Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 273732, 9 pages
http://dx.doi.org/10.1155/2014/273732

Research Article

An Efficient Approach for Solving Mesh Optimization Problems Using Newton's Method

Jibum Kim

Department of Computer Science and Engineering, Incheon National University, Incheon 406-772, Republic of Korea

Correspondence should be addressed to Jibum Kim; jibumkim@incheon.ac.kr

Received 22 May 2014; Revised 12 September 2014; Accepted 6 October 2014; Published 21 October 2014

Academic Editor: Tiedong Ma

We present an efficient approach for solving various mesh optimization problems. Our approach is based on Newton's method, which uses both first-order (gradient) and second-order (Hessian) derivatives of the nonlinear objective function. The volume and surface mesh optimization algorithms are developed such that mesh validity and surface constraints are satisfied. We also propose several Hessian modification methods when the Hessian matrix is not positive definite. We demonstrate our approach by comparing our method with nonlinear conjugate gradient and steepest descent methods in terms of both efficiency and mesh quality.

1. Introduction

Mesh quality improvement and mesh untangling are important topics for partial differential equation- (PDE-) based simulations, because elements of poor quality can be detrimental to accuracy and efficiency of the solution, and an inverted (tangled) element can result in the failure to obtain a PDE solution [1, 2]. Mesh quality improvement (also, mesh untangling) problems are often formulated as nonlinear objective functions [2–5]. Improvement of mesh quality is most often done by minimization of the nonlinear objective function over the whole mesh based on the quality of individual element in the mesh. In order to efficiently minimize the nonlinear objective function, various nonlinear solvers have been developed. These solvers include steepest descent, conjugate gradient, inexact Newton, feasible Newton, quasi Newton, and trust-region methods [6–8]. These solvers can be roughly divided into two groups: gradient-based methods and Hessian-based methods. On one hand, gradient-based methods require less computation and memory than Hessian-based methods, since they do not require Hessian computation. Hessian-based methods on the other hand enjoy the use of more information at the cost of more computation.

The performance of gradient and Hessian-based methods for solving volume mesh optimization problems has been investigated in [9]. Freitag Diachin et al. [10] compared inexact Newton and the block coordinate descent method for solving volume mesh optimization problems. The authors report that the block coordinate descent method is more efficient in achieving a suboptimal solution than the inexact Newton method, because the inexact Newton method incurs high setup costs. The inexact Newton method outperforms the block coordinate method when higher accuracy is desired. The block coordinate descent studied in [10] is basically identical to Newton's method, which uses the Newton direction as a line search direction. However, it focuses on volume mesh quality improvement problems, which do not include any inverted elements and do not include surface mesh optimization problems. Many researchers have used nonlinear conjugate gradient method (NLCG) for solving mesh optimization problems for various applications [1, 2, 5, 11, 12]. However, it turns out that the NLCG method is slow, prone to getting stuck in local minima, and often fails to converge, especially for surface meshes due to surface mesh constraints.

In this paper, we employ Newton's method for solving various nonlinear mesh optimization problems. We extend our preliminary results in [13] and present a comparison between Newton's method and gradient-based methods. The use of Newton's method for solving nonlinear optimization problem is motivated by the observation that if the nonlinear

functional is sufficiently smooth, its optimal points are roots of the derivative function. Thus, a nonlinear optimization problem is transformed into a root finding problem. We choose Newton's method for solving the mesh optimization problem, because it provides both search direction and a step size using both first-order (gradient) and second-order (Hessian) information. Therefore, for many cases, Newton's method enjoys a full step length. In addition, close to the solution, its convergence is typically quadratic.

There might exist several potential issues when Newton's method is used for solving mesh optimization problems. First, computational overhead may increase when we need to compute entries in the Hessian matrix of the objective function [6]. However, if we locally optimize one vertex at a time, Hessian matrix is very small (e.g., 2-by-2 matrix for 2D meshes) and the computational overhead is minimal. A second problem occurs when the Hessian matrix of the objective function is not positive definite. In that case, the Newton direction is not reliable. However, if a Hessian modification method is utilized, the Hessian matrix becomes positive definite and the new search direction becomes reliable. We also develop several Hessian modification methods, which can be easily applied to various mesh optimization problems.

2. Mesh Quality Optimization

Objective function for mesh optimization, $F(x)$, is most often formulated by accumulating a quality metric over all the elements of the mesh and independent variables are coordinates of the movable vertices of the mesh. Let q_i be the quality of ith vertex and N the total number of vertices in the mesh. Then, $F(x)$ is formulated as $\sum_{i=1}^{N} q_i$. We use a condition number quality metric for mesh quality improvement. For trivalent mesh corners in 3D, given by edge vectors e_1, e_2, and e_3, the quality of the vertex corner is given by [11]

$$q_i = \frac{A \cdot L}{V}, \tag{1}$$

where $A = \sqrt{\|\mathbf{e}_1 \times \mathbf{e}_2\|^2 + \|\mathbf{e}_2 \times \mathbf{e}_3\|^2 + \|\mathbf{e}_3 \times \mathbf{e}_2\|^2}$, $L = \sqrt{\sum_{j=1}^{3} \|\mathbf{e}_j\|^2}$, and $V = (\mathbf{e}_1 \times \mathbf{e}_2) \cdot \mathbf{e}_3$. The condition number quality metric is scale-invariant and prefers a right angle corner. Here, the objective function, $F(x)$, is a nonlinear function, since the quality metric is nonlinear and each vertex position affects the quality of all its connected elements.

