $g_i^-$, respectively.

Since $H$ only involves the Hessian of the convex-concave parts of the $h_i$’s and $g_i$’s, it is simple to compute. In fact, it often takes an analytic form, as we demonstrate in Section 4. Moreover, we show in practice that this choice of $H$ typically outperforms alternative approaches for choosing a convex approximate Hessian.

3.2 Derivation of majorizer and Hessian
To simplify the notations, let us omit the summation in equation (6) and consider a function of the form

$$f(x) = h(g(x)) = h(g_1(x), \ldots, g_k(x))$$

(10)

and let $h = h^+ + h^-$ and $g_j = g_j^+ + g_j^-$ be the respective convex-concave decompositions. Our derivations and results straightforwardly extend to the form (6).

**Preliminaries.** For a scalar function $r$ with a convex-concave decomposition $r = r^+ + r^-$ we define its majorizer at $x_0$ as

$$\tilde{r}(x; x_0) = r^+(x) + r^-(x_0) + \nabla r^-(x_0)(x - x_0),$$

(11)

and its minorizer at $x_0$ as

$$\bar{r}(x; x_0) = r^-(x) + r^+(x_0) + \nabla r^+(x_0)(x - x_0).$$

(12)

Note that $\tilde{r}$ (or $\bar{r}$) satisfies:

1. It is a convex (concave) function.
2. It coincides with $r$ up to first order (i.e., value and first derivative) at $x = x_0$.
3. It is a global majorizer (minorizer) of $r$. That is, for all $x$

$$\tilde{r}(x; x_0) \leq r(x) \leq \bar{r}(x; x_0).$$

**Derivation.** Fix $x_0$, set $u_0 = g(x_0)$ and let $s_j(u) = \text{sign} \left( \frac{\partial h}{\partial u_{ij}} \right)$, where we define here the sign function via

$$\text{sign}(t) = \begin{cases} 1 & t \geq 0 \\ -1 & t < 0 \end{cases}$$

(13)

Intuitively, $s_j$ is determined by whether the majorizer of $h$ at $u_0$ is increasing- or decreasing-monotone in its $j$-th entry.

Next, define the function $\left[ g_j \right] : \mathbb{R}^n \to \mathbb{R}^k$ by each of its entries

$$g_j(x; x_0) = \begin{cases} \frac{\partial g_j}{\partial x} & S_j > 0 \\ \frac{\partial g_j}{\partial x} & S_j < 0 \end{cases}$$

(14)

Note that each entry of $[g]$ is either a majorizer or a minorizer of the corresponding entry in $g$ at $u_0$, in correspondence with the monotonicity pattern of $h$.

Equipped with these definitions we now propose our majorizer for $f$ at $x_0$, and define

$$\tilde{f}(x; x_0) = \tilde{h}(\left[ g \right](x; x_0) ; u_0),$$

(15)

thus concretizing the definition of equation (8).

To show that $\tilde{f}$ is a majorizer, we first have to define the domain on which it holds. We define

$$S(x_0) = \{ x | s_j([g](x; x_0)) = s_j(u_0) \},$$

(16)

the set of all points $x$ for which $[g]$ is compatible with the monotonicity pattern of $h$ at $u_0$. Our majorizing domain $\Omega (x_0)$ is defined by restricting our attention to the path connected component subset containing $x_0$:

$$\Omega (x_0) = \{ x | x \text{ is path connected to } x_0 \text{ in } S(x_0) \}. $$

(17)
The inset figure in Section 3.1 depicts an example of a majorizer of a 2-dimensional function $f$ over the domain $\Omega(x_0)$ marked red over the graph of $f$.

**Proposition 3.1.** The function $\tilde{f}$ defined in equation (15) is a convex majorizer of $f$ at $x_0$ over the set $\Omega(x_0)$ defined in equation (16).

