A Comparison of Gradient- and Hessian-Based Optimization Methods for Tetrahedral Mesh Quality Improvement*

Shankar Prasad Sastry[1] and Suzanne M. Shontz[1]

Department of Computer Science and Engineering,
The Pennsylvania State University
University Park, PA 16802
sps210@cse.psu.edu, shontz@cse.psu.edu

1 Introduction

Discretization methods, such as the finite element method, are commonly used in the solution of partial differential equations (PDEs). The accuracy of the computed solution to the PDE depends on the degree of the approximation scheme, the number of elements in the mesh [1], and the quality of the mesh [2, 3]. More specifically, it is known that as the element dihedral angles become too large, the discretization error in the finite element solution increases [4]. In addition, the stability and convergence of the finite element method is affected by poor quality elements. It is known that as the angles become too small, the condition number of the element matrix increases [5].

Recent research has shown the importance of performing mesh quality improvement before solving PDEs in order to: (1) improve the condition number of the linear systems being solved [6], (2) reduce the time to solution [7], and (3) increase the solution accuracy. Therefore, mesh quality improvement methods are often used as a post-processing step in automatic mesh generation. In this paper, we focus on mesh smoothing methods which relocate mesh vertices, while preserving mesh topology, in order to improve mesh quality.

Despite the large number of papers on mesh smoothing methods (e.g., [8, 9, 10, 11, 12, 13, 14]), little is known about the relative merits of using one solver over another in order to smooth a particular unstructured, finite element mesh. For example, it is not known in advance which solver will converge to an optimal mesh faster or which solver will yield a mesh with better quality in a given amount of time. It is also not known which solver will most aptly handle mesh perturbations or graded meshes with elements of heterogeneous volumes.

*This work was funded in part by NSF grant CNS 0720749, a Grace Woodward grant from The Pennsylvania State University, and an Institute for CyberScience grant from The Pennsylvania State University.

The answers may likely depend on the context. For example, one solver may find an approximate solution faster than the others, whereas another solver may improve the quality of meshes with heterogeneous elements more quickly than its competitors.

To answer the above questions, we use Mesquite [15], a mesh quality improvement toolkit, to perform a numerical study comparing the performance of several local mesh quality improvement methods to improve the global objective function representing the overall mesh quality as measured with various shape quality metrics. We investigate the performance of the following gradient-based methods: steepest descent [16] and Fletcher-Reeves conjugate gradient [16], and the following Hessian-based methods: quasi-Newton [16], trust-region [16], and feasible Newton [17]. Mesh quality metrics used in this study include the aspect ratio [18], inverse mean ratio [19, 20], and vertex condition number metrics [21]. The optimization solvers are compared on the basis of efficiency and ability to smooth several realistic unstructured tetrahedral finite element meshes to both accurate and inaccurate levels of mesh quality. We used Mesquite in its native state with the default parameters. Only Mesquite was employed for this study so that differences in solver implementations, data structures, and other factors would not influence the results.

In this paper, we report the results of an initial exploration of the factors stated above to determine the circumstances when the various solvers may be preferred over the others. In an effort to make the number of experiments manageable, we limit the number of free parameters. Hence, we consider a fixed mesh type and objective function. In particular, we use unstructured tetrahedral meshes and an objective function which sums the squared qualities of individual tetrahedral elements. The free parameters we investigate are the problem size, initial mesh configuration, heterogeneity in element volume, quality metric, and desired degree of accuracy in the improved mesh.

The main results of this study are as follows: (1) the behavior of the optimization solvers is influenced by the degree of accuracy desired in the solution and the size of the mesh; (2) most of the time, the gradient-based solvers exhibited superior performance compared to that of the Hessian-based solvers; (3) the rank-ordering of the optimization solvers depends on the amount of random perturbation applied; (4) the rank-ordering of the optimization solvers is the same for the affine perturbation meshes; (5) the rank-ordering of the majority of the solvers is the same for graded meshes; however, the rank of conjugate gradient is a function of time; (6) graded meshes are sensitive to changes in the mesh quality metric.

2 Problem Statement

2.1 Element and Mesh Quality

Let V and E denote the vertices and elements, respectively, of an unstructured mesh, and let $|V|$ and $|E|$ denote the numbers of vertices and elements,

respectively. Define V_B and V_I to be the set of boundary and interior mesh vertices. Let $x_v \in \mathbb{R}^n$ denote the coordinates for vertex $v \in V$. For the purposes of this paper, $n = 3$. Denote the collection of all vertex coordinates by $x \in \mathbb{R}^{n \times |V|}$. Let e be an element in E. Finally, let $x_e \in \mathbb{R}^{n \times |e|}$ the matrix of vertex coordinates for e.