For initially tangled meshes, we employ Escobar et al.'s [4] modification to the quality metric, which allows us to simultaneously untangle and improve mesh quality. The quality metric for 3D meshes [4] is defined as

$$q_i = \frac{(A)\,(L)}{V + \sqrt{V^2 + \delta^2}}, \tag{2}$$

where δ is a user-defined constant value. If V is positive and δ is zero, the quality metric is the same as the condition number quality metric. Introducing δ makes the objective function continuous at zero volume. The choice of δ for different problem sizes is not well-defined in [4]. The δ value should be as small as possible (close to zero) in order to remain close to the original condition number quality metric, but bigger values of δ make the function less steep and ensure more robust gradient computations. In order to satisfy these conflicting requirements, we developed an adaptive and smooth δ function, which satisfies the above requirements using a sigmoid function,

$$\delta = \frac{1}{1 + ce^{V/V_0}}, \tag{3}$$

where V_0 is a reference element volume for scale invariance, which could be chosen as some average element volume around the element being untangled and c is a constant. In our experiments we determined that $10 < c < 100$ is a good practical choice. Figure 1 shows an example of an adaptive δ function using a sigmoid function.

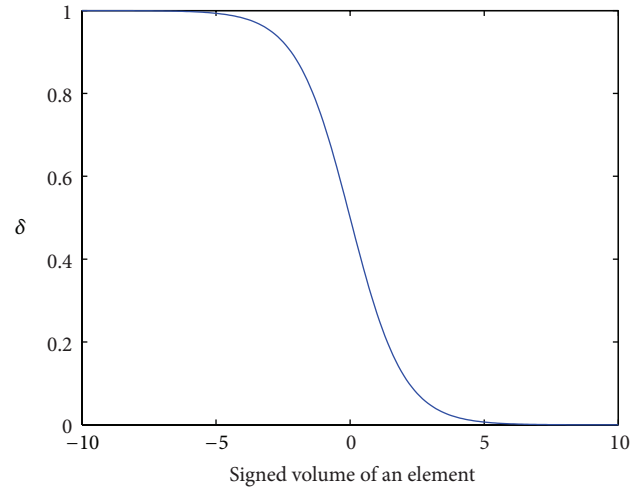


Figure 1: Example of an adaptive δ function using a sigmoid function.

In surface meshes embedded in 3D space, the movement of vertices is subject to the additional constraint that the vertices must remain on the original surface. This is best done by moving the vertices in a parametric space that maps to the surface. This method is superior to the method of allowing the optimization to derive the vertices off the surface and then pulling them back to the surface, since errors can be made in the projection step. The parametric space used can be a global parametric space [14–16] or a local parametric space based on each surface element as detailed in [3]. We use a local parametric space, since a global parametric space requires a huge computational cost. A local parametric space for a triangle element and a quadrilateral element are derived using a barycentric mapping and isoparametric mapping, respectively [3]. The optimization of the objective function with respect to a parametric space introduces additional nonlinearities in the objective function making it harder to optimize.

In any case, optimization of all types of unstructured meshes is challenging and numerical methods often experience problems of slow convergence, getting stuck in local minima far from the optimal vertex positions. Typically, meshes that need to be optimized have thousands to millions

of vertices, so in principle, the objective function and the independent vector involve all these coordinates. While it is possible to iteratively compute vertex updates to this large vector, it is quite inefficient to do so for two reasons: (1) the Jacobian of a mesh quality based objective function is quite sparse, since a vertex only affects the quality of its connected elements; (2) if the update of a vertex is to be truncated due to an explicit constraint like mesh validity or surface constraint, then the update of all vertex positions gets truncated.

Therefore, it is much more useful to use a Gauss-Seidel type (local) approach to optimize an objective function by considering the movement of one vertex at a time driven by minimization of the local component of the global objective function. It is important to reemphasize that the local objective function must necessarily include all the terms of the global objective function that the vertex position influences. Otherwise, we will be optimizing some other global function. This local approach is more memory efficient, since the local derivative matrices are small but dense, and it also leads to faster convergence because vertices are not severely constrained by the lack of movement in other far away vertices. For these reasons, we employ local mesh optimization, which optimizes only one free vertex at a time. We visit one free vertex at a time and iteratively move to other vertices. More details on local mesh optimization are described in [3, 8].

3. Nonlinear Conjugate Gradient Method Applied to Mesh Optimization

Let x_k be the (free) vertex coordinate at the kth iteration. Nonlinear conjugate gradient (NLCG) method determines a step length, λ_k, by using a line search technique along a search direction, p_k. The step length, λ_k, decreases $F(x)$ along p_k. The x_k is updated by computing

$$x_{k+1} = x_k + \lambda_k p_k. \tag{4}$$

The search direction, p_k, is given by

$$p_k = -\nabla F(x_k) + \phi_k p_{k-1}, \tag{5}$$

where $p_0 = -\nabla F(x_0)$, and ϕ_k is a parameter given by

$$\phi_k = \frac{\nabla F(x_k)^T \nabla F(x_k)}{\nabla F(x_{k-1})^T \nabla F(x_{k-1})}, \tag{6}$$

for the Fletcher and Reeves NLCG method [6]. At each iteration, the variation in NLCG is from the computation of ϕ_k [2].

For volume meshes, the line search with the *Armijo condition* [6] is performed such that the updated vertex position decreases $F(x)$ and does not result in a tangled mesh. For surface meshes, the line search with the *Armijo condition* is performed with local parametric coordinates as described in [3]. The updated parametric space should satisfy both the *mesh validity* and surface constraints, while decreasing $F(x)$. Here, *mesh validity* means the updated mesh should not be a tangled mesh. The updated vertex position should stay within parametric bounds to satisfy surface constraints. More details on applying NLCG method to surface meshes are described in [3].

Choose $\beta > 0$
While H be not positive definite **do**
  $H \leftarrow H + \beta I$
  Increase β
end while

Algorithm 1: HM1: Adding a multiple of the identity on the diagonal.

4. Steepest Descent Method Applied to Mesh Optimization

The steepest descent method [6, 9] is a line search technique, which chooses the search direction p_k as

$$p_k = -\nabla F(x_k). \tag{7}$$

The vertex coordinate is updated by computing

$$x_{k+1} = x_k + \lambda_k p_k, \tag{8}$$

where λ_k is a step length. Similar to the nonlinear conjugate gradient method, the λ_k is determined by *Armijo condition* such that the updated vertex position should sufficiently decrease in the objective function, while preventing a tangled mesh.

5. Newton's Method Applied to Mesh Optimization

We minimize $F(x)$ using Newton's method by finding vertex coordinates, x, such that $\nabla F(x)$ is zero. We use finite difference approximations to compute the gradient ($\nabla F(x)$) and Hessian matrix ($H_F(x)$) of $F(x)$. In the kth iteration, we compute a Newton update, $p(x_k)$, by solving $p(x_k) = -(H_F(x_k))^{-1}\nabla F(x_k)$. The kth vertex position is updated as $x_{k+1} = x_k + \alpha p(x_k)$, where α is a step-length. Similar to nonlinear conjugate gradient and steepest descent methods, the parameter α is determined by the *Armijo condition* [6] dictating that the update should lead to a sufficient decrease in the objective function. In practice, we start with $\alpha = 1$ and decrease it by 20% until the *Armijo condition* is satisfied. The Newton direction, p_k, reliably produces a decrease of the objective function only if the matrix, $H_F(x_k)$, is positive definite.

When $H_F(x_k)$ is not positive definite, we perform a Hessian modification step such that all eigenvalues of $H_F(x_k)$ are positive [6]. In order to make $H_F(x_k)$ (simply, H) positive definite, we consider three different Hessian modification methods. The first Hessian modification method shown in Algorithm 1 (simply, HM1) is to find a scalar β such that $H + \beta I$ is a positive definite matrix, where β is a positive constant and I is an identity matrix [17]. The β value increases until H becomes positive definite.

The second Hessian modification method shown in Algorithm 2 (HM2) is to modify eigenvalues of H by changing the signs of the negative eigenvalues in H. By simply

Find a spectral decomposition of $H = SDS^t$, where D is a diagonal matrix.
$H \leftarrow S|D|S^t$

Algorithm 2: HM2: Changing the signs of negative eigenvalues.

flipping the sign of the negative eigenvalues, it is ensured that all eigenvalues of H are positive and the updated Newton direction, p_k, is a descent direction.

The third Hessian modification method (HM3) is a more sophisticated method that uses a Cholesky factorization [6]. The goal of this Hessian modification method is to find a matrix E of small norm such that $H + E$ becomes positive definite and is decomposed using the Cholesky factorization such that $H+E = LDL^T$. Here, matrix D should be sufficiently positive definite but should not be too large [6]. The two positive parameters, η and γ, are chosen such that the following inequality holds:

$$D_{jj} \geq \eta, \qquad \left|m_{ij}\right| \leq \gamma, \tag{9}$$

where $m_{ij} = l_{ij}\sqrt{D_{jj}}$. Using these bounds, diagonal elements, D_{jj}, are computed using the equation

$$D_{jj} = \max\left(\left|C_{jj}\right|, \left(\frac{\theta_j}{\gamma}\right)^2, \eta\right), \tag{10}$$

where $\theta_j = \max_{j<i\leq n}|C_{ij}|$. Here, n values are 2 and 3 for 2D and 3D meshes, respectively. It is known that output matrices computed using HM3 have bounded condition numbers [6].

Our volume mesh optimization algorithm using Newton's method is summarized in Algorithm 4. Here, γ is a constant, which starts from $\gamma = 1$ and decreases until the updated vertex position, x_k, satisfies the mesh validity. This means the updated vertex position should not tangle the mesh.

As described earlier, surface meshes are optimized by performing optimizations with respect to local parametric coordinates (s_k). We use a condition number quality metric for mesh quality improvement [3]. For surface meshes, the updated vertex coordinates should satisfy both surface constraints and the *mesh validity*. If the updated vertex coordinates are located beyond the parametric bound, we change the parametric space to the neighborhood parametric space to satisfy the surface constraints. Similar to volume meshes, the Hessian modification step is employed when the Hessian matrix, $(H_F(s_k))$, is not positive definite. Algorithm 5 describes the surface mesh optimization algorithm using Newton's method. Similar to volume meshes, α is a constant which starts from $\alpha = 1$ and decreases until the updated vertex position satisfies *Armijo condition*. Here, γ is a constant, which starts from $\gamma = 1$ and decreases until the updated parametric space s_k and corresponding x_k satisfy *mesh validity*.

6. Numerical Experiments

First, we discuss the Hessian modification methods for solving mesh optimization problems and identify the most efficient one. Next, we compare Newton's method with the nonlinear conjugate gradient (NLCG) and steepest descent methods in terms of mesh quality and computational cost for both volume and surface meshes. We also discuss whether each method is able to untangle all inverted elements, when initial mesh is tangled.

We compare three different Hessian modification (HM) methods discussed in Section 5 for surface meshes. We focus on surface meshes since, for volume meshes, the difference between the three Hessian modification methods is negligible. The η value that is needed for the modified Cholesky factorization method (HM3) is chosen such that it is a sufficiently positive constant. We observe that a small value of η close to zero in Algorithm 3 is not desirable for test meshes, since it could result in one of the eigenvalues of the modified Hessian matrix to be zero and thus results in an indefinite Hessian matrix. For our test meshes, we observe that η value around one works well.

Figure 2 shows the worst element quality with respect to the number of Newton iterations for the pig surface mesh (see Figure 6(a)). Here, a smaller value indicates a better mesh quality. We observe that the choice of Hessian modification method considerably affects the mesh quality. We also observe that the modified Cholesky factorization method (HM3) results in the smallest worst element quality compared with the other two Hessian modification methods. The other two methods show a similar worst element quality. Table 1 summarizes the worst element quality of surface meshes (see Figure 6) for three Hessian modification methods when Newton's method converges to the local optimum. We observe that HM3 results in the smallest worst element quality for three meshes out of our four example meshes. When HM1 and HM2 methods are employed, we observe that the modified Hessian matrix sometimes becomes a nearly singular matrix or struggles with a huge condition number. Also, these two methods sometimes result in a Hessian matrix where the second-order information is not well preserved [6]. The modified Hessian matrix using the HM3 method has a bounded condition number, while preserving the second-order information. Therefore, the optimized mesh using the HM3 method results in better output meshes with good mesh qualities than other two Hessian modification methods. For these reasons, we employ HM3 as our Hessian modification method for the rest of the paper.

6.1. Volume Mesh Optimization and Untangling. We consider three volume meshes, which are shown in Figure 3. The first two meshes are polyhedral meshes, which are generated using the polyhedral mesh generation algorithm in [18]. For the cube mesh, the initial mesh is randomly perturbed such that 60% of the elements are inverted. The 3D slope

$C \leftarrow \text{diag}(H)$ (C is a diagonal matrix whose diagonal entries are diagonal entries of H)
$E \leftarrow 0$
$D \leftarrow 0$
for $j = 1, 2, \ldots, n$ **do**
  $\theta_j \leftarrow 0$
  $D_{jj} = \max\left(\left|C_{jj}\right|, \left(\frac{\theta_j}{\gamma}\right)^2, \eta\right)$, with $\theta_j = \max_{j<i\leq n}\left|C_{ij}\right|$
  $E_{jj} \leftarrow D_{jj} - C_{jj}$
  $H_{jj} \leftarrow H_{jj} + E_{jj}$
end for

Algorithm 3: HM 3: Modified Cholesky factorization.

while not converged **do**
  Compute the gradient $(\nabla F(x_k))$ and Hessian $(H \leftarrow H_F(x_k))$
  if H is not positive definite **then**
    Perform Hessian modification
  end if
  Newton direction: $p(x_k) = -H^{-1}\nabla F(x_k)$
  Update the vertex position: $x_k \leftarrow x_k + \alpha p(x_k)$ such that α satisfies the *Armijo condition*
  if x_k satisfies mesh validity **then**
    Update the vertex position x_k
  else
    Backtracking Line Search: set $x_k \leftarrow x_k + \gamma p(x_k)$,
  end if
end while

Algorithm 4: Volume mesh optimization and untangling using Newton's method.

Map the vertex coordinate x_k to a parametric space s_k [3]
while not converged **do**
  Compute the gradient $(\nabla F(s_k))$ and Hessian $(H \leftarrow H_F(s_k))$ of $q(s_k)$
  if H is not positive definite **then**
    Perform Hessian modification
  end if
  Newton direction: $p(s_k) = -H^{-1}\nabla F(s_k)$
  Update the vertex position: $s_k \leftarrow s_k + \alpha p(s_k)$ such that α satisfies the *Armijo condition*
  if s_k satisfies both the mesh validity and surface constraints **then**
    Update the vertex position s_k and map s_k to x_k
  else
    if s_k does not satisfy the mesh validity **then**
      Backtracking Line Search: set $s_k \leftarrow s_k + \gamma p(s_k)$,
      such that γ satisfies mesh validity
    end if
    if s_k does not satisfy the surface constraints **then**
      Change the parametric space to the neighborhood parametric space
    end if
    Update the vertex position s_k and map s_k to x_k
  end if
end while

Algorithm 5: Surface mesh optimization using Newton's method.