**Proof.** First, to see that $\tilde{f}$ is convex over $\Omega(x_0)$ let us consider its Hessian. Applying the chain rule yields

$$
\nabla^2 \tilde{f}(x; x_0) = \frac{\partial |g|^T}{\partial x} \nabla^2 h + \frac{\partial |g|}{\partial x} + \sum_j \frac{\partial h}{\partial u_j} \left[ \nabla^2 g_j^+ + s_j(u_0) \right] \geq 0
$$

where $\nabla^2 h$, $\nabla^2 g_j^+$ and $-\nabla^2 g_j^-$ are all PSD matrices. Note, however, that the claims of Proposition 3.1 are not guaranteed to hold in case of a convex-concave decomposition of Hessians that does not correspond to a global functional convex-concave decomposition.

**Relation to generalized Gauss-Newton.** It is interesting to note that the first term of equation (9) is exactly the generalized Gauss-Newton matrix [Schraudolph 2002], which involves second order derivatives of $h_i$ but only first order derivatives of $g_i$. From that aspect, our approach supplements the generalized Gauss-Newton approximate Hessian by adding positive semidefinite terms accounting for the second order derivatives of $g_i$ as well.

**Hessian approximation and term gathering.** Proposition 3.1 implies that the derived PSD Hessian $H \succcurlyeq 0$ satisfies $H \succeq \nabla^2 f(x_0)$ (inequalities are in the PSD sense). A natural question is whether equality holds when $\nabla^2 f(x_0) \succeq 0$. Unfortunately, this is generally not the case, similarly to other non-full Hessian projection methods.

A tighter approximation to the Hessian can be achieved by term gathering, as we explain next. Note that the second and third terms of equation (9) are of the basic form

$$
M = (a)_+ M_1 + (b)_+ M_2,
$$

where $M_1$ and $M_2$ are both PSD. We observe that if $M_1$, $M_2$ share a common term $P$, such that $M_1 - P$, $M_2 - P$ and $P$ are all PSD, then since $(a + b)_+ \leq (a)_+ + (b)_+$ we have

$$
M \succeq (a)_+ (M_1 - P) + (b)_+ (M_2 - P) + (a + b)_+ P \succeq 0.
$$

Thus, term gathering produces a Hessian approximation $H'$ that satisfies $H \succeq H' \succeq 0$ and $H' \succeq \nabla^2 f(x_0)$. Empirically, we observed that term gathering may improve convergence by a small margin.

**4 ENERGIES ON TRIANGULAR MESHES**

Variational approaches in geometry processing often aim to minimize energies related to some notion of distortion. They often strive to approximate an isometric map [Aigerman et al. 2015; Chao et al. 2010; Liu et al. 2008; Smith and Schaefer 2015; Sorkine and Alexa 2007], conformal map [Ben-Chen et al. 2008; Desbrun et al. 2002; Hormann and Greiner 2000; Lévy et al. 2002; Muller et al. 2008; Weber et al. 2012], or a harmonic map [Ben-Chen et al. 2009]. Other approaches minimize energies that represent physical quantities [Wardetzky et al. 2007; Xu et al. 2015].

Due to their prevalence in geometry processing we concentrate on triangular meshes. We represent a piecewise linear mapping of a surface mesh with $N$ vertices into the plane as a vector $x \in \mathbb{R}^{2N}$, encoding the image of each of the vertices. We further denote by

$$
J_i = J_i(x) = \begin{bmatrix} a_i(x) & b_i(x) \\ c_i(x) & d_i(x) \end{bmatrix}
$$

the $2 \times 2$ Jacobian matrix, with respect to an arbitrary local frame (i.e., deformation gradient), associated with the $i$-th mesh triangle, and note that its entries $a_i, b_i, c_i, d_i$ are linear in $x$.

**Singular value energy templates.** Energies defined in terms of the singular values of $J_i$ are prevalent in geometry processing. They
can often be formulated as

$$f(x) = \sum_i h_i(\Sigma_i, \sigma_i) |t_i|,$$  \hspace{1cm} (20)

where summation is over all triangles, $\Sigma_i$ and $\sigma_i$ are the singular values of the Jacobian $J_i$ of the $i$-th triangle, and $|t_i|$ is its area.

Energies of the type (20) are already in the form our approach addresses, namely, $\sum_i h_i \circ g_i$ with

$$g_i = (\Sigma_i, \sigma_i).$$  \hspace{1cm} (21)

To obtain a convex-concave decomposition for $g_i$ recall that [Lipman 2012; Smith and Schaefer 2015] use the fact that

$$\Sigma_i = \|\alpha_i\| + \|\beta_i\|$$

$$\sigma_i = \|\alpha_i\| - \|\beta_i\|,$$

where

$$\alpha_i = \frac{1}{2} \left[ a_i + d_i \right], \quad \beta_i = \frac{1}{2} \left[ a_i - d_i \right]$$  \hspace{1cm} (22)

represent the closest similarity (complex derivative) and closest anti-similarity (complex anti-derivative), respectively [Chien et al. 2016].

Therefore, it is only natural to decompose $g_i = g_i^+ + g_i^-$ with

$$g_i^+ = (\|\alpha_i\| + \|\beta_i\|, \|\alpha_i\|)$$

$$g_i^- = (0, -\|\beta_i\|).$$  \hspace{1cm} (23)

Clearly, $g_i^+$ is convex and $g_i^-$ is concave. Note that $g_i^+$ and $g_i^-$ have $\|\alpha_i\|$ and $\|\beta_i\|$ in common; and thus, in our implementation, we use the common term gathering in equation (18).

Next, we apply this template for two energies that measure isometric distortion.

**Symmetric Dirichlet energy.** The isometric distortion energy considered in [Schreiner et al. 2004; Smith and Schaefer 2015] is

$$f_{\text{ISO}}(x) = \sum_i \left( \Sigma_i^2 + \Sigma_i^{-2} + \alpha_i^2 + \alpha_i^{-2} \right) |t_i|.$$  \hspace{1cm} (24)

It immediately takes the form $\sum_i h_i \circ g_i$, with $g_i = (\Sigma_i, \sigma_i)$ as in equation (21) and

$$h_i(u, v) = \left( u^2 + u^2 + v^2 + v^2 \right) |t_i|.$$  \hspace{1cm} (25)

Note that $h_i$ is convex with respect to $u, v > 0$ and thus, with the decomposition (23) of $g_i$, the formula for $H$ in equation (9) can be readily computed to obtain a PSD Hessian. Figure 3 depicts mesh deformations obtained by minimizing this energy.

**Symmetric as-rigid-as-possible (ARAP) energy.** This isometric distortion energy is a symmetric version of the ARAP energy that equally penalizes stretching and shrinking and hence also resists element flips. We formulate it as

$$f_{\text{ARAP}}(x) = \sum_i \left( \left( \Sigma_i - 1 \right)^2 + \left( \sigma_i^{-1} - 1 \right)^2 \right) |t_i|.$$  \hspace{1cm} (25)

It takes the form $\sum_i h_i \circ g_i$, with $g_i = (\Sigma_i, \sigma_i)$ and

$$h_i(u, v) = \left( (u - 1)^2 + (v - 1)^2 \right) |t_i|.$$  \hspace{1cm} (25)

Note that while the term dependent on $u$ in $h_i$ is convex, the term dependent on $v$ is not. Since $(v - 1)^2$ is convex on $(0, 1.5]$ and concave on $(1.5, \infty)$, it is easy to analytically decompose it, using its linearization at $v = 1.5$, as (omitting the area term for brevity)

$$h^+(u, v) = \begin{cases} 
(u - 1)^2 + (v - 1)^2 & 0 < v \leq 1.5 \\
(u - 1)^2 - \frac{8}{\pi^2} v & v > 1.5 
\end{cases}$$

$$h^-(u, v) = \begin{cases} 
0 & 0 < v \leq 1.5 \\
(v - 1)^2 - \frac{8}{\pi^2} v & v > 1.5 
\end{cases}$$

In turn, the formula for $H$ in equation (9) can be readily used for minimizing the symmetric ARAP energy, see Figure 3 for an example.

**Strain energy density for Neo-Hookean material.** In two dimensions, the strain energy density for Neo-Hookean material can be formulated as

$$f_{\text{NH}}(x) = \sum_i \left[ \frac{\mu}{2} \left( \frac{\|J_i\|^2}{\det J_i} - 2 \right) + \frac{\kappa}{2} (\det J_i - 1)^2 \right] |t_i|,$$  \hspace{1cm} (26)

where $\mu$ and $\kappa$ are modeling coefficients, related to the response of the material to shear stress and compression resistance, respectively [Xu et al. 2015]. Geometrically, the first term of the objective is related to conformal distortion (MIPS) [Hormann and Greiner 2000], whereas the second term accounts for area change.

This energy takes the form $\sum_i h_i \circ g_i$ with

$$h_i(u, v) = \left( \frac{\mu u^2}{2} + \frac{\kappa}{2} (v - 1)^2 \right) |t_i|$$

$$g_i(x) = (\|J_i\|_F, \det J_i).$$

Note that $h_i$ is convex and so is $g_i$. For its second entry we note that, with the notations of equation (19), we have

$$\det J_i = a_i d_i - b_i c_i,$$

$$= \frac{1}{4} \left[ (a_i + d_i)^2 + (b_i - c_i)^2 \right] - \frac{1}{4} \left( (a_i - d_i)^2 + (b_i + c_i)^2 \right).$$
We tested our approach on a number of standard problems in geometry processing formulated using the energies introduced in the previous section. We compared our approach (CM-ours) to the following state of the art approaches:

- **Newton’s Algorithm** – We followed the approach described in [Liu et al. 2016; Teran et al. 2005] and implemented a projected Newton (PN) solver, wherein the Hessian of each triangle is computed using automatic differentiation, and then individually projected via eigen-decomposition onto the PSD cone. We used our analytic Hessian formula in equation (9) to efficiently compute the indefinite Hessian (by simply dropping the clamping).

- **Scalable Locally Injective Mappings** – We implemented the approach of [Rabinovich et al. 2017], wherein a modified Laplacian takes the place of the Hessian (SLIM). We followed the authors’ update formula for the reweighting.

- **Accelerated Quadratic Proxy** – We implemented the approach of [Kovalsky et al. 2016], wherein the Laplacian itself replaces the Hessian. We compared to their accelerated (AQP) as well as non-accelerated (QP) variants.

Thus we take

\[ g_i^+(x) = \left( \|J_i\|_F, \frac{1}{4} \left( (a_i + d_i)^2 + (b_i - c_i)^2 \right) \right) \]

\[ g_i^-(x) = \left( 0, -\frac{1}{4} \left( (a_i - d_i)^2 + (b_i + c_i)^2 \right) \right). \]

The advantage of using our approach for this energy is demonstrated in Figure 4; see Section 5.2 for additional details.

5 EXPERIMENTAL EVALUATION

We tested our approach on a number of standard problems in geometry processing formulated using the energies introduced in the previous section. We compared our approach (CM-ours) to the following state of the art approaches:

- **Newton’s Algorithm** – We followed the approach described in [Liu et al. 2016; Teran et al. 2005] and implemented a projected Newton (PN) solver, wherein the Hessian of each triangle is computed using automatic differentiation, and then individually projected via eigen-decomposition onto the PSD cone. We used our analytic Hessian formula in equation (9) to efficiently compute the indefinite Hessian (by simply dropping the clamping).

- **Scalable Locally Injective Mappings** – We implemented the approach of [Rabinovich et al. 2017], wherein a modified Laplacian takes the place of the Hessian (SLIM). We followed the authors’ update formula for the reweighting.

- **Accelerated Quadratic Proxy** – We implemented the approach of [Kovalsky et al. 2016], wherein the Laplacian itself replaces the Hessian. We compared to their accelerated (AQP) as well as non-accelerated (QP) variants.

5.1 Surface parameterization

We tested our approach on surface parameterization computed by minimizing the symmetric Dirichlet energy, equation (24). We followed the standard protocol of using Tutte’s embedding for computing a bijective initial parameterization [Kovalsky et al. 2016; Rabinovich et al. 2017; Smith and Schaefer 2015]. In turn, we note that the resulting parameterizations are guaranteed to be locally injective, as we minimize an inversion-resisting energy and accordingly restrict our line-search.

We conducted an extensive evaluation of our approach on a dataset of 30 surfaces taken from [Myles et al. 2014; Rabinovich et al. 2017]. Figure 5 shows the results obtained in four examples, comparing the performance of our approach with that of alternative approaches. Our approach (blue) outperforms the others by a significant margin. Table 1 further summarizes our evaluation. Noticeably, the second order approaches (CM-ours, PN) require substantially fewer iterations to converge. Consequently, their overall convergence time is also lower, even though the first order alternatives (SLIM, AQP) spend less time per iteration.

5.2 Mesh deformation

We evaluated our approach for computing mesh deformations. Similarly to surface parameterization, our method demonstrates superior performance. Figure 1 shows a comparative experiment where we minimize the symmetric ARAP of equation (25): we first bend a bar, then release it and measure the time it takes to reach the rest pose. In this example our method is more than 4 times faster than projected Newton and substantially faster than the others. Figure 3 demonstrates the deformations obtained by minimizing the symmetric Dirichlet and symmetric ARAP energies using our method.

We further argue that, for deformation, short per iteration time is as important as convergence rate. Since deformation is an interactive process, it must be reactive to the input of the user. In our experiment we observed that our iterations are also faster than the projected
Figure 5. Surface parameterization: comparing our approach with the state of the art for the minimization of the symmetric Dirichlet Energy. The graphs show how the different optimization approaches reduce the energy as a function of the number of iterations or run time. Noticeably, our approach (blue) outperforms the alternative approaches. The top row shows the planar parameterization obtained using our optimization (shading indicating isometric distortion) and a texture respectively mapped back to the surface.

Figure 6. Scalability. We applied our approach and SLIM for parameterizing meshes of increasing resolutions representing the same surface. The graphs show the number of iterations and run time required for problems of different sizes to converge.

5.3 Seamless parameterization

Introducing compatibility constraints on the seams of a cut surface finds applications in quadrangulation [Bommes et al. 2009] and surface mapping [Aigerman et al. 2015] among others. We have experimented with supplementing our surface parameterization algorithm (Section 5.1) with two types of seam compatibility constraints:

**Similarity** – where we constrain corresponding edges that belong to the same seam, to be related by a similarity transformation. Namely, if \( u_1, u_2 \in \mathbb{R}^2 \) are the end-points of a seam edge corresponding to \( v_1, v_2 \in \mathbb{R}^2 \), we require that

\[
  v_2 - v_1 = T (u_2 - u_1) = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} (u_2 - u_1),
\]

where \( T \) is a similarity matrix parameterized by \( a, b \in \mathbb{R} \).

**Seamless** – where the matrix \( T \) above is constrained to be a rotation by a multiple of 90 degrees. We realize this constraint by enforcing equation (27) with the additional quadratic constraints

\[
  a^2 + b^2 = 1; \quad ab = 0.
\]

Note that this compatibility constraint is a fundamental prerequisite for producing a quadrangulation, however, it is insufficient on its own [Bommes et al. 2013].

Figure 7 illustrates the difference between parameterizations that satisfy either of these compatibility constraints.
Our experiments used the same naive penalty schedule for all experiments. Figure 8 (top) shows results obtained using our approach. Examples (weights increased from $10^5$) suffices to satisfy the similarity seam compatibility constraints on surfaces from the dataset of [Myles et al. 2014]. We have computed parameterizations with seam compatibility constraints on surfaces of [Myles et al. 2014]. We cut each of the surfaces along geodesics corresponding to a minimal spanning tree defined by the singularity points provided with the dataset. We then used our Algorithm 1 with our choice of Hessian spanning tree defined by the singularity points provided with the dataset. We demonstrated the utility of our approach in surface parameterization and mesh deformation, where its performance exceeds the state of the art. Note that SLIM and AQP typically fail to reduce the energy to the values achieved by our approach.

Table 1. Surface parameterization: comparing our approach with state-of-the-art approaches for the minimization of the symmetric Dirichlet Energy. Experiments that did not converge within 6000 iterations are omitted from the table. Our algorithm compares favorably in both iteration count and convergence time in all cases excluding two instances where it is found to be comparable.

We have computed parameterizations with seam compatibility constraints on surfaces of the dataset from [Myles et al. 2014]. We cut each of the surfaces along geodesics corresponding to a minimal spanning tree defined by the singularity points provided with the dataset. We then used our Algorithm 1 with our choice of Hessian spanning tree defined by the singularity points provided with the dataset. We demonstrated the utility of our approach in surface parameterization and mesh deformation, where its performance exceeds the state of the art. Note that SLIM and AQP typically fail to reduce the energy to the values achieved by our approach.

6 CONCLUDING REMARKS

In this paper we presented an optimization approach for energies that are compositions of simpler parts, each admitting a convex-concave decomposition. Our approach exploits this structure to devise a novel convex majorizer, and consequently an efficient second order optimization algorithm. Thus, each iteration of our algorithm is in a sense aware of the global energy landscape, rather than relying on strictly local second order approximation. We demonstrated the utility of our approach in surface parameterization and mesh deformation, where its performance exceeds the state of the art.
One limitation of our approach is that the majorizer we define depends on the particular decomposition used, i.e., different decompositions of the objective function generally correspond to different majorizers, and thus to different choices of $H$. In this aspect, we note that the notions of optimality or “quality” of $H$ are ill-defined; we are unaware of a method for comparing different choices of $H$ that predicts their effectiveness for optimization. For instance, we observed that the difference between our choice of $H$ and that of projected Newton’s is an indefinite matrix.

We also note that our choice of Hessian in equation (9) is, in general, not guaranteed to coincide with the true Hessian of the objective function on its convex regions. In some cases, this can be directly revealed by the functions composing our majorizer. Detecting this discrepancy can perhaps be used, in future work, to improve convergence, e.g., by reverting to the true Hessian whenever possible.

Lastly, the approach presented in this work can be extended and applied to problems in higher dimension. In these cases, we believe that our analytic formula would provide an even better relative performance improvement over approaches such as projected Newton, that need to project (e.g., via eigen-decomposition) larger matrices.

ACKNOWLEDGMENTS

This research was supported in part by the European Research Council starting grants Surf-Comp (Grant No. 307754) and iModel (Grant No. 306877), I-CORE program of the Israel PBC and ISF (Grant No. 4/11) and the Simons Foundation Math+X Investigator award. The authors would like to thank Olga Diamanti, Michael Rabinovich and Ashish Myles for sharing their code; Amir Porat for producing the supplemental video; and the anonymous reviewers for their helpful comments and suggestions.

REFERENCES


We also note that our choice of Hessian in equation (9) is, in general, not guaranteed to coincide with the true Hessian of the objective function on its convex regions. In some cases, this can be directly revealed by the functions composing our majorizer. Detecting this discrepancy can perhaps be used, in future work, to improve convergence, e.g., by reverting to the true Hessian whenever possible.

Lastly, the approach presented in this work can be extended and applied to problems in higher dimension. In these cases, we believe that our analytic formula would provide an even better relative performance improvement over approaches such as projected Newton, that need to project (e.g., via eigen-decomposition) larger matrices.

ACKNOWLEDGMENTS

This research was supported in part by the European Research Council starting grants Surf-Comp (Grant No. 307754) and iModel (Grant No. 306877), I-CORE program of the Israel PBC and ISF (Grant No. 4/11) and the Simons Foundation Math+X Investigator award. The authors would like to thank Olga Diamanti, Michael Rabinovich and Ashish Myles for sharing their code; Amir Porat for producing the supplemental video; and the anonymous reviewers for their helpful comments and suggestions.

REFERENCES