We associate with the mesh a continuous function $q : \mathbb{R}^{n \times |e|} \to \mathbb{R}$ to measure the mesh quality as measured by one or more geometric properties of elements as a function of their vertex positions. In particular, let $q(x_e)$ measure the quality of element e. We assume a *smaller* value of $q(x_e)$ indicates a better quality element. A specific choice of q is an element quality metric. There are various metrics to measure shape, size, and orientation of elements [22].

The overall quality of the mesh is a function of the individual element qualities. The mesh quality depends on both the choice of the element quality metric q and the function used to combine them.

2.2 Aspect Ratio Quality Metric

An important parameter in this study is the choice of mesh quality metric. In general, we expect that the results could vary significantly depending on the choice of mesh quality metric. Thus, we consider three mesh quality metrics in this study, starting with the aspect ratio.

Various formulas have been used to compute the aspect ratio. The aspect ratio definition we employ is the one implemented in Mesquite. In particular, it is the average edge length divided by the normalized volume. Thus for tetrahedra, the aspect ratio is defined as follows:

$$\left(\frac{l_1^2 + l_2^2 + \cdots + l_6^2}{6}\right) / \left(\text{vol} \times \frac{12}{\sqrt{2}}\right),$$

where $l_i, i = 1, 2, \ldots, 6$ represent the six edge lengths, and vol represents its volume.

2.3 Inverse Mean Ratio Quality Metric

In order to derive the inverse mean ratio mesh quality metric, we let a, b, c, and d denote the four vertices of a tetrahedron labeled according to the right-hand rule. Next, define the matrix A by fixing the vertex a and denoting by e_1, e_2, and e_3 the three edge vectors emanating from a towards the remaining three vertices. Then, $A = [e_1;\ e_2;\ e_3] = [b - a;\ c - a;\ d - a]$. Next, define W to be the incidence matrix for the ideal element which is an equilateral tetrahedron in the isotropic case. In this case,

$$W = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{6} \\ 0 & 0 & \frac{\sqrt{2}}{\sqrt{3}} \end{pmatrix}.$$

Next, let $T = AW^{-1}$ transform the ideal element to the physical element. Finally, the inverse mean ratio of a tetrahedral element is as follows:

$$\frac{\|T\|_F^2}{3|\det(T)|^{\frac{2}{3}}}.$$

2.4 Vertex Condition Number Quality Metric

In order to specify the vertex condition number quality metric, we first define some notation. Let x be any vertex of an element. Let x_k denote the k^{th} neighboring vertex, for $k = 1, 2, \ldots, \mathrm{n}$. Define k edge vectors $e_k = x_k - x$. Then the Jacobian of the element is given by the matrix $A = [e_1\ e_2\ \cdots\ e_{\mathrm{n}}]$. Using A, we can define its vertex condition number as follows:

$$\|A\|_F \|A^{-1}\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius matrix norm.

All three mesh quality metrics range from 1 (for an equilateral tetrahedron) to ∞ (for a degenerate element). Invalid elements can be detected by the inverse mean ratio mesh quality metric when a complex value results.

2.5 Quality Improvement Problem

To improve the overall quality of the mesh, we assemble the local element qualities as follows: $Q = \sum_e q(x_e)^2$, where Q denotes the overall mesh quality, and $q(x_e)$ is the quality of element e. We compute an $x^* \in \mathbb{R}^{n \times |V|}$ such that x^* is a locally optimal solution to

$$\min_x Q(x) \tag{1}$$

subject to the constraint that $x_{v_B} = \overline{x_{v_B}}$, where $\overline{x_{V_B}}$ are the initial boundary vertex coordinates. In addition, we require that the initial mesh and subsequent meshes to be noninverted. This translates to the constraint $det(A^{(i)}) > 0$ for every element. In order to satisfy the two constraints, Mesquite fixes the boundary vertices and explicity checks for mesh inversion at each iteration.

3 Improvement Algorithms

In this paper, we consider the performance of five numerical optimization methods, namely, the steepest descent, conjugate gradient, quasi-Newton, trust-region, and feasible Newton methods, as implemented in Mesquite. The steepest descent and conjugate gradient solvers are gradient-based, whereas the remaining three are Hessian-based, i.e., they employ both the gradient and Hessian in the step computation. We describe each method below.