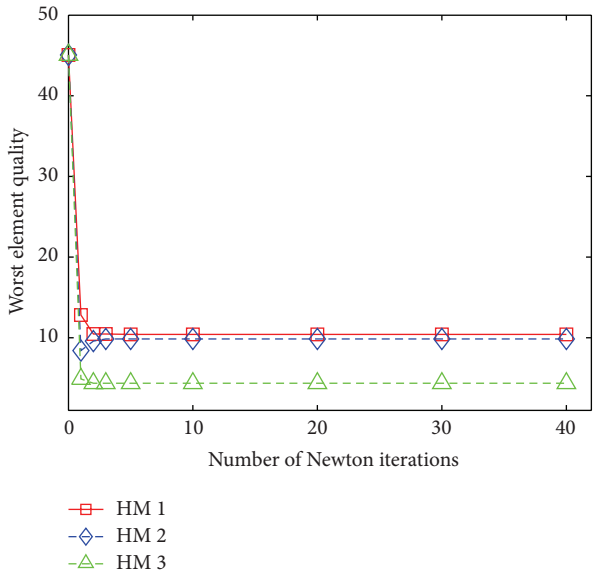


Figure 2: Convergence of three different Hessian modification (HM) methods on the Pig mesh (Figure 6(a)).

Table 1: Worst element quality after convergence for three Hessian modification methods on surface meshes (Figure 6). A smaller value indicates a better mesh quality.

Mesh (number of elements)	HM 1	HM 2	HM 3
Pig (surface, 3K)	10.46	9.85	4.35
Cow (surface, 3K)	8.74	15.22	13.28
Igea (surface, 40K)	7.46	7.19	4.20
Slope (surface, 3K)	1.70	1.70	1.70

mesh (Figure 3(a)) does not include any inverted elements and the 2-material model (Figure 3(b)) and cube meshes (Figure 3(c)) are tangled meshes which include inverted elements. For the 2-material model mesh, two different colors indicate two different materials. For the polyhedral mesh such as the 2-material model mesh, the definition of *mesh validity* is not well-defined [18]. Similar to [18], we perform a *star-shape* test to verify the *mesh validity* for this polyhedral mesh. Here, the *star-shape* test is a quite conservative definition of *mesh validity* such that all decompositions of tetrahedra from a polyhedron should have positive volumes. Figure 4 shows one example of a *star-shape* test for polyhedral meshes. More details on the *star-shape* test for polyhedral meshes are described in [18].

Figure 5 shows both initial and optimized meshes using Newton's method for a 2-material model. The initial polyhedral mesh includes several inverted elements (Figure 5(a)). Newton's method is able to simultaneously untangle and improve element qualities as shown in Figures 5(b) and 5(c). For the 2-material model, both NLCG and steepest descent methods are also able to untangle inverted elements, but they take more number of iterations to untangle all inverted elements compared with Newton's method. For a cube mesh, we observe that two elements fail to be valid and remain tangled after optimization with NLCG and steepest descent methods. However, Newton's method is able to untangle all inverted elements. The objective function for simultaneous untangling and smoothing is quite smooth; it converges to the local optimum even if the initial guess (initial mesh) is far from optimal.

Table 2: Time (sec) to optimize various volume meshes.

Mesh (number of elements)	Newton	NLCG	Steepest descent
3D slope (volume, 3K)	442	1,256	1,248
2-material (volume, 2K)	902	1,760	1,756
Cube (volume, 3K)	82	411	409

Table 2 shows timing results. We observe that, for our three example meshes, Newton's method converges between about 50% and 85% faster than the NLCG method. NLCG and steepest descent methods show similar convergence time. Newton's method is fast since, for most cases, it enjoys the full step length. Also, its convergence is quadratic around the optimal point. Table 3 shows worst element qualities computed using the condition number metric. A smaller value indicates a better mesh quality. The worst element quality using NLCG is up to 20% worse than the one using Newton's method.

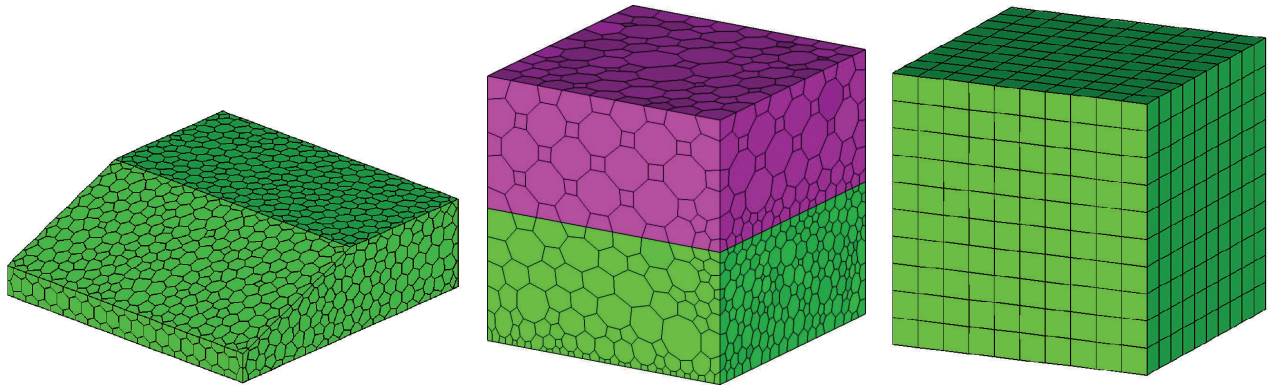
6.2. Surface Mesh Optimization. Figure 6 shows four surface meshes to optimize. Table 4 shows timing results in seconds for various surface meshes until they are optimized. Similar to volume meshes, we observe that Newton's method converges significantly faster than the NLCG and steepest descent methods. Our experimental results show that time to convergence for Newton's method is up to 4.8 times faster than for the NLCG method. Table 5 shows worst element qualities for various surface meshes. A smaller value indicates a better mesh quality. We observe that the mesh quality difference between NLCG and Newton's method is huge for surface meshes. For all these examples, Newton's method outperforms both the NLCG and steepest descent methods. For the cow (surface) mesh, the output mesh using Newton's method yields the optimized mesh which has 61% better quality than the mesh that was optimized using the NLCG method. This is because Newton's method is able to approach the local optimum while NLCG and steepest descent fail to do that when surface constraints exist. Table 6 shows surface mesh quality statistics of the cow mesh (see Figure 6(b)). We observe that Newton's method outperforms both the NLCG and steepest descent methods in terms of both average element quality and worst element quality. We observe similar statistics for other surface meshes in Figure 6.

7. Conclusion

We have proposed an efficient method for solving various mesh optimization problems using Newton's method. We observed that Newton's method significantly outperforms both the nonlinear conjugate gradient (NLCG) and steepest descent methods in terms of both the speed and the mesh quality. Obviously, the major benefit of using Newton's method is fast convergence. We observe that, for

Table 3: Worst element quality computed by the condition number metric for volume meshes.

Mesh (number of elements)	Initial	Optimized (Newton)	Optimized (NLCG)	Optimized (steepest descent)
3D slope (volume, 3K)	1,026.80	2.17	2.36	2.36
2-material (volume 2K)	191.12	1.08	1.29	1.31
Cube (volume, 3K)	1,026.00	1.09	1.17	1.17



(a) 3D slope model (b) 2-material model (tangled) (c) Cube model (tangled)

Figure 3: The element types of the three initial volume meshes are (a) 3D slope: polyhedral, (b) 2-material: polyhedral, and (c) cube: hexahedral.

Table 4: Time (sec) to optimize various surface meshes.

Mesh (number of elements)	Newton	NLCG	Steepest descent
Pig (surface, 3K)	61	226	218
Cow (surface, 3K)	90	250	241
Igea (surface, 40K)	297	1,422	1,398
Slope (surface, 3K)	504	1,810	1,802

Table 5: Worst element quality computed by the condition number metric for surface meshes. A smaller value indicates a better mesh quality.

Mesh (number of elements)	Initial	Optimized (Newton)	Optimized (NLCG)	Optimized (steepest descent)
Pig (surface, 3K)	45.07	4.35	11.34	11.34
Cow (surface 3K)	18.26	12.68	15.26	16.22
Igea (surface, 40K)	19.08	3.68	3.91	3.76
Slope (surface, 3K)	48.30	1.75	2.06	1.88

our examples, Newton's method converges up to 4.8 times faster than the NLCG method for solving mesh optimization problems. We highlight that, even for initially tangled meshes, Newton's method exhibits robust performance to untangle inverted elements and fast convergence. The benefit of using Newton's method is more critical for surface meshes due to surface constraints. For many surface meshes with surface constraints, Newton's method is able to approach the local optimum while NLCG and steepest descent methods fail to do so. For our surface mesh examples, the optimized mesh with Newton's method shows up to 61% better mesh quality when compared with the optimized mesh with the NLCG method in terms of the worst element quality. We used relatively simple finite difference approximations to compute the gradient and Hessian. We plan to apply more accurate finite difference approximations in the future.

Table 6: Surface mesh quality statistics of cow mesh (Figure 6(b)).

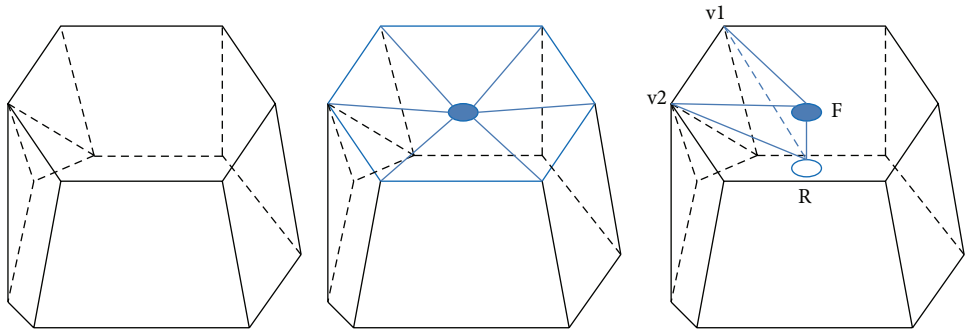
Surface element quality	Initial	Optimized (Newton)	Optimized (NLCG)	Optimized (steepest descent)
1.0–1.5	2,143	2,930	2,922	2,926
1.5–2.0	628	146	144	150
2.0–3.0	216	54	53	51
3.0–4.0	79	9	10	12
4.0–5.0	54	3	3	2
5.0–7.5	24	11	9	10
7.5–10.0	6	4	6	6
10.0–15.0	6	1	0	0
>15.0	2	0	1	1

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

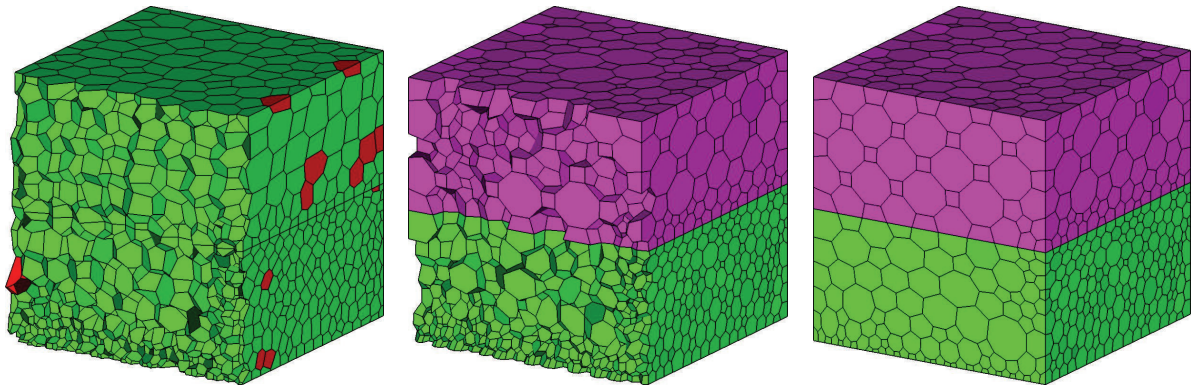
Acknowledgments

The author would like to thank Rao V. Garimella and Markus Berndt at the Los Alamos National Laboratory for helpful



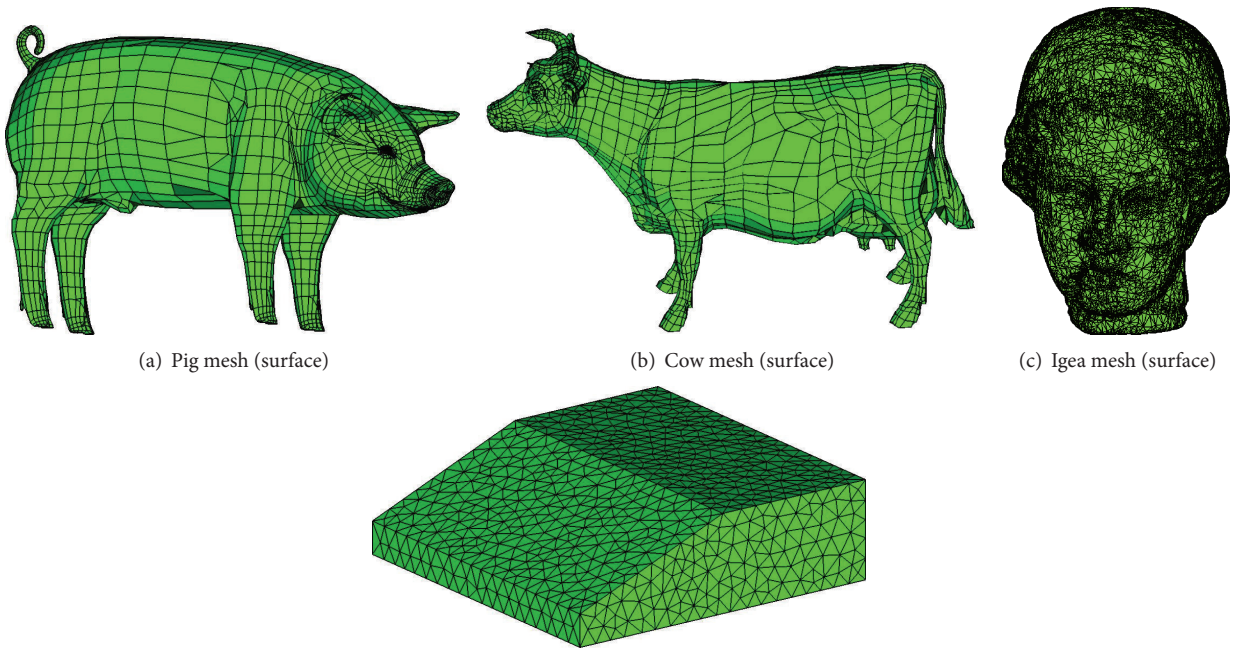
(a) (b) (c)

FIGURE 4: *Star-shape* test for polyhedral meshes. (a) General polyhedron. (b) Triangulation of face and face center. (c) One tetrahedron in the symmetric decomposition of polyhedron which is used to check *mesh validity*. Here, v1 and v2 are two vertices on the face. F and R are face and region centers, respectively.



(a) 2-material model (initial) (b) 2-material model (optimized) (c) 2-material model (optimized)

FIGURE 5: The element types of the two initial volume meshes. (a) Cut of initial polyhedral mesh showing invalid elements. Here, red elements indicate inverted elements. (b) Cut of optimized polyhedral mesh with no inverted elements. (c) Full polyhedral mesh with no inverted elements.



(a) Pig mesh (surface) (b) Cow mesh (surface) (c) Igea mesh (surface)

(d) Slope mesh (surface)

FIGURE 6: The element types of the four surface meshes are (a) Pig: mixed elements, (b) Cow: hexahedral elements, (c) Igea: tetrahedral elements, and (d) Slope: polyhedral elements.

discussions. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014R1A1A1036677).

References

[1] S. Bhowmick and S. M. Shontz, "Towards high-quality, untangled meshes via a force-directed graph embedding approach," in *Proceedings of the International Conference on Computational Science*, vol. 1, pp. 357–366, Procedia Computer Science, May 2010.

[2] J. Kim, T. Panitanarak, and S. M. Shontz, "A multiobjective mesh optimization framework for mesh quality improvement and mesh untangling," *International Journal for Numerical Methods in Engineering*, vol. 94, no. 1, pp. 20–42, 2013.

[3] R. V. Garimella, M. J. Shashkov, and P. M. Knupp, "Triangular and quadrilateral surface mesh quality optimization using local parametrization," *Computer Methods in Applied Mechanics and Engineering*, vol. 193, no. 9–11, pp. 913–928, 2004.

[4] J. M. Escobar, E. Rodríguez, R. Montenegro, G. Montero, and J. M. González-Yuste, "Simultaneous untangling and smoothing of tetrahedral meshes," *Computer Methods in Applied Mechanics and Engineering*, vol. 192, no. 25, pp. 2775–2787, 2003.

[5] Y. Zhang, C. Bajaj, and G. Xu, "Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow," in *Proceedings of the 14th International Meshing Roundtable*, pp. 449–468, 2005.

[6] J. Nocedal and S. Wright, *Numerical Optimization*, Springer, Berlin , Germany, 2006.

[7] T. Munson, "Mesh shape-quality optimization using the inverse mean-ratio metric," *Mathematical Programming: A Publication of the Mathematical Programming Society*, vol. 110, no. 3, pp. 561–590, 2007.

[8] M. Brewer, D. L. Freitag, P. M. Knupp, T. Leurent, and D. Melander, "The Mesquite mesh quality improvement toolkit," in *Proceedings of the 12th International Meshing Roundtable*, pp. 239–250, Sandia National Laboratories, September 2003.

[9] S. P. Sastry and S. M. Shontz, "Performance characterization of nonlinear optimization methods for mesh quality improvement," *Engineering with Computers*, vol. 28, no. 3, pp. 269–286, 2012.

[10] L. Freitag Diachin, P. Knupp, T. Munson, and S. Shontz, "A comparison of two optimization methods for mesh quality improvement," *Engineering with Computers*, vol. 22, no. 2, pp. 61–74, 2006.

[11] V. Dyadechko, R. V. Garimella, and M. J. Shashkov, "Reference Jacobian Rezoning Strategy for Arbitrary Lagrangian-Eulerian Methods on Polyhedral Grids," in *Proceedings of the 13th International Meshing Roundtable*, Sandia National Laboratories Report SAND #2004-3765C, pp. 459–470, Williamsburg, Va, USA, September 2004.

[12] S. P. Sastry, S. M. Shontz, and S. A. Vavasis, "A log-barrier method for mesh quality improvement," in *Proceedings of the 20th International Meshing Roundtable (IMR '11)*, pp. 329–346, October 2011.

[13] J. Kim, R. Garimella, and M. Berndt, "A practical approach for solving mesh optimization problems using," in *Proceedings of the 22nd International Meshing Roundtable*, Research Note, Sandia National Laboratories, Orlando, Fla, USA, October 2013.

[14] A. Sheffer and E. De Sturler, "Parameterization of faceted surfaces for meshing using angle-based flattening," *Engineering with Computers*, vol. 17, no. 3, pp. 326–337, 2001.

[15] M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic parameterizations of surface meshes," in *Proceedings of Eurographics Conference*, Saarbrücken, Germany, September 2002.

[16] M. S. Floater, "Parametrization and smooth approximation of surface triangulations," *Computer Aided Geometric Design*, vol. 14, no. 3, pp. 231–250, 1997.

[17] M. Heath, *Scientific Computing*, McGraw-Hill, New York, NY, USA, 2002.

[18] R. Garimella, J. Kim, and M. Berndt, "Polyhedral mesh generation and optimization for non-manifold domains," in *Proceedings of the 22nd International Meshing Roundtable*, pp. 239–250, Sandia National Laboratories, October 2013.