Accelerated Quadratic Proxy for Geometric Optimization

Shahar Z. Kovalsky Meirav Galun Yaron Lipman
Weizmann Institute of Science

Abstract

We present the Accelerated Quadratic Proxy (AQP) - a simple first-order algorithm for the optimization of geometric energies defined over triangular and tetrahedral meshes.

The main stumbling block of current optimization techniques used to minimize geometric energies over meshes is slow convergence due to ill-conditioning of the energies at their minima. We observe that this ill-conditioning is in large part due to a Laplacian-like term existing in these energies. Consequently, we suggest to locally use a quadratic polynomial proxy, whose Hessian is taken to be the Laplacian, in order to achieve a preconditioning effect. This already improves stability and convergence, but more importantly allows incorporating acceleration in an almost universal way, that is independent of mesh size and of the specific energy considered.

Experiments with AQP show it is rather insensitive to mesh resolution and requires a nearly constant number of iterations to converge; this is in strong contrast to other popular optimization techniques used today such as Accelerated Gradient Descent and Quasi-Newton methods, *e.g.*, L-BFGS. We have tested AQP for mesh deformation in 2D and 3D as well as for surface parameterization, and found it to provide a considerable speedup over common baseline techniques.

Keywords: optimization, first order methods, acceleration, preconditioning, simplicial meshes, distortion, geometry

Concepts: •**Computing methodologies** → **Mesh models;** •**Mathematics of computing** → **Nonconvex optimization;**

1 Introduction

Problems in computer graphics, including deformation and parameterization, often take the form of an optimization problem. Typically, these problems share a common structure – they are all defined over tessellations of the domain (*e.g.*, meshes) and aim at minimizing a geometric energy defined for each element and summed over the tessellation. Nonetheless, generic off-the-shelf optimization tools, which are generally used for their optimization, do not explicitly exploit this structure.

The goal of this work is to take a first step in the direction of utilizing this structure, and introduce a simple first-order algorithm designed for the optimization of geometric functionals over meshes. In particular, we aim at an efficient, effective and scalable algorithm.

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. © 2016 ACM.
SIGGRAPH '16 Technical Paper,, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925920

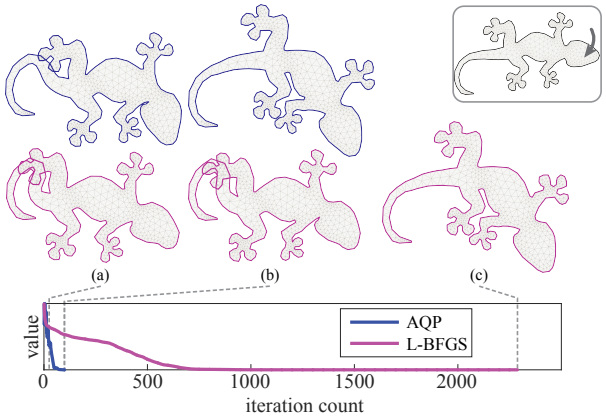


Figure 1: *2D deformation computed by minimizing the isometric distortion energy,* f_{ISO}. *Our method (AQP) converges in just* 105 *iterations within* 0.11 *seconds. Top row shows an intermediate iteration of our method (a) and its final result (b). Bottom row shows three iterations of an L-BFGS solver, which requires about* 2300 *iterations to converge. The rest-pose is shown at the top right.*

The main pitfall encountered in the optimization of geometric energies over meshes is long convergence time. In second-order algorithms, such as Newton's algorithm, it is caused by the need to compute and solve a linear system involving the full Hessian at each iteration; in fact, second-order methods quickly become intractable as problem size increases. Much more scalable are first-order methods such as Gradient Descent (GD) and L-BFGS, wherein each iteration uses only the energy and gradient evaluated at the current and previous iterations. Although a single iteration is very efficient computationally, convergence can still be slow. This often happens as a result of ill-conditioning of the problem, leading an off-the-shelf optimizer into making small steps at each iteration. This problem intensifies as the mesh size increases, and in turn standard optimization techniques often become excruciatingly slow and practically halt when optimizing large scale geometric problems.

Our first-order algorithm uses only objective (value) and gradient information of the previous two iterations to determine its next step,

$$\mathbf{x}_n = \mathcal{A}_\theta\left(\mathbf{x}_{n-1}, \mathbf{x}_{n-2}\right). \tag{1}$$

This is similar to GD, which uses a single previous iteration, and L-BFGS, which allocates memory for previous iterations according to available resources.

We make two observations that enable us to devise an efficient algorithm with improved convergence rate: first, geometric problems commonly suffer from ill-conditioning dominated by a Laplacian-like term in their energy. The effect of this term can be substantially reduced by locally approximating the energy with a convex *quadratic proxy* function whose Hessian is chosen to be the Laplacian. This quadratic proxy can then be efficiently minimized, by utilizing the fact that the Laplacian is a sparse constant matrix. Therefore, each iteration only requires an efficient back-substitution with sparse precomputed factors.

Second, the convergence rate of the algorithm can be significantly improved by correctly balancing how it uses the information from its two last iterations; this balance comes in the form of a parameter θ in algorithm (1). This point of view is exactly the one advocated

in acceleration (momentum) techniques, such as Nesterov's Accelerated Gradient Descent (AGD) [Nesterov 1983; Beck and Teboulle 2009], and is adapted to fit our quadratic proxy framework.

These two observations in fact support each other. The convergence analysis we perform suggests that the optimal acceleration parameter θ should be set according to the condition number associated with the problem, which is problem dependent and typically unknown. Nonetheless, the preconditioning effect achieved by using our quadratic proxy leads to a nearly constant condition number. Consequently, it enables setting this parameter in an almost universal way, that works well for different geometric energies and extremely different mesh sizes, without the need to carefully tune.

We tested the performance of our algorithm on a variety of deformation and parameterization energies and on meshes of varying sizes, overall observing a considerable speedup over standard, state-of-the-art optimization techniques; for example, Figure 1 demonstrates a speed-up of $\times 200$ in optimizing an isometric distortion energy in comparison to L-BFGS solver. Furthermore, our algorithm is nearly scale-independent: the iteration count required for convergence grows slowly (or remarkably remains constant for certain problems) as mesh sizes increase.

2 Approach

Our goal is to devise an efficient algorithm for solving geometric optimization problems taking the following form

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) \tag{2a}$$

$$\text{s.t.} \quad A\mathbf{x} = \mathbf{b} \tag{2b}$$

We see the variable $\mathbf{x} \in \mathbb{R}^{dn}$ as representing the target d-dimensional locations of n vertices of a triangular or tetrahedral mesh; namely, $\mathbf{x} = \text{vec}(\mathbf{X})$ is the column stack of the vertex location matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$. The linear equality constraints (2b) commonly represent positional constraints.

A key step of our approach is to decompose the energy functional into

$$f(\mathbf{x}) = h(\mathbf{x}) + g(\mathbf{x}), \tag{3}$$

where h is a quadratic form $h(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T H\mathbf{x}$ that is *strictly convex subject to the constraint* $A\mathbf{x} = \mathbf{b}$; namely, the matrix $H \in \mathbb{R}^{dn\times dn}$ is strictly positive definite when restricted to the null-space of the full-rank constraints matrix A.

In fact, many interesting problems in geometry processing naturally admit such decomposition, with a quadratic term h that corresponds to the discrete Laplacian matrix L. For concreteness, we demonstrate the proposed optimization framework on several useful and popular energies: As-Rigid-As-Possible (ARAP) [Sorkine and Alexa 2007; Liu et al. 2008; Chao et al. 2010], isometric distortion (ISO) [Smith and Schaefer 2015], and conformal distortion (CONF) [Aigerman et al. 2015]. We decompose those as follows (see Appendix A for additional details):

$$f_{\text{ARAP}}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T H\mathbf{x} - \sum_j \|T_j\|_* |t_j| + c_0 \tag{4}$$

$$f_{\text{ISO}}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \frac{1}{2}\sum_j \left\|T_j^{-1}\right\|_F^2 |t_j| \tag{5}$$

$$f_{\text{CONF}}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \frac{1}{2}\sum_j \|T_j\|_F^2 \left(\frac{1}{\sigma_d(T_j)^2} - 1\right) |t_j| \tag{6}$$

where $H = L \otimes I_d$ is a Kronecker product composed of copies of the (positive semidefinite) Laplacian matrix acting on each coordinate; $\|\cdot\|_F$ is the Frobenius norm; $\|\cdot\|_*$ is the nuclear norm, *i.e.*, the sum of (signed) singular values; $T_j = T_j(\mathbf{x})$ is the differential matrix of the j-th mesh element with respect to $\mathbf{x}$; c_0 is a constant; and $\sigma_d(T_j)$ is the smallest (signed) singular value of $T_j(\mathbf{x})$. f_{ARAP} and f_{ISO} are defined for triangular as well as tetrahedral meshes, while the decomposition (6) for f_{CONF} holds only for the case of triangular meshes.

Algorithm 1: Accelerated Quadratic Proxy (AQP)

Data: feasible initialization $\mathbf{x}$; parameter η

$\mathbf{x}_{-1} = \mathbf{x}_0 = \mathbf{x}$;

$\theta = \frac{1-\sqrt{1/\eta}}{1+\sqrt{1/\eta}}$;

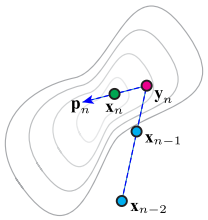
while *not converged* **do**

```
/* Acceleration                                          */
```
$\mathbf{y}_n = (1+\theta)\mathbf{x}_{n-1} - \theta\mathbf{x}_{n-2}$;

```
/* Quadratic proxy minimization                          */
```
$\mathbf{p}_n = \arg\min_{\mathbf{p}} \; h(\mathbf{y}_n + \mathbf{p}) + g(\mathbf{y}_n) + \nabla g(\mathbf{y}_n)\mathbf{p}$ s.t. $A\mathbf{p} = 0$

```
/* Line search                                           */
```
$\mathbf{x}_n = \text{linesearch}_{\,0<t\leq 1} \; f(\mathbf{y}_n + t\mathbf{p}_n)$

Algorithm. Our approach for solving problem (2) makes use of this particular decomposition of f to devise an efficient first-order optimization method that enjoys favorable convergence and scalability properties. The proposed Accelerated Quadratic Proxy (AQP) algorithm is very simple and is described in Algorithm 1. It is an iterative algorithm producing a sequence of approximations $\mathbf{x}_n$ to the optimal point $\mathbf{x}_*$, wherein each iteration comprises three intermediate steps: acceleration ($\mathbf{y}_n$), quadratic proxy minimization ($\mathbf{p}_n$), and a line search from $\mathbf{y}_n$ at the direction $\mathbf{p}_n$, producing the next approximation $\mathbf{x}_n$; see the inset for an illustration.



Next, we describe each of the steps of the AQP algorithm. Thereafter, in section 3 we provide background on related optimization approaches, and in section 4 an analysis of the algorithm, further motivating its specific design.

Acceleration. The acceleration step takes an affine combination of the two previous iterations with a constant $\theta > 0$ to produce an intermediate guess $\mathbf{y}_n$. The value of θ is defined in terms of a parameter $\eta > 0$, whose role and selection are to be discussed in Section 4.

Quadratic proxy minimization. A surrogate convex quadratic function is minimized to provide a descent direction $\mathbf{p}_n$ with respect to $\mathbf{y}_n$. This is done by solving the following linear KKT system,

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_n \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\nabla f(\mathbf{y}_n) \\ 0 \end{bmatrix}. \tag{7}$$

Note that the left-hand side of this linear system is invertible, as H is strictly positive definite on the null-space of A. Moreover, it is constant and thus can be prefactorized once in preprocessing (*e.g.*, using an LU decomposition). The solution of (7), therefore, boils down to an efficient back-substitution, typically with highly sparse factors (as is the Laplacian matrix).

Line search: Lastly, line search is used to produce the next iteration $\mathbf{x}_n$. It searches for $0 < t \leq 1$ such that $\mathbf{x}_n = \mathbf{y}_n + t\mathbf{p}_n$ sufficiently reduces the energy. In our implementation we use a standard backtracking algorithm (Algorithm 3.1 in [Nocedal and Wright 2006]).

Note that: (1) the algorithm presented above only requires evaluating the energy value f and gradient ∇f. Thus, it may be used for a variety of geometric energies other than (4)-(6), even without an explicit decomposition; and (2) for barrier-type energies feasibility is guaranteed by restricting the step size as detailed in Section 5.3.

3 Related work

Optimization. Algorithm 1 falls into the scope of the highly active field of first order optimization methods. Next, we discuss optimization approaches that are related to the proposed algorithm.

Newton methods. Newton's method is one of the most well-known and effective optimization techniques [Nocedal and Wright 2006]. In the context of linearly constrained optimization of the form (2), Newton's method relies on the iterative solution of a linear KKT system of the form

$$\begin{bmatrix} \nabla^2 f(\mathbf{y}_n) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_n \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\nabla f(\mathbf{y}_n) \\ 0 \end{bmatrix}.$$

Consequently, its main drawback is that each iteration requires computing and storing the Hessian as well as solving a linear system with varying left-hand side (unlike in our system, Equation (7)), both may become computationally prohibitive as the problem size increases. Figure 2 demonstrates the performance of Newton's method for the minimization of the isometric distortion energy f_{ISO} on a refined version of the example shown in Figure 1.

Quasi-Newton algorithms, such as BFGS and its limited memory version, L-BFGS, provide an alternative to Newton's method that reduces the computational costs and/or memory footprint but share many of its advantages [Nocedal and Wright 2006]. Although each quasi-Newton iteration is typically less effective than Newton's, its computational efficiency made it a popular choice for generic nonlinear optimization. We consider L-BFGS as a baseline algorithm and compare to it extensively throughout the paper.

Proximal methods. More recently, proximal algorithms have become increasingly popular for both convex and non-convex optimization, see [Combettes and Pesquet 2011] and [Parikh and Boyd 2014] for comprehensive surveys. The key component of these methods is the proximal mapping operator, which can be seen as a generalization of the set projection operator to functions. Proximal splitting methods, like our algorithm, take advantage of a decomposition of the functional $f = h + g$; wherein typically g is smooth and h is proximable, *i.e.*, its proximal map can be efficiently computed. For our decomposition (3), with $h(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T H\mathbf{x}$, a step of the proximal gradient method, also known as the proximal forward-backward algorithm, boils down to solving the following linear system

$$\begin{bmatrix} H + \frac{1}{t} I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_n \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\nabla f(\mathbf{y}_n) \\ 0 \end{bmatrix}, \tag{8}$$

where t is a proximal parameter related to step size. Despite its resemblance to our quadratic proxy step, Equation (7), there are several substantial differences worth noting: The proximal gradient system, Equation (8), depends on the parameter t and thus cannot be trivially prefactorized as (7) in our algorithm. Moreover, proximal gradient with line search typically requires computing additional solutions of the linear system (8), one for each value of t to be tested in backtracking [Beck and Teboulle 2009; Parikh and Boyd 2014; Ochs et al. 2014]. A few approaches, *e.g.*, [Lee et al. 2012], mitigate this requirement by performing a linear line search after solving the above linear system. Our algorithm solves the *prefactorized* system (7) only once per iteration, then requires only simple energy evaluations in its line search. Lastly, for small values of the proximal parameter t, Equation (8) essentially computes a gradient descent step, potentially leading to an inferior convergence behavior. As Figure 2 demonstrates, these differences may result in deterioration of performance in comparison to Algorithm 1.

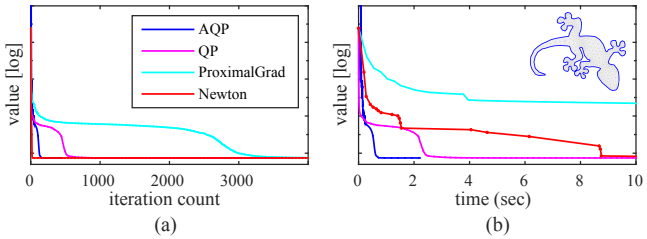


Figure 2: *Related optimization methods. Comparing our algorithm (AQP), its non-accelerated version (QP), the proximal gradient method (ProximalGrad) and Newton's method. Newton is most effective in terms of iteration count, as seen in (a); however, its iterations are inefficient as seen in (b). Our first order approach is both effective and efficient.*

Preconditioning. Our choice of using a fixed quadratic proxy, leading to Equation (7) can be interpreted as choosing the Laplacian as a problem-dependent preconditioner for geometric energies over meshes. Preconditioning is a well known in scientific computing [Saad and Van Der Vorst 2000], and related ideas have appeared, for example, in computational physics [Farago and Karatson 2008; Tuckerman 2015] and simulation [Baraff and Witkin 1998; Wardetzky et al. 2007; Liu et al. 2013]. However, to the best of our knowledge, Laplacian preconditioning has not been previously employed for optimization in geometry processing.

Acceleration. The idea of using an affine combination of current and previous iterations to achieve acceleration dates back to Polyak [Polyak 1964] and Nesterov [Nesterov 1983]. Their acceleration techniques were later generalized to proximal convex optimization [Beck and Teboulle 2009] as well as non-convex optimization [Ochs et al. 2014; Li and Lin 2015]. The convergence analysis we present in Section 4 suggests that the acceleration and quadratic proxy minimization steps of Algorithm 1 are not decoupled and in fact support one another; the effect of preconditioning introduced by our quadratic proxy enables setting the acceleration parameter in an almost universal way.

Geometric optimization. Optimization of geometric energies is a central theme in computer graphics in general and geometry processing in particular.

Deformation. Deformation algorithms often model the deformation problem as an energy minimization problem; the energy could be physically [Terzopoulos and Fleischer 1988; Grinspun et al. 2003] or geometrically inspired [Botsch and Sorkine 2008]. As-Rigid-As-Possible (ARAP) approaches [Sorkine and Alexa 2007; Chao et al. 2010] minimize the local deviation from rigidity. They are often optimized using the global-local interleaving procedure [Liu et al. 2008] or by using higher order methods, such as Newton [Chao et al. 2010] or Gauss-Newton [Huang et al. 2009]. The latter, higher-order methods, solve a different linear system in each iteration and therefore scale poorly with problem size. Minimizing the ARAP energy does not prevent element flips or degeneration. This has been partly mitigated with the introduction of energy barriers [Schüller et al. 2013] as well as per-element constraints [Kovalsky et al. 2014]; these approaches, however, employ computationally demanding Newton-based interior point methods. Practical deformation algorithms often use multigrid, hierarchical structure [Botsch et al. 2006] or a subspace to reduce the number of degrees of freedom [Huang et al. 2006; Ben-Chen et al. 2009; Hildebrandt et al. 2011; Wang et al. 2015].

Parameterization. Similarly to deformations, Parameterization algorithms aim at minimizing an energy measuring the distortion of the mapping [Floater and Hormann 2005; Sheffer et al. 2006]. Parameterization algorithms differ in the energy they attempt to minimize as well as in algorithm they use for its optimization. As linear energies [Lévy et al. 2002; Desbrun et al. 2002] generally cannot avoid flipped elements, many papers focus on non-linear energies [Degener et al. 2003]; MIPS [Hormann and Greiner 2000] uses a non-linear energy that explodes for flipped elements and optimizes one vertex at a time; [Fu et al. 2015] improve this optimization by simultaneously moving groups of vertices using block gradient descent; [Smith and Schaefer 2015] suggest using L-BFGS incorporated with a restricted line search to avoid flipped elements during the optimization.

Relation to global-local. It is insightful to note that the non-accelerated version of Algorithm 1 (QP) reduces to the global-local algorithm [Liu et al. 2008] for the case of the As-Rigid-As-Possible energy (4). This follows by noting that plugging the gradient ∇f_{ARAP} into Equation (7) reduces to the "global" step in the global-local algorithm, where the "local" step is incorporated in the computation of ∇f_{ARAP}; full details are provided in Appendix B.

4 Algorithm analysis

In this section we provide a local analysis to Algorithm 1, explaining its favorable convergence properties; specifically, we explain the roles of the acceleration and the quadratic proxy.

For the analysis of Algorithm 1, let $\mathbf{x}_*$ denote a strict local minimum of $f(\mathbf{x})$ satisfying $A\mathbf{x}_* = \mathbf{b}$, and set $\mathbf{e}_n = \mathbf{x}_n - \mathbf{x}_*$ to be the error vector between $\mathbf{x}_n$, the state of the algorithm at its n-th iteration, and the optimal solution. Analyzing the convergence rate of the algorithm amounts to bounding the error size $\|\mathbf{e}_n\|$. As we derive next, this can be done by showing that the error sequence $\{\mathbf{e}_n\}$ satisfies a certain recurrence relation.

We begin by writing the second order expansion of $f(\mathbf{x})$ at $\mathbf{x}_*$ in the following form

$$f(\mathbf{x}) = \overbrace{\frac{1}{2}\mathbf{x}^T H\mathbf{x}}^{h(\mathbf{x})} + \overbrace{\frac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{a}^T\mathbf{x} + \mathbf{d} + \boldsymbol{\varepsilon}}^{g(\mathbf{x})}, \tag{9}$$

where $G = \nabla^2 g(\mathbf{x}_*)$, and $\boldsymbol{\varepsilon} = \mathcal{O}(\|\mathbf{x} - \mathbf{x}_*\|^3)$. In what follows, we make the assumption that $\boldsymbol{\varepsilon}$ is negligible. This assumption simplifies our analysis, and still well captures the behavior of the algorithm in the vicinity of the strict local minima $\mathbf{x}_*$; this stems from the observation that every sufficiently smooth function can be well approximated by convex quadratic polynomials in the vicinity of its strict local minima. The analysis for the general case follows similar principles but remains outside the scope of this work.

Let K be a matrix whose columns form an orthonormal basis to the null-space of A. Under the above assumptions we have the following lemma, characterizing the error sequence in the vicinity of the optimal point $\mathbf{x}_*$:

Lemma 1. *The error sequence $\boldsymbol{e}_n = \boldsymbol{x}_n - \boldsymbol{x}_*$ produced by Algorithm 1 satisfies the following recurrence relation:*

$$K^T\boldsymbol{e}_n = M\left[(1+\theta)K^T\boldsymbol{e}_{n-1} - \theta K^T\boldsymbol{e}_{n-2}\right], \tag{10}$$

where $\theta > 0$ and M is the iteration matrix:

$$M = I - tQ \quad ; \quad Q = (K^T HK)^{-1}(K^T(H+G)K). \tag{11}$$

The proof of Lemma 1 is given in Appendix C. Lemma 1 implies that the local convergence properties of Algorithm 1 depend on two factors: (1) the value of the scalar θ; and (2) the spectral radius of the matrix M, *i.e.*, the maximal magnitude of its eigenvalues. These, in fact, correspond to two of the key steps of Algorithm 1: the acceleration and the quadratic proxy minimization. We will next discuss the role of these steps in reducing the error sequence.

4.1 Acceleration

We begin with discussing the generalization of the notion of acceleration [Nesterov 1983] to our novel quadratic proxy setup. The following lemma, proved in Appendix C (in similar spirit to [Polyak 1964]), provides intuition on how to choose the parameter θ in (10) to achieve optimal convergence rate (under the assumptions specified above):

Lemma 2. *Let $M \in \mathbb{R}^{p\times p}$ be a diagonalizable matrix, not necessarily symmetric, with positive eigenvalues and spectral radius $\rho = \rho(M) < 1$. Let $\boldsymbol{z}_n$ be a series defined by the recurrence relation*

$$\boldsymbol{z}_n = M\left[(1+\theta)\boldsymbol{z}_{n-1} - \theta\boldsymbol{z}_{n-2}\right]. \tag{12}$$

Then, the series $\boldsymbol{z}_n$ satisfies

1. $\|\boldsymbol{z}_n\| \le c_1\rho^n$, *for* $\theta = 0$.
2. $\|\boldsymbol{z}_n\| \le c_2(1-\sqrt{1-\rho})^n$, *for* $\theta = \theta_{acc} = \frac{2}{\rho}\left(1-\sqrt{1-\rho}\right)-1$

where $c_1, c_2 > 0$ are constants. Furthermore, the latter provides an optimal choice $\theta = \theta_{acc}$ leading to an optimal convergence rate.

This lemma is concerned with a one-parameter (θ) family of recurrence relations. It identifies two special choices of θ: For $\theta = 0$ (*i.e.*, without acceleration) the convergence rate of the series $\mathbf{z}_n$ to zero is ρ, the spectral radius of the iteration matrix M. However, the lemma also asserts that with *the same iteration matrix M* there is a *better* choice of θ: for $\theta_{acc} = \frac{2}{\rho}\left(1-\sqrt{1-\rho}\right) - 1$ the convergence rate is $1-\sqrt{1-\rho}$, which is considerably smaller (*i.e.*, better) than ρ.

According to (10) the error sequence of our algorithm is exactly of the form (12) with $\mathbf{z}_n = K^T\mathbf{e}_n$. Therefore, with the choice θ_{acc}, the error sequence of Algorithm 1 satisfies

$$\|\mathbf{e}_n\| = \|K^T\mathbf{e}_n\| \le c(1-\sqrt{1-\rho})^n, \tag{13}$$

where $c > 0$ is a constant and ρ is the spectral radius of the iteration matrix M in (11). The first equality is due to the fact that $\mathbf{e}_n \in \ker A$ and the columns of K form an orthonormal basis to $\ker A$.

4.2 Quadratic proxy

The convergence rate of the algorithm depends on the spectral radius ρ of the iteration matrix M; as can be observed from (13) – faster convergence is attained for smaller ρ.

By our assumption, $\mathbf{x}_*$ is a *strict* local minimum and therefore $H + G \succ 0$. In turn, Q as defined in (11) is diagonalizable with positive eigenvalues; let $\lambda_1 \ge \ldots \ge \lambda_p > 0$ denote its eigenvalues. The iteration matrix M is therefore also diagonalizable with real spectrum contained within $[1 - t\lambda_1, 1 - t\lambda_p] \subset (-\infty, 1)$. Choosing a constant step size $t = \lambda_1^{-1}$ yields an iteration matrix M that satisfies the conditions required in Lemma 2; in particular, its spectral radius is $\rho = 1 - \kappa^{-1}$, where $\kappa = \kappa(Q) = \lambda_1/\lambda_p$ is the *condition number* of Q. Hence, we can summarize with the following lemma:

Lemma 3. *There exists a constant step size t for which the spectral radius of the iteration matrix M in* (10) *is $\rho(M) = 1 - \kappa^{-1}$, with $\kappa = \kappa(Q)$ the condition number of Q.*

Plugging $\rho = \rho(M)$ into the expression of θ_{acc} of Lemma 2 yields

$$\theta_{acc} = \frac{2}{\rho}\left(1 - \sqrt{1-\rho}\right) - 1 = \frac{1-\sqrt{\kappa^{-1}}}{1+\sqrt{\kappa^{-1}}},$$

which is exactly the choice of θ used in Algorithm 1. Moreover, note the this simple derivation recovers an acceleration coefficient closely related to coefficient sequence used in Nesterov accelerated methods (*e.g.*, [Nesterov 1983; Beck and Teboulle 2009]).

The following theorem summarizes our convergence result, under the assumption of a convex quadratic approximation at the vicinity of a strict local minimum:

Theorem 1. *With the parameter choice $\eta = \kappa(Q)$, Algorithm 1 has an error sequence $\{\boldsymbol{e}_n\}$ that decays according to*

$$\|\boldsymbol{e}_n\| \leq c(1 - \sqrt{\kappa^{-1}})^n.$$

This theorem implies faster convergence is to be expected if Q is better conditioned. At the same time, Equation (11) suggests that the effect of the quadratic proxy $h(\mathbf{x})$ is that of using H as a preconditioner for Q.

This observation motivates our choice $h(\mathbf{x}) = \mathbf{x}^T (L \otimes I_d)\, \mathbf{x}$ for geometric functionals; the Laplacian L typically has a condition number of scale δ^{-2} when discretized over a grid of element size δ [Iserles 2009]. This suggests that for large meshes cancelling the Dirichlet part of the energy has potential for considerably improving the condition number of the matrix Q, and hence convergence.

5 Evaluation

In this section we empirically study the specific choices made in the design of Algorithm 1 as well as demonstrate its performance in comparison with baseline solvers.

5.1 The role of acceleration and quadratic proxy

Two of the principal steps of the AQP algorithm are the acceleration step and the quadratic proxy minimization step. In a previous section we have argued that these step support each other. In order to demonstrate this, we have evaluated the algorithm performance when each of these steps is skipped. Figure 3(a) shows the number of iterations required for minimizing the isometric distortion energy, f_{ISO}, as a function of problem size. Our algorithm is compared to Quadratic Proxy (QP) in which the acceleration is disabled, *i.e.*, $\theta = 0$; and Accelerated Gradient Descent (AGD) which amounts to disabling our Laplacian-based quadratic proxy, *i.e.*, set $H = I$ in Eq. (7). Clearly, AQP outperforms both AGD and QP.

To further demonstrate the role of the quadratic proxy as a preconditioner for Q, we have (numerically) estimated its condition number. Figure 3(b) compares the condition number of $Q = (K^T H K)^{-1}(K^T(H+G)K)$, as defined by Equation (11), to that of $Q_{\mathrm{GD}} = K^T(H+G)K$. The latter corresponds to the iteration of a standard gradient descent algorithm; in particular, note that Q_{GD} corresponds to an iteration of Algorithm 1 with $H = I$ in Eq. (7). We have computed Q and Q_{GD} by estimating the Hessian $H+G$ at the vicinity of a local minimum for problems of increasing size. As can be seen in this graph $\kappa(Q_{\mathrm{GD}})$ grows much more rapidly with problem size in comparison to $\kappa(Q)$; thus, suggesting the advantage of using our Laplacian-based quadratic proxy for the solution of large scale problems.

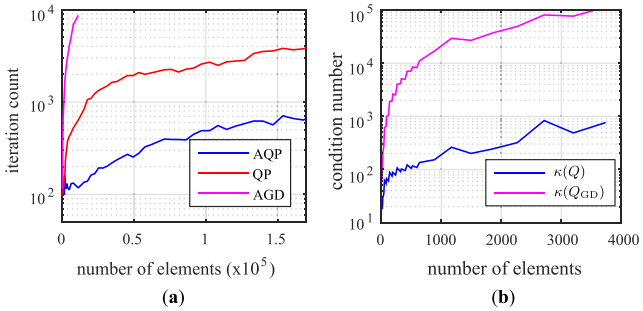


Figure 3: *The role of acceleration and quadratic proxy. (a) shows the iteration count of our approach (AQP) when disabling either the acceleration (QP) or quadratic proxy (AGD); this is in agreement with the analysis, arguing that the acceleration and quadratic proxy minimization steps are intertwined. (b) shows the condition number of Q which dominates the convergence rate of the algorithm; it demonstrates the preconditioning effect of our quadratic proxy, $\kappa(Q)$, compared with the condition number $\kappa(Q_{\mathrm{GD}})$ obtained by a standard gradient descent step.*

5.2 Comparison with standard approaches

Figure 4 compares the performance of Algorithm 1 with that of generic first-order methods. Each experiment compares our algorithm with Accelerated Gradient Descent (AGD) and L-BFGS quasi-Newton algorithms. For comparability, we manually tuned the acceleration parameter of the AGD for each problem, to maximize its performance.

All algorithms were applied on a set of 2- and 3-dimensional problems, for the minimization of the As-Rigid-As-Possible and isometric distortion energies, f_{ARAP} and f_{ISO}, respectively. Each experiment evaluated both the number of iterations and the runtime until convergence; the former indicates the effectiveness of each iteration in reducing the functional, while the latter also takes into account the computational efficiency of each iteration. Instances for which the iteration count exceeded 10^5 were omitted from the evaluation.

Clearly, our AQP algorithm outperforms both the L-BFGS and AGD. Expectedly, the AGD scales poorly with problem size, as the energies to be minimized become severely ill-conditioned; L-BFGS partially addresses this by employing Hessian approximations for the computation of a search direction; our approach, however, explicitly leverages the energy decomposition of Equations (4) and (5) to gain a substantial performance boost. Notably, in many of the examples, the number of iteration AQP requires until convergence grows only moderately with problem size, in some of the cases remaining close to constant.

5.3 Implementation details

In our evaluations and experiments Algorithm 1 was implemented in MATLAB; functional and gradient evaluations were implemented as a sequential single-thread C function; we used an Intel Xeon 2.40GHz CPU. Our algorithm has a single tunable parameter η, which we have set to either 100 or 1000 in all evaluations and experiments. In comparisons with L-BFGS we have used the implementation provided with MATLAB's optimization toolbox. For parameterization we use the implementation provided by [Smith and Schaefer 2015], see Section 6.2 for additional details.

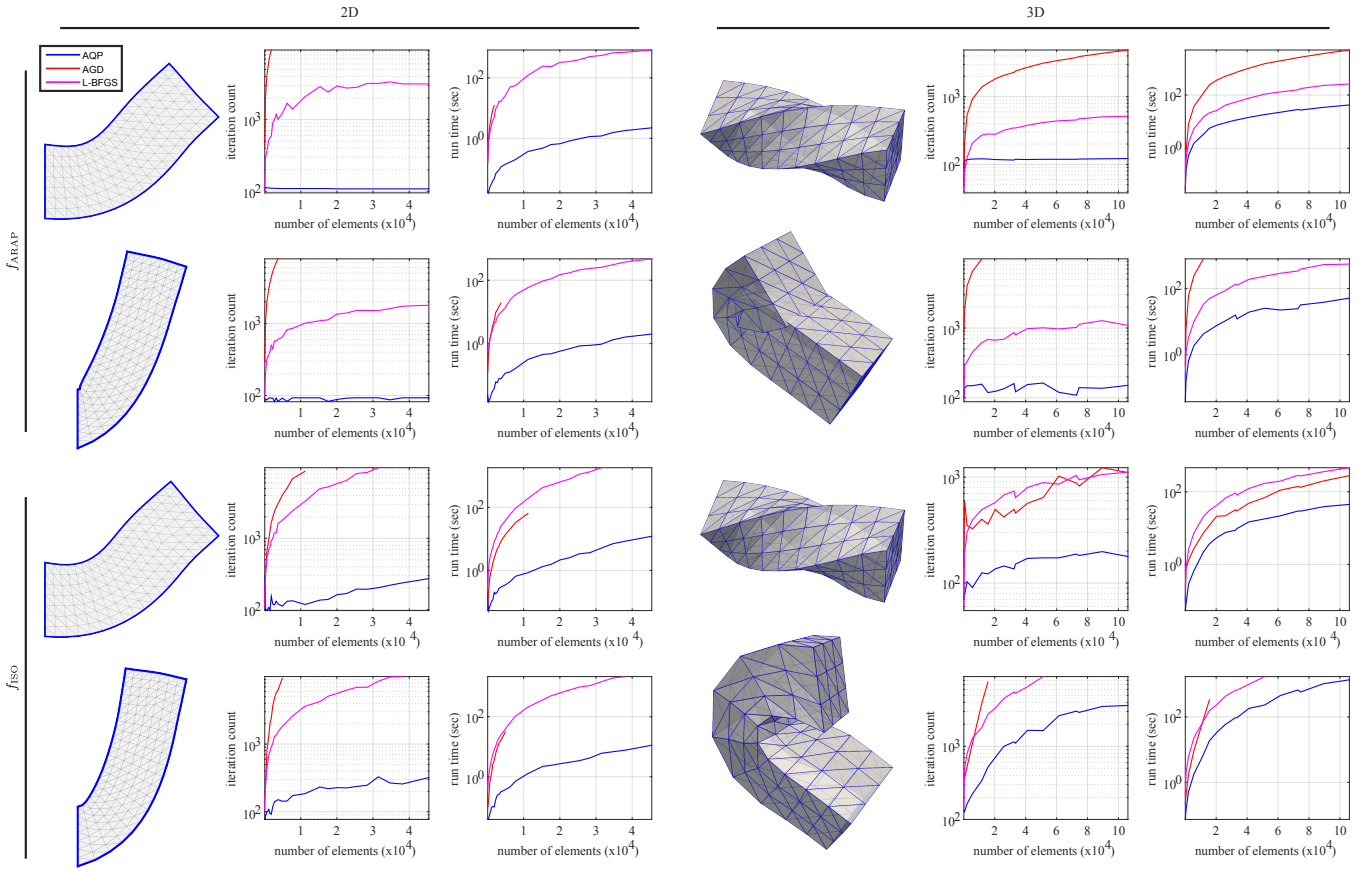


Figure 4: *Comparison with standard first-order methods. Our algorithm (AQP) is compared with Accelerated Gradient Descent (AGD) and L-BFGS on a set of 2- and 3-dimensional problems; the insets show representative instances. Each experiment measures the iteration count and runtime of each solver for various scales of the same problem, indicating their effectiveness and computational efficiency. The AQP algorithm explicitly leverages the underlying geometry of the energies to achieve substantially improved performance.*

For the minimization of the isometric and conformal distortion energies, f_{ISO} and f_{CONF} we have adopted the barrier criterion proposed in [Smith and Schaefer 2015] for determining a maximal feasible step. Namely, let $t_{\max}(\mathbf{z}, \mathbf{u})$ denote the maximal feasible step size at $\mathbf{z}$ in the direction $\mathbf{u}$, as defined in [Smith and Schaefer 2015]. We employ their criterion in two different parts of the Algorithm 1: (1) In the acceleration step we set $\mathbf{y}_n = (1+\theta')\mathbf{x}_{n-1} - \theta'\mathbf{x}_{n-2}$ where $\theta' = \min\left\{\theta, \frac{1}{2}t_{\max}(\mathbf{x}_{n-1}, \mathbf{x}_{n-1} - \mathbf{x}_{n-2})\right\}$; and (2) in the line search part we set the maximal step size to be $\min\left\{1, \frac{1}{2}t_{\max}(\mathbf{y}_n, \mathbf{p}_n)\right\}$. Lastly, using this criterion requires an orientation preserving initialization, which was computed with the code of [Kovalsky et al. 2015].

6 Experiments

6.1 Deformation

In this experiment we show the utility of the proposed approach for computing deformations of triangular and tetrahedral meshes.

2-dimensional deformations. Figure 5 shows the result of minimizing the As-Rigid-As-Possible energy, f_{ARAP}, for the deformation of a triangular mesh comprising 8k triangles. The performance of our approach is compared with that of the local-global algorithm [Liu et al. 2008], popularly used for the minimization of this energy. Our algorithm terminates much faster than the global-local approach, after 0.38 seconds compared to 1.55 seconds. Moreover, a near optimal result, often sufficient for interactive modeling, is achieved even faster, after less than 0.1 second.

A more robust approach (*i.e.*, resisting flips) for computing deformations is to minimize the isometric distortion energy f_{ISO}. Figure 6 shows the result of minimizing the isometric distortion energy for the same problem shown in Figure 5. Our algorithm converges in 0.81 seconds, and its result is guaranteed to be non-degenerate and orientation preserving (see [Smith and Schaefer 2015]). This is compared to 36 seconds it take the L-BFGS to terminate.

Another example of a 2D deformation obtained by minimizing the isometric distortion energy is presented in Figure 1; this example demonstrates a speedup by a factor of 200 over a standard L-BFGS solver. Lastly, as a stress test, Figure 7 presents the solution of the same problem starting with an extreme initialization, demonstrating the robustness of the algorithm.

3-dimensional deformations. Figures 9 and 11 show the result of employing our approach for deforming volumetric tetrahedral meshes, obtained by tetrahedralizing surfaces taken from [Sacht et al. 2015] using TetGen [Si 2015]. The figures present the results of minimizing the As-Rigid-As-Possible energy f_{ARAP} and isometric distortion energy f_{ISO}, respectively. The latter, as before, is guaranteed to produce a non-degenerate and orientation preserving volumetric deformation.

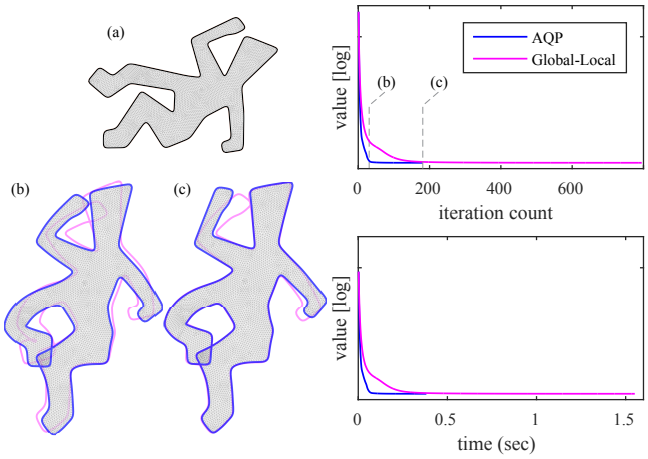


Figure 5: *As-Rigid-As-Possible 2D deformation. The rest pose is shown in (a). (b) and (c) compare intermediate iterations of our approach and an alternating global-local algorithm. In (b) our algorithm almost converges while global-local lags behind. In (c), after* 0.38 *seconds, our algorithm converges while global local requires* 1.55 *seconds to terminate.*

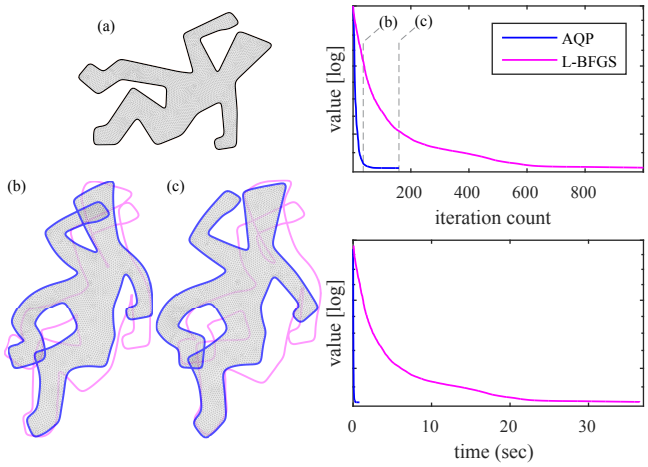


Figure 6: *2D deformation attained by minimizing the isometric distortion energy f_{ISO}. The rest pose is shown in (a). (b) and (c) compare intermediate iterations of our approach and an L-BFGS solver. In (b) our algorithm almost converges while the L-BFGS lags significantly behind. In (c), after* 0.81 *seconds, our algorithm converges while L-BFGS requires over* 36 *seconds to terminate.*

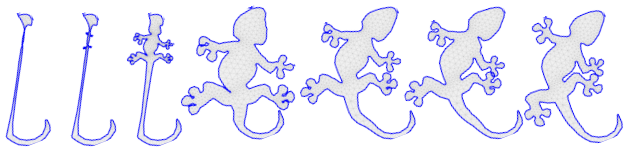


Figure 7: *2D deformation – robustness of the algorithm. Minimization of the isometric distortion energy f_{ISO} subject to an extreme initialization (left). Our algorithm converges (right) after* 174 *iterations within* 0.27 *seconds.*

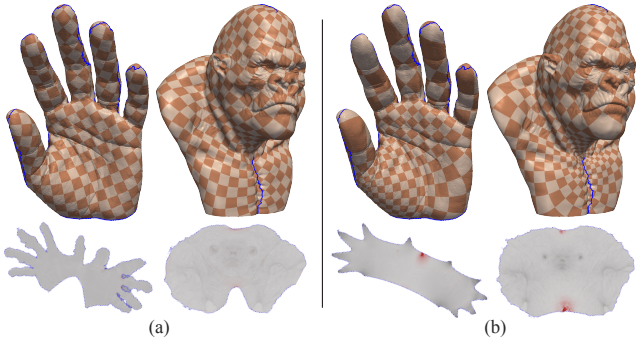


Figure 8: *Parameterization of Hand and Gorilla models, comprising 390k and 200k triangles, respectively. In (a) our algorithm is used to minimize the isometric distortion energy, f_{ISO}, terminating after 333 and 41 seconds, respectively. (b) presents the result of minimizing the conformal distortion, f_{CONF}, obtained in 388 and 105 seconds, respectively. Colors depict energy distributions.*

Interactive computation rates. As demonstrated above, our algorithm enjoys preferable convergence properties; it terminates earlier, both in iteration count and runtime, compared to standard baseline approaches. Moreover, it attains near optimal results in a small number of iterations – in many cases, providing visually pleasing results.

This makes the proposed approach particularly adequate for incorporating into applications requiring interactive computation rates. Figure 10 demonstrates a standard subspace approach for deformation computation [Huang et al. 2006; Wang et al. 2015]: a low resolution volumetric tetrahedral mesh is deformed, in turn inducing a deformation on an encaged high resolution surface. (For this example we used a naive piecewise linear interpolation, *i.e.*, barycentric coordinates, however, any linear or non-linear subspace method can be used instead [Ju et al. 2005; Wang et al. 2015].)

6.2 Parameterization

Our AQP algorithm can be straightforwardly used for computing surface parameterizations. We have experimented with parameterizations computed by minimizing the isometric and conformal distortion energies, f_{ISO} and f_{CONF}, respectively.

Figure 8 demonstrates the parameterization of two high resolution surfaces obtained using our algorithm. Tutte's embedding [Tutte 1963] to the unit disk was used to compute a feasible (bijective) initial parameterization.

In Figure 12 we compare the performance of our approach to the limited-memory BFGS method adopted by [Smith and Schaefer 2015] for the minimization of the isometric distortion energy, f_{ISO}. Both algorithms produce essentially identical results, however, our requires a smaller number of iterations and lower runtime to converge. A significant speedup, in a factor of about $\times 9$, can be observed for the higher resolution (Horse) example.

We use the implementation of Smith and Schaefer [2015]. For comparability with our single thread implementation, we also configure their code to use a single core computation, thus its performance is slightly inferior to that reported in their paper. The same multicore strategy used in their code for the computation of gradients can be straightforwardly adopted for our algorithm, thus achieving similar speedup.

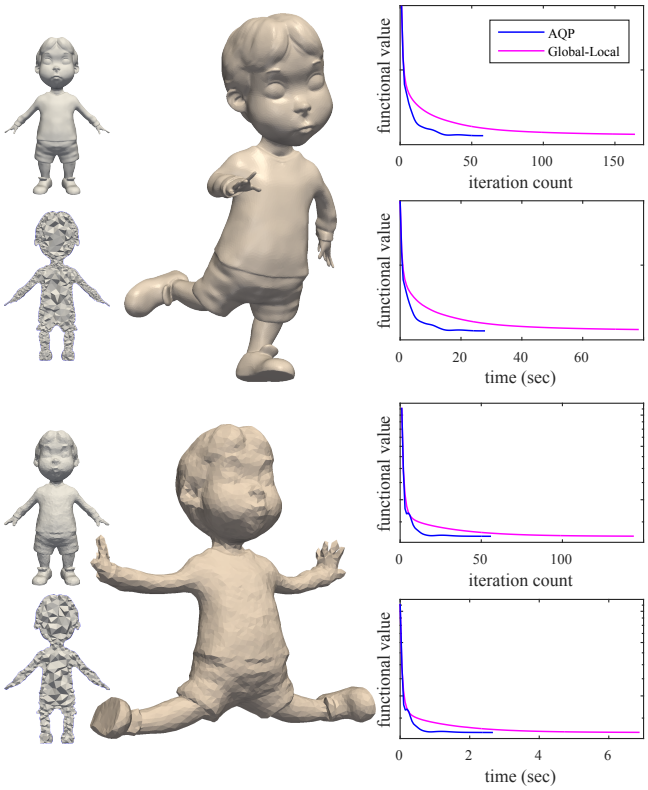


Figure 9: *As-Rigid-As-Possible 3D volumetric deformation of a high-resolution mesh (top – 285k tets) and a low-resolution mesh (bottom – 37k tets). The insets show the rest poses of the tetrahedral meshes. The proposed approach terminates approximately* $3\times$ *faster than the global-local alternating approach; it also achieves near optimal results significantly sooner.*

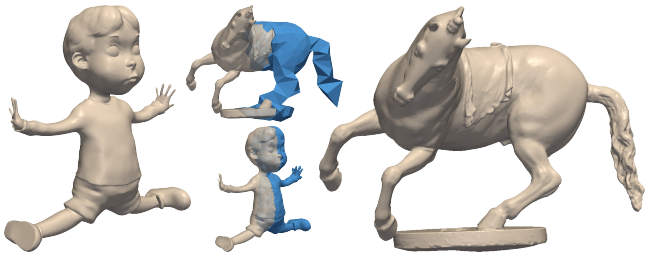


Figure 10: *Deformation subspaces. A low resolution tetrahedral mesh (blue) is deformed (Figures 9 and 11), in turn, inducing a deformation on an encaged high resolution surface. Computation of the boy's deformation (left) is completed in* 2.6 *seconds, by minimizing the As-Rigid-As-Possible energy for a 37k tets control cage. The horse's deformation (right) is computed by minimizing the isometric distortion energy of a control cage comprising 2k tets in* 0.28 *seconds; in this case, the deformation of the encapsulating volume is guaranteed to be non-degenerate and orientation preserving.*

7 Concluding remarks

We presented the Accelerated Quadratic Proxy algorithm - a simple first-order algorithm for optimizing geometric functionals defined over triangular and tetrahedral meshes. Our method utilizes the common structure of optimization problems over meshes to improve iteration efficiency and incorporate acceleration in an almost universal way (*i.e.*, insensitive to different energy types and mesh-sizes).

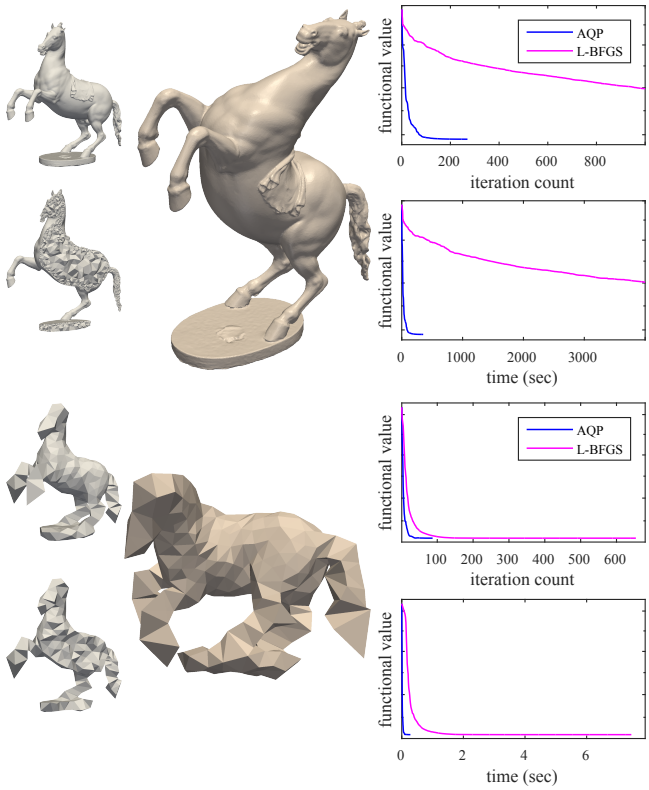


Figure 11: *3D tetrahedral deformation attained by minimizing the isometric distortion energy* f_{ISO}*. Deformation of meshes comprising 630k and 2k tets are shown in the top and bottom, respectively; rest poses shown in the insets. The proposed approach converges significantly faster than L-BFGS. Notably, the resulting deformations are guaranteed to be without flips and non-degenerate.*

Currently, our line-search algorithm is rather naive and often requires many function evaluation to perform a descent step. Incorporating more sophisticated line-search strategy is expected to further improve performance. Improving our current MATLAB implementation is also likely to gain a significant speedup.

We have tested our algorithm on three popular energies; a very interesting future research direction is exploring how the algorithm behaves for other energies. A limitation of our approach is that we do not have a principled way of determining how effective the decomposition $h + g$ is for an arbitrary energy; specifically, how the conditioning of the matrix Q in (11) is improved. Numerical experiments provide a partial answer, however being able to theoretically bound $\kappa(Q)$ would provide a powerful theoretical justification for the algorithm.

Lastly, we would like to apply this algorithm to other applications within graphics and related fields that solve optimization problems on tesselated domains (*e.g.*, images) and even more general graphs (*e.g.*, skeletons, maps).

8 Acknowledgements

This work was supported in part by the European Research Council (ERC starting grant No. 307754 "SurfComp"), the Israel Science Foundation (grant No. 1284/12) and the I-CORE program of the Israel PBC and ISF (Grant No. 4/11). The Hand model is from the Aim@Shape repository and the Gorilla model is from TurboSquid. The authors would like to thank Noam Aigerman, Ehud Galun and the anonymous reviewers for their comments and suggestions.

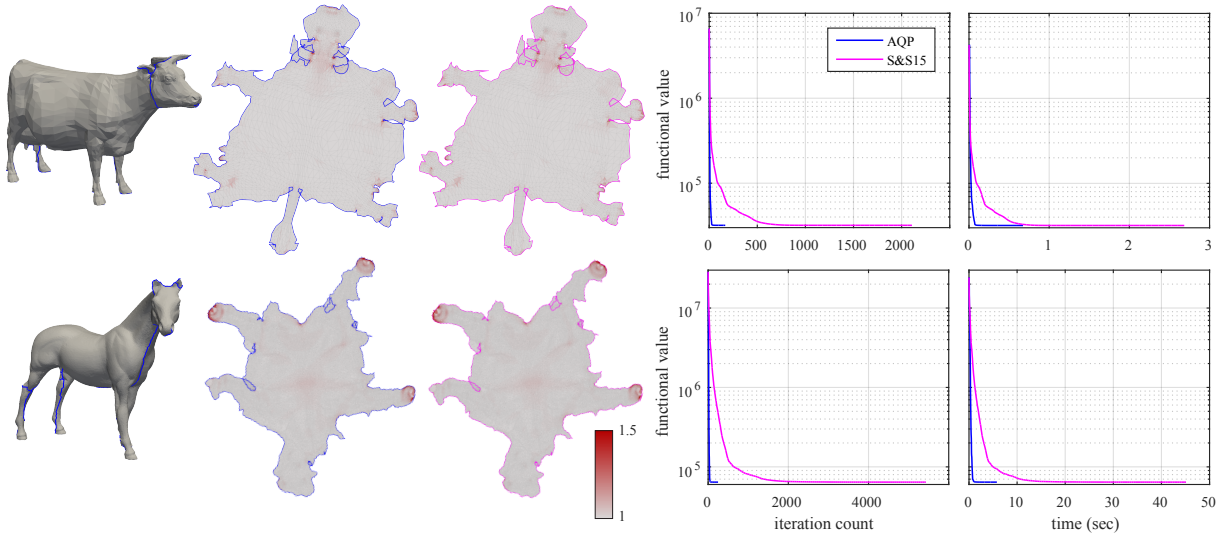


Figure 12: *Parameterization minimizing the isometric distortion energy. Comparing the performance of our approach with that of [Smith and Schaefer 2015] for the minimization of f_{ISO}. In the cow example (6.4k triangles) our algorithm converges in* 0.67 *seconds compared to* 2.68 *in Smith and Schaefer's method. The difference is more significant for the larger horse example (40k triangles), where our algorithm converges in* 5.8 *seconds compared to theirs in* 45.1 *seconds. The resulting parameterizations (ours – left, Smith and Schaefer's – right) are essentially identical. (For comparability we have used a single core implementation of [Smith and Schaefer 2015], see the text for details.)*

References

AIGERMAN, N., PORANNE, R., AND LIPMAN, Y. 2015. Seamless surface mappings. *ACM Transactions on Graphics (TOG) 34*, 4, 72.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, 43–54.

BECK, A., AND TEBOULLE, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences 2*, 1, 183–202.

BEN-CHEN, M., WEBER, O., AND GOTSMAN, C. 2009. Variational harmonic maps for space deformation. In *ACM Transactions on Graphics (TOG)*, vol. 28, ACM, 34.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *Visualization and Computer Graphics, IEEE Transactions on 14*, 1, 213–230.

BOTSCH, M., PAULY, M., GROSS, M. H., AND KOBBELT, L. 2006. Primo: coupled prisms for intuitive surface modeling. In *Symposium on Geometry Processing*, no. EPFL-CONF-149310, 11–20.

CHAO, I., PINKALL, U., SANAN, P., AND SCHRÖDER, P. 2010. A simple geometric model for elastic deformations. In *ACM Transactions on Graphics (TOG)*, vol. 29, ACM, 38.

COMBETTES, P. L., AND PESQUET, J.-C. 2011. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*. Springer, 185–212.

DEGENER, P., MESETH, J., AND KLEIN, R. 2003. An adaptable surface parameterization method. *IMR 3*, 201–213.

DESBRUN, M., MEYER, M., AND ALLIEZ, P. 2002. Intrinsic parameterizations of surface meshes. In *Computer Graphics Forum*, vol. 21, Wiley Online Library, 209–218.

FARAGO, I., AND KARATSON, J. 2008. Sobolev gradient type preconditioning for the saint-venant model of elasto-plastic torsion. *Int. J. Numer. Anal. Model 5*, 2, 206–221.

FLOATER, M. S., AND HORMANN, K. 2005. Surface parameterization: a tutorial and survey. *Advances in multiresolution for geometric modelling 1*, 1.

FU, X.-M., LIU, Y., AND GUO, B. 2015. Computing locally injective mappings by advanced mips. *ACM Transactions on Graphics (TOG) 34*, 4, 71.

GRINSPUN, E., HIRANI, A., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete Shells. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 62–67.

HILDEBRANDT, K., SCHULZ, C., TYCOWICZ, C. V., AND POLTHIER, K. 2011. Interactive surface modeling using modal analysis. *ACM Transactions on Graphics (TOG) 30*, 5, 119.

HORMANN, K., AND GREINER, G. 2000. Mips: An efficient global parametrization method. Tech. rep., DTIC Document.

HUANG, J., SHI, X., LIU, X., ZHOU, K., WEI, L.-Y., TENG, S.-H., BAO, H., GUO, B., AND SHUM, H.-Y. 2006. Subspace gradient domain mesh deformation. *ACM Transactions on Graphics (TOG) 25*, 3, 1126–1134.

HUANG, Q.-X., WICKE, M., ADAMS, B., AND GUIBAS, L. 2009. Shape decomposition using modal analysis. In *Computer Graphics Forum*, vol. 28, Wiley Online Library, 407–416.

ISERLES, A. 2009. *A first course in the numerical analysis of differential equations*. No. 44. Cambridge University Press.

JU, T., SCHAEFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. In *ACM Transactions on Graphics (TOG)*, vol. 24, ACM, 561–566.

KOVALSKY, S. Z., AIGERMAN, N., BASRI, R., AND LIPMAN, Y. 2014. Controlling singular values with semidefinite programming. *ACM Transactions on Graphics 33*, 4, 68.

KOVALSKY, S. Z., AIGERMAN, N., BASRI, R., AND LIPMAN, Y. 2015. Large-scale bounded distortion mappings. *ACM Transactions on Graphics (TOG) 34*, 6, 191.

LEE, J., SUN, Y., AND SAUNDERS, M. 2012. Proximal newton-type methods for convex optimization. In *Advances in Neural Information Processing Systems*, 836–844.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (TOG) 21*, 3, 362–371.

LI, H., AND LIN, Z. 2015. Accelerated proximal gradient methods for nonconvex programming. In *Advances in Neural Information Processing Systems*, 379–387.

LIU, L., ZHANG, L., XU, Y., GOTSMAN, C., AND GORTLER, S. J. 2008. A local/global approach to mesh parameterization. In *Computer Graphics Forum*, vol. 27, Wiley Online Library, 1495–1504.

LIU, T., BARGTEIL, A. W., O'BRIEN, J. F., AND KAVAN, L. 2013. Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG) 32*, 6, 214.

NESTEROV, Y. 1983. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, vol. 27, 372–376.

NOCEDAL, J., AND WRIGHT, S. 2006. *Numerical optimization*. Springer Science & Business Media.

OCHS, P., CHEN, Y., BROX, T., AND POCK, T. 2014. ipiano: Inertial proximal algorithm for nonconvex optimization. *SIAM Journal on Imaging Sciences 7*, 2, 1388–1419.

PAPADOPOULO, T., AND LOURAKIS, M. I. 2000. Estimating the jacobian of the singular value decomposition: Theory and applications. In *Computer Vision-ECCV 2000*. Springer, 554–570.

PARIKH, N., AND BOYD, S. P. 2014. Proximal algorithms. *Foundations and Trends in optimization 1*, 3, 127–239.

PETERSEN, K. B., PEDERSEN, M. S., ET AL. 2008. The matrix cookbook. *Technical University of Denmark 7*, 15.

POLYAK, B. T. 1964. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics 4*, 5, 1–17.

SAAD, Y., AND VAN DER VORST, H. A. 2000. Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics 123*, 1, 1–33.

SACHT, L., VOUGA, E., AND JACOBSON, A. 2015. Nested cages. *ACM Transactions on Graphics (TOG) 34*, 6, 170.

SCHÜLLER, C., KAVAN, L., PANOZZO, D., AND SORKINE-HORNUNG, O. 2013. Locally injective mappings. *Computer Graphics Forum (proceedings of Symposium on Geometry Processing) 32*, 5.

SHEFFER, A., PRAUN, E., AND ROSE, K. 2006. Mesh parameterization methods and their applications. *Foundations and Trends® in Computer Graphics and Vision 2*, 2, 105–171.

SI, H. 2015. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS) 41*, 2, 11.

SMITH, J., AND SCHAEFER, S. 2015. Bijective parameterization with free boundaries. *ACM Trans. Graph. 34*, 4 (July), 70:1–70:9.

SORKINE, O., AND ALEXA, M. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, vol. 4.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *ACM Siggraph Computer Graphics*, vol. 22, ACM, 269–278.

TUCKERMAN, L. S. 2015. Laplacian preconditioning for the inverse arnoldi method. *Communications in Computational Physics 18*, 05, 1336–1351.

TUTTE, W. T. 1963. How to draw a graph. *Proc. London Math. Soc 13*, 3, 743–768.

WANG, Y., JACOBSON, A., BARBIC, J., AND KAVAN, L. 2015. Linear subspace design for real-time shape deformation. *ACM Trans. Graph. 34*, 4.

WARDETZKY, M., BERGOU, M., HARMON, D., ZORIN, D., AND GRINSPUN, E. 2007. Discrete quadratic curvature energies. *Computer Aided Geometric Design 24*, 8, 499–518.

Appendix A Energies

For completeness, we provide additional details on the decompositions and gradients of the energies (4)-(6). First, note that these energies take the form

$$f(\mathbf{x}) = \sum_j E(T_j)\,|t_j|,$$

where

$$\begin{aligned} E_{\text{ARAP}}(T) &= \frac{1}{2}\|T-R\|_F^2, \\ E_{\text{ISO}}(T) &= \frac{1}{2}\left(\|T\|_F^2 + \left\|T^{-1}\right\|_F^2\right), \\ E_{\text{CONF}}(T) &= \frac{1}{2}\left(\sigma_1(T)/\sigma_d(T)\right)^2. \end{aligned}$$

Here, R is the projection of T onto rotations, and $\sigma_k(T)$ denotes the k-th (signed) singular value of the differential T.

Decompositions. Note that, in terms of the singular values of the differentials, the As-Rigid-As-Possible energy takes the following form,

$$f_{\text{ARAP}}(\mathbf{x}) = \frac{1}{2}\sum_j \|T_j - R_j\|_F^2\,|t_j|$$

$$= \frac{1}{2}\sum_{j,k}\left(\sigma_k(T_j)-1\right)^2|t_j| = \frac{1}{2}\sum_{j,k}\left(\sigma_k(T_j)^2 - 2\sigma_k(T_j)+1\right)|t_j|.$$

Decomposition (4) then follows from the definition of the nuclear norm and the observation that

$$\frac{1}{2}\mathbf{x}^T H\mathbf{x} = \frac{1}{2}\sum_j \|T_j\|_F^2\,|t_j| = \frac{1}{2}\sum_{j,k}\sigma_k(T_j)^2|t_j|. \qquad (14)$$

Using (14), the decomposition (5) stems from the matrix form of the isometric distortion,

$$f_{\text{ISO}}(\mathbf{x}) = \frac{1}{2}\sum_j \left(\|T_j\|_F^2 + \left\|T_j^{-1}\right\|_F^2\right)|t_j|$$

The decomposition for the conformal distortion energy, in the case $d=2$, is obtained by simply adding and subtracting (14) from

$$f_{\text{CONF}}(\mathbf{x}) = \frac{1}{2}\sum_j \left(\frac{\sigma_1(T_j)}{\sigma_d(T_j)}\right)^2|t_j|.$$

Gradients. Consider the gradients of these per-element energies with respect to a single differential $T \in \mathbb{R}^{d\times d}$. From [Chao et al. 2010] we have that

$$\nabla E_{\text{ARAP}}(T) = T - R.$$

Using [Petersen et al. 2008] we see that

$$\nabla E_{\text{ISO}}(T) = T - T^{-T}T^{-1}T^{-T}.$$

From [Papadopoulo and Lourakis 2000] we conclude that

$$\nabla E_{\text{CONF}}(T) = \frac{\sigma_1(T)}{\sigma_d(T)}\frac{\sigma_d(T)u_1v_1^T - \sigma_1(T)u_dv_d^T}{\sigma_d(T)^2},$$

where u_k and v_k are the k-th left and right singular vectors of T, respectively.

Then, the chain rule implies that the gradients are given by plugging ∇E_{ARAP}, ∇E_{ISO} or ∇E_{CONF} into

$$\nabla f(\mathbf{x}) = \sum_j J_j^T \text{vec}\left(\nabla E(T_j)\right)|t_j|, \tag{15}$$

where

$$J_j = \frac{d\text{vec}(T_j)}{d\mathbf{x}},$$

is the $d^2 \times dn$ Jacobian matrix satisfying $\text{vec}(T_j(\mathbf{x})) = J_j\mathbf{x}$.

Appendix B Relation to global-local

To show that QP reduces to the global-local algorithm [Liu et al. 2008] for the As-Rigid-As-Possible energy, we note that by (15),

$$-\nabla f_{\text{ARAP}}(\mathbf{x}_{n-1}) = -H\mathbf{x}_{n-1} + \sum_j J_j^T \text{vec}\left(R_j(\mathbf{x}_{n-1})\right)|t_j|,$$

where $R_j(\mathbf{x}_{n-1})$ is the projection of the differential $T_j(\mathbf{x}_{n-1})$ of previous iteration onto rotations. Plugging this into Equation (7) and simple manipulation gives

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix}\begin{bmatrix} \mathbf{x}_n \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \sum_j J_j^T \text{vec}\left(R_j(\mathbf{x}_{n-1})\right)|t_j| \\ b \end{bmatrix},$$

where $\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{p}_n$. In turn, this linear system minimizes

$$\frac{1}{2}\sum_j \|T_j(\mathbf{x}_n) - R_j(\mathbf{x}_{n-1})\|_F^2 |t_j|$$

subject to $A\mathbf{x}_n = b$. Thus, QP with a constant unit step size coincides with the global-local algorithm: each linear solve constitutes the "global" step, whereas the "local" step corresponds to the projection R_j of each differential onto rotations in the gradient computation.

Appendix C Proofs

Proof of Lemma 1. First subtracting $\mathbf{x}_n$ from the optimal solution $\mathbf{x}_*$ leads to

$$\mathbf{e}_n = \mathbf{x}_* - \mathbf{x}_n = \mathbf{x}_* - \mathbf{y}_n - t\mathbf{p}_n$$

and after multiplying from the left with K^T we have

$$K^T\mathbf{e}_n = K^T(\mathbf{x}_* - \mathbf{y}_n) - tK^T\mathbf{p}_n \tag{16}$$

The optimal solution $\mathbf{x}^*$ of $f(\mathbf{x})$ satisfies the following KKT equation:

$$\begin{bmatrix} H+G & A^T \\ A & 0 \end{bmatrix}\begin{bmatrix} \mathbf{x}_* \\ \boldsymbol{\lambda}_* \end{bmatrix} = \begin{bmatrix} -\mathbf{a} \\ \mathbf{b} \end{bmatrix}$$

and the quadratic proxy step of the algorithm satisfies

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix}\begin{bmatrix} \mathbf{p}_n \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{a} - (H+G)\mathbf{y}_n \\ 0 \end{bmatrix}.$$

Subtracting the two equations and rearranging we get

$$H\mathbf{p}_n = (H+G)(\mathbf{x}_* - \mathbf{y}_n) + A^T(\boldsymbol{\lambda}_* - \boldsymbol{\lambda}).$$

Multiplying this equation by K^T from the left and noticing that $K^TA^T = 0$ we get

$$K^TH\mathbf{p}_n = K^T(H+G)(\mathbf{x}_* - \mathbf{y}_n).$$

Since $\mathbf{p}_n, \mathbf{x}_* - \mathbf{y}_n \in \ker A$, and $KK^T\mathbf{u} = \mathbf{u}$ for all $\mathbf{u} \in \ker A$ we have

$$\left(K^THK\right)K^T\mathbf{p}_n = \left(K^T(H+G)K\right)K^T(\mathbf{x}_* - \mathbf{y}_n).$$

Solving for $K^T\mathbf{p}_n$ and plugging in (16) gives:

$$\begin{aligned} K^T\mathbf{e}_n &= MK^T(\mathbf{x}_* - \mathbf{y}_n) \\ &= M\left((1+\theta)K^T\mathbf{e}_{n-1} - \theta K^T\mathbf{e}_{n-2}\right) \end{aligned}$$

where the last equality uses $\mathbf{y}_n = (1+\theta)\mathbf{x}_{n-1} - \theta\mathbf{x}_{n-2}$. □

Proof of Lemma 2. Using the eigen-decomposition $M = UDU^{-1}$ we write (12) as the recurrence relation

$$\mathbf{z}_n = D\left((1+\theta)\mathbf{z}_{n-1} - \theta\mathbf{z}_{n-2}\right),$$

where $\mathbf{z}_n = U^{-1}\mathbf{x}_n$. This gives n decoupled scalar recurrence relations, each is of the form

$$z_n - \lambda(1+\theta)z_{n-1} + \lambda\theta z_{n-2} = 0,$$

where λ is an eigenvalue of M. The solution this recurrence equation satisfies $|z_n| \le cn|\xi|^n$ where ξ is the root of the largest magnitude of the basic polynomial $\xi^2 - \lambda(1+\theta)\xi + \lambda\theta = 0$ given by (using $\theta > 0$ and $\lambda \ge 0$),

$$\xi(\lambda,\theta) = \frac{\lambda(1+\theta) + \sqrt{\lambda^2(1+\theta)^2 - 4\lambda\theta}}{2}.$$

We next bound $|\xi(\lambda,\theta)|$ for all λ. Fixing θ, $\xi_\theta(\lambda) = \xi(\lambda,\theta)$ is real outside the open interval $(0,\nu(\theta))$, where $\nu(\theta) = \frac{4\theta}{(1+\theta)^2}$. For $\lambda \in (0,\nu(\theta))$, $|\xi_\theta(\lambda)| = \sqrt{\lambda\theta}$. Therefore, $|\xi_\theta(\lambda)|$ is monotonically increasing for $\lambda \ge 0$. Therefore, we have that $|\xi(\lambda,\theta)| \le |\xi(\rho,\theta)|$, for all θ, where $\rho = \rho(M)$ is the spectral radius of M.

To assure fastest convergence we therefore would like to minimize the bound $|\xi_\rho(\theta)|$, where $\xi_\rho(\theta) = \xi(\rho,\theta)$. Its minimum will be attained at one of the two points: $\frac{2}{\rho}(1 \pm \sqrt{1-\rho}) - 1$. Simple check shows that the minimum is achieved at $\theta_- = \frac{2}{\rho}(1-\sqrt{1-\rho}) - 1$ and $|\xi| \le \xi_\rho(\theta_-) = 1 - \sqrt{1-\rho}$. □