3.1 Steepest Descent Method

The steepest descent method [16] is a line search technique which takes a step along the direction $p_k = -\nabla f(x_k)$ at each iteration. In Mesquite the steplength, α_k, is chosen to satisfy the Armijo condition [23], i.e.,

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k)^T p_k$$

for some constant $c_1 \in (0, 1)$, which ensures that the step yields sufficient decrease in the objective function.

3.2 Conjugate Gradient Method

The conjugate gradient method [16] is a line search technique which takes a step in a direction which is a linear combination of the negative gradient at the current iteration and the previous direction, i.e.,.

$$p_k = -\nabla f(x_k) + \beta_k p_{k-1},$$

where $p_0 = -\nabla f(x_0)$. Conjugate gradient methods vary in their computation of β_k. The Fletcher-Reeves conjugate gradient method implemented in Mesquite computes

$$\beta_k^{FR} = \frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_{k-1})^T \nabla f(x_{k-1})}.$$

Care is taken in the line search employed by Mesquite to compute a steplength yielding both a feasible step (i.e., one which does not result in a tangled mesh) and an approximate minimum of the objective function along the line of interest.

3.3 Quasi-Newton Method

Quasi-Newton methods [16] are line search (or trust-region) algorithms which replace the exact Hessian in Newton's method with an approximate Hessian in the computation of the Newton step. Thus, quasi-Newton methods solve $B_k p_k = -\nabla f(x_k)$, for some $B_k \approx \nabla^2 f(x_k)$ at each iteration in an attempt to find a stationary point, i.e., a point where $\nabla f(x) = 0$. The quasi-Newton implementation in Mesquite [15] is a line search that approximates the Hessian using the gradient and true values of the diagonal blocks of the Hessian.

3.4 Trust-Region Method

Trust-region methods [16] are generalizations of line search algorithms in that they allow the optimization algorithm to take steps in any direction provided that the steps are no longer than a maximum steplength. Steps are computed by minimizing a quadratic model of the function over the trust region. The trust region is expanded or contracted at each iteration depending upon how reflective the model is of the objective function at the given iteration.

3.5 Feasible Newton Method

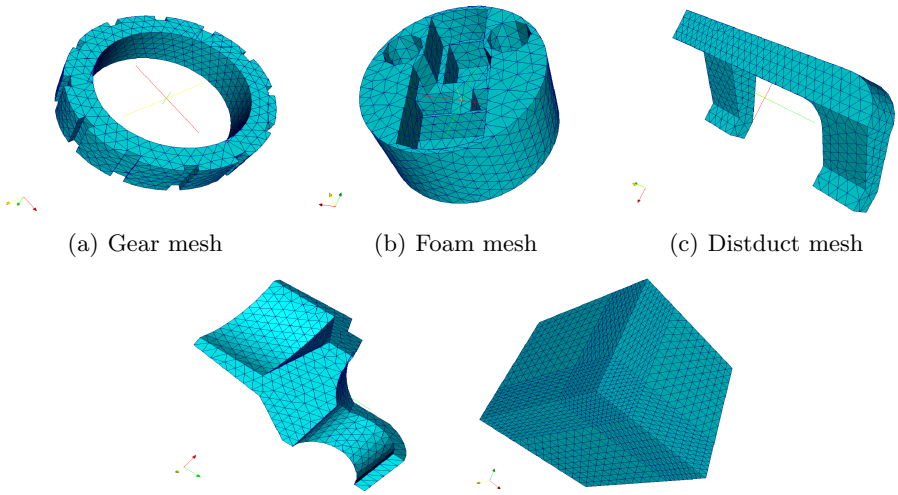
The feasible Newton method [17] is a specialized method for mesh quality improvement. In particular, it uses an inexact Newton method [24, 16] with an Armijo line search [23] to determine the direction in which to move the vertex coordinates. At each iteration, the algorithm solves the Newton equations via a conjugate gradient method with a block Jacobi preconditioner [24]. The solver also obtains good locality of reference.

4 Numerical Experiments

In this section, we report results from four numerical experiments designed to determine when each of the five solvers are preferred according to their time to convergence for local mesh smoothing. All solvers are implemented in Mesquite 2.0, the Mesh Quality Improvement Toolkit [15], and were run with their default parameter values. We solve the optimization problem (1) on a series of tetrahedral meshes generated with the CUBIT [25] and Tetgen [26] mesh generation packages. We consider the following geometries: distduct, foam, gear, hook [27] and cube. Sample meshes are shown in Figure 1. In the first three experiments, we study the effects of three different problem parameters on the time taken to reach x^*, a locally optimal solution. The problem parameters of interest are: problem size, initial mesh configuration, and grading of mesh elements. For each of the three parameters studied, we create a set of test meshes in which we isolate the parameter of interest and allow it to vary; these experiments were inspired by [28, 29]. Particular attention was paid to ensure that the remaining parameters were held as constant as possible. Due to space limitations, we have omitted most of the tables of initial mesh quality statistics which demonstrate this. In the fourth experiment, we investigate the effect that mesh quality metric has on solver performance.

Because the objective functions used for our experiments are non-convex, the optimization techniques may converge to different local minima. To ensure that this did not effect our study, we verified for each experiment whether or not the solvers converge to the same optimal mesh by comparing vertex coordinates of the optimal meshes.

In the following subsections, we describe the problem characteristics of the test meshes in terms of the numbers of vertices and elements, initial mesh quality (according to the mesh quality metric of interest), and parameter values of interest (such as magnitude of perturbation). We then specify performance results for the five optimization solvers. In all cases, the solution is considered optimal when it has converged to six significant digits. The machine employed for this study is equipped with an Intel P4 processor (2.67 GHz). The 32-bit machine has 1GB of RAM, a 512KB L2 cache, and runs Linux.



(a) Gear mesh (b) Foam mesh (c) Distduct mesh

(d) Hook mesh (e) Cube mesh

Fig. 1. Sample meshes on the gear, foam, distduct, hook, and cube geometries. Geometries (a)-(d) were provided to us by Dr. Patrick Knupp of Sandia National Laboratories [27].

4.1 Increasing problem size

To test the effect that increasing the problem size has on optimization solver performance, we used CUBIT to generate a series of tetrahedral meshes with an increasing number of vertices while maintaining uniform mesh quality and element size. A series of meshes were generated for the distduct, foam, gear, and hook geometries shown in Figures 1(a) through 1(d); for each series of meshes, the number of elements is increased from approximately 5000 to 500,000 elements.

In the creation of the test meshes, care was taken to ensure that, for each mesh geometry, we achieve our goal of maintaining roughly uniform element size and mesh quality distributions. Table 1 shows the initial and final aspect ratio quality before and after conjugate gradient method was applied on three of the meshes. Such changes in mesh quality were typical of the results seen in this experiment.

For each mesh geometry, when the aspect ratio mesh quality metric was employed, the time to convergence required increased linearly with an increase in problem size. Figure 2 illustrates this trend for the use of the various solvers on the distduct geometry. Solver behavior was identical on the remaining geometries; in particular, the solvers also converged to the same optimal meshes. Thus, additional figures have been omitted. This is expected as the number of iterations to convergence is more or less a constant, and the time per iteration increases linearly with the number of elements used for local mesh smoothing.

Distduct Mesh			Mesh Quality (Aspect Ratio)				
# Vertices	# Elements	Phase	min	avg	rms	max	std dev
1,262	5,150	Initial	1.00557	1.33342	1.35118	2.71287	0.218363
		Final	1.00077	1.27587	1.28932	2.83607	0.185684
19,602	99,895	Initial	1.0007	1.28014	1.29531	10.3188	0.197718
		Final	1.00065	1.21742	1.22755	4.8624	0.157424
92,316	498,151	Initial	1.00009	1.27055	1.28513	18.5592	0.193054
		Final	1.00004	1.18949	1.1977	18.5592	0.139968

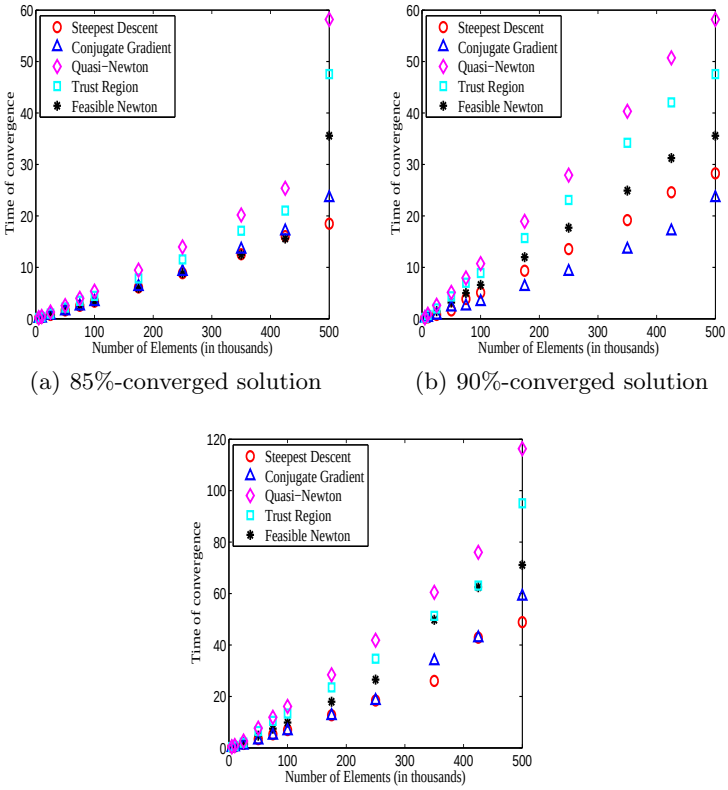
Table 1. Initial and final mesh quality after smoothing the distduct mesh with the conjugate gradient method using the aspect ratio mesh quality metric.

There are instances where a deviation from linearity is seen in larger meshes. These are likely due to limitations on the size of the mesh which can fit in the cache; small meshes may fit entirely in the cache, whereas larger meshes may only partially fit in the cache.

We now examine the behavior of the various solvers on the distduct meshes with the use of the aspect ratio quality metric. For engineering applications, a highly accurate solution is not often needed or even desired. Thus, we consider partially-converged as well as fully-converged solutions. In each case, we consider smoothing with 85%, 90%, and 100%-converged solutions; the results are shown in Figure 2. The legend for the remaining plots in the paper is as follows: 'circle' (steepest descent), 'triangle' (conjugate gradient), 'diamond' (quasi-Newton), 'square' (trust-region), and 'star' (feasible Newton).

In all the cases, i.e., for the 85%−, 90%-, and 100%−converged solutions, the five optimization solvers converged towards the same optimal mesh. For the 85%−converged solutions, feasible Newton is the fastest method to reach an optimal solution (see Figure 2(a)); few iterations were required since the initial CUBIT-generated meshes were of fairly good quality. Feasible Newton was possibly the quickest method since it takes fewer iterations than the other methods; however, each iteration takes a greater amount of time than the other solvers. The ranking of all solvers in order of fastest to slowest on the larger meshes is: feasible Newton < steepest descent < conjugate gradient < trust-region < quasi-Newton. For the smaller meshes, the rank-ordering is: conjugate gradient < feasible Newton < steepest descent < trust-region < quasi-Newton. In general, the gradient-based solvers (i.e, steepest descent and conjugate gradient) performed better than the Hessian-based solvers (trust-region and quasi-Newton). However, feasible Newton, which is a Hessian-based solver, performed very competitively. This is likely due to the fact that local mesh smoothing was performed with a highly-tuned solver. In addition, the rank ordering of the solvers depends on the mesh size as noted above.

In the majority of the 90%-converged solution cases (see Figure 2(b)), the conjugate gradient algorithm reached convergence faster than the other methods. This was followed by the steepest descent, feasible Newton, trust-region, and quasi-Newton methods, respectively. This ordering is different



(a) 85%-converged solution (b) 90%-converged solution

(c) 100%-converged solution

Fig. 2. Mesh smoothing to various convergence levels: (a) 85%-converged solution; (b) 90%-converged solution; (c) 100%-converged solution. Results are for the distduct meshes with the aspect ratio quality metric.

than that which was obtained for the 85% case. Because local mesh smoothing was performed, only one vertex in the mesh is moved at a time. The steepest descent and conjugate gradient methods use only the gradient of the objective function to move a vertex to its optimal location. The other methods also use the Hessian of the objective function to move the vertex. The calculation of the Hessian adds computational expense, making the Hessian-based methods comparatively slower. However, Hessians may effect local mesh smoothing results less than global mesh smoothing results where the Hessian matrices are much larger. The conjugate gradient method is superior to steepest descent since it uses gradient history to determine the optimal vertex position.

In the majority of the 100%-converged solution case (see Figure 2(c)), the conjugate gradient algorithm was the fastest to reach convegence for smaller meshes; however, the steepest descent method proved to be faster for larger

meshes. This is probably due to the increase in memory which is required for larger meshes. Eventually the increased requirements on the performance of the cache may slow down the conjugate gradient algorithm relative to the steepest descent algorithm since it must store and access an additional vector.

In conclusion, the behavior of the optimization solvers is influenced by the degree of accuracy desired in the solution and the size of the mesh. Most of the time, the gradient-based optimization solvers exhibited superior performance to that of the Hessian-based solvers.

4.2 Initial mesh configuration

In order to investigate the effect that the initial mesh configuration (as measured by distance from optimal mesh) had on the performance of the five solvers, a series of perturbed meshes, based on the 500,000 element distduct, foam, gear, and hook meshes from the previous experiment, were designed. In particular, the meshes were smoothed initially using the aspect ratio mesh quality metric. Then, random or systematic perturbations were applied to the interior vertices of the optimal mesh. For all experiments, the perturbations were applied to all interior vertices and to a randomly chosen subsets of vertices of size 5%, 10%, 25%, and 100% of the interior vertices. The formulas for the perturbations are as follows:

Random: $x_v = x_v + \alpha_v r$, where r is a vector of random numbers generated using the rand function, and α_v is a multiplicative factor controlling the amount of perturbation. For our experiments, we chose a random value for α_v; the resulting meshes were checked to verify that they were of poor quality.

Translational: $x_v = x_v + \alpha s$, where s is a direction vector giving the coordinates to be shifted, and α is a multiplicative factor controlling the degree of perturbation. In this case, we consider the shift with $s = [1\ 0\ 0]^T$ and α values ranging from 0.016 to 1.52 were used to maximize the amount of perturbation a particular mesh could withstand before the elements became inverted. Thus, the specific value of α chosen for a mesh depended upon the size of the elements.

Random Perturbations

The results obtained here differ somewhat from the results obtained from the scalability experiment above. They are similar in that the gradient-based methods performed better than the Hessian-based methods. This can be attributed to the greater computational expense of computing the Hessian matrices for a smaller payoff in terms of a decrease in the objective function. The main difference here is that, in almost all cases, the steepest descent algorithm performs better than the conjugate gradient algorithm.

For this experiment, the meshes to be smoothed were perturbed from the fully optimized CUBIT-generated meshes. Thus, the initial meshes are of poorer quality. Starting with poor quality meshes, i.e., far away from an

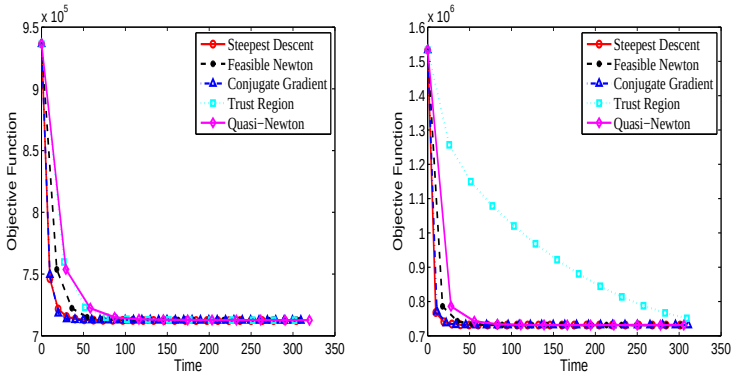
optimal mesh, had a very significant impact on the performance of the solvers. There are cases when the conjugate gradient method does better than the steepest descent method when the quality of the input mesh is reasonably good. In this case, all solvers converged to the same optimal mesh.

However, when we start with a poor quality initial mesh, a coarse-scale improvement in the the mesh is needed. Once the mesh has been sufficiently smoothed, fine-scale improvements can be obtained through the use of superior solvers. In most cases, because the perturbation was large, coarse-scale smoothing was needed. As a result, the performance of steepest descent was the best (also due to the lower complexity of the algorithm). When the perturbations were small, fine-scale smoothing requirements imply that superior methods will converge faster. This was indeed seen when the perturbations were small. The conjugate gradient method's performance was better than that of steepest descent in such cases. However, the Hessian-based methods were slower because of their inherent computational complexity. Figure 3(a) shows typical objective function versus time plots for our experiments.

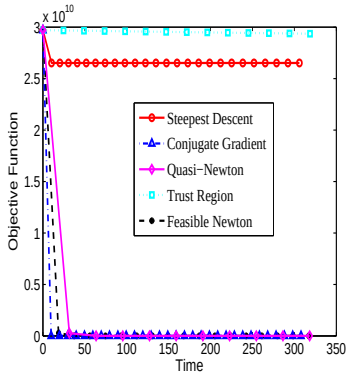
The behavior of the trust region method was distinctly different than that of the other algorithms. For small perturbations from the optimal mesh, the behavior of the trust-region method almost coincided with that of the other methods in the quality versus time plots. Figure 3(b) below illustrates an example of such behavior.

However, when the perturbations were large, the trust-region method was much slower than the other methods in terms of time to convergence. This is due to the constraint of the spherical trust-region bounding the maximum acceptable steplength at each iteration. This conservative approach slows the time to convergence of the trust-region method. It was also observed that, for large perturbations, the steepest descent method does not converge to the same optimal mesh as the other methods. In particular, it converges to an optimal mesh with a higher objective function value. The plot shown in Figure 3(c) is a good example of the dismal performance of the trust-region and steepest descent methods in the large perturbation case.

In conclusion, the rank-ordering of the optimization solvers depends upon the amount of random perturbations applied to the initial meshes in the context of mesh smoothing using the aspect ratio quality metric. In particular, all five methods performed competitively for the small perturbation case; however, the steepest descent and conjugate gradient methods performed the best. In the case of medium-sized perturbations, the steepest descent method performed the best, and the trust-region method performed very slowly. The other three methods exhibited average performance. Finally, for the case of large perturbations, the trust-region method is very slow to converge, and the steepest descent method may converge to a mesh of poorer quality.



(a) 10% interior vertices perturbed (b) 10% interior vertices perturbed

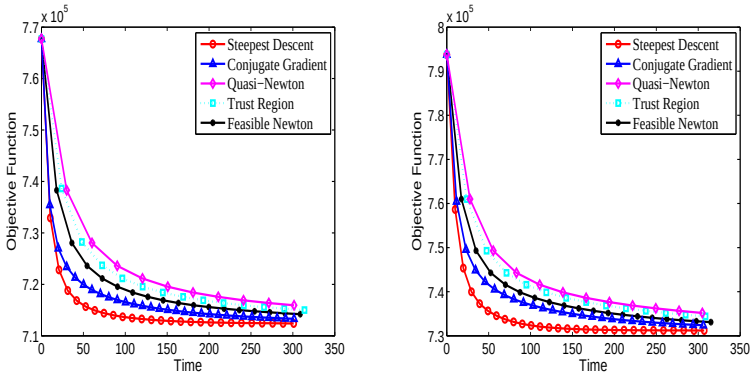


(c) 5% interior vertices perturbed

Fig. 3. Typical results from the random perturbation experiment. Results were obtained by smoothing the 500,000 element meshes using the aspect ratio quality metric. (a) The result is for the gear mesh with 10% of its vertices perturbed; because the perturbation was small, the behavior of the trust-region method was almost coincident with that of the other solvers. (b) The result is for the distduct mesh with 10% of its vertices perturbed; here the trust-region method is competitive when the initial mesh is of reasonable quality due to the medium-size perturbation. (c) The result is for the distduct mesh with 5% of its vertices perturbed. Because the perturbations were large, the steepest descent and trust-region methods performed very poorly.

Affine Perturbations

In order to determine the effect that affine perturbations had on the performance of the optimization solvers, the affine (translation) perturbation shown above was applied to all interior mesh vertices once the appropriate initial 500,000 element distduct, foam, gear, and hook meshes were smoothed according to the aspect ratio mesh quality metric.



(a) Mesh smoothing of affinely perturbed distduct mesh

(b) Mesh smoothing of affinely perturbed gear mesh

Fig. 4. Typical results for the affine perturbation experiment using the aspect ratio for local mesh smoothing. The results are for smoothing the distduct and hook meshes with 500,000 elements after all interior vertices were affinely perturbed.

The qualities of the interior elements of the perturbed meshes were still fairly good since the transformation applied was affine; however the qualities of the boundary elements was much worse. It is expected that the convergence plots for all of the solvers will start with rapid decrease in the objective function and will end with a small decrease in the objective function. This is because the initial meshes were created by applying as large an affine perturbation as possible before mesh inversion occurred, thus generating meshes rather far away from the optimal ones. This behavior is typical and is observed in the plots shown in Figure 4. The time taken per nonlinear iteration varies with the computational complexity of the algorithm. However, the objective function values (for the various solvers) remain rather similar over the first few iterations until, eventually, more vertex movement occurs, and the objective function values become less predictable. However, all solvers did converge to the same optimal mesh.

The steepest descent method, being the least computationally expensive method, spends less time per iteration and converges to an optimal mesh fairly quickly. The ranking of the optimization solvers for the affine perturbation meshes is as follows: steepest descent $<$ conjugate gradient $<$ feasible Newton $<$ trust-region $<$ quasi-Newton. This rank-ordering demonstrates that methods for which every iteration is faster converge before methods for which each iteration is slower.

In conclusion, the optimization solvers exhibited a distinct rank-ordering for the affine perturbation meshes in the context of local mesh smoothing using the aspect ratio quality metric. In particular, the rank-ordering was as

follows: steepest descent < conjugate gradient < feasible Newton < trust-region < quasi-Newton.

4.3 Graded Meshes

Our second test set was generated using Tetgen in order to test the effect that grading of mesh elements has on the performance of the five optimization solvers, as graded meshes have a larger distribution of element mesh qualities. For this experiment, three sets of structured tetrahedral meshes were generated which contain the same numbers of vertices and elements but whose elements have different volumes. The meshes were constructed on a cube domain having a side length of 20 units. In the first set of meshes, the vertices were evenly distributed in two of the three axes, but, for the other axis, half of the vertices were placed in first 10%, 20%, 30%, or 40% of the volume. Two additional sets of test meshes were created with the density of vertices varying in two and three directions instead of variation in only one direction. After the point clouds were created, Tetgen was used to create a volume mesh of the cube domain. The resulting Delaunay meshes, which were created without using any quality control features, was used for the graded mesh experiment. See Figure 1(e) for an example of a mesh created with half of its vertices occupying 30% of the space in all three axes and distributed uniformly throughout the rest of the cube volume.

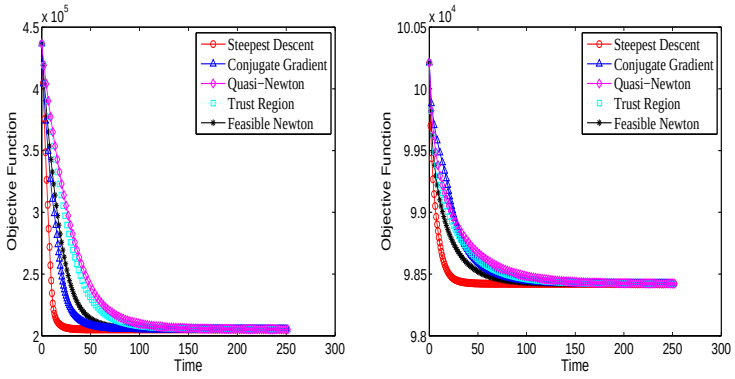
This mesh generation technique results in a structured mesh with heterogeneous elements in terms of volume. In particular, approximately one-fourth, one-half, and one-fourth of the mesh elements can be considered small, medium, and large, respectively. All of the meshes generated contain 8000 vertices and 41,154 tetrahedra.

The results obtained from this experiment are shown in Figure 5. The mesh smoothing results for the graded meshes are similar to those observed in the affine perturbation case. The main difference between the two experiments is the behavior of the conjugate gradient method. For the graded meshes, there is a definite hierarchy among the other four solvers; the rank-ordering is as follows: steepest descent < feasible Newton < trust-region < quasi-Newton. However, the rank of the conjugate gradient method with respect to the other solvers varies as a function of time.

In conclusion, the rank-ordering of the conjugate gradient method varied as a function of time as the graded meshes were smoothed using the aspect ratio mesh quality metric. However, the rank-ordering of the remaining four optimization solvers was as follows: steepest descent < feasible-Newton < trust-region < quasi-Newton.

4.4 Mesh quality metric

Our final experiment was designed to investigate the effect of the choice of mesh quality metric on the performance of the optimization methods. For



(a) 10% (b) 40%

Fig. 5. Mesh smoothing results for the graded meshes using the aspect ratio mesh quality metric. The percentages indicate the amount of volume used by the first half of the vertices in a given axis within the cube domain.

this experiment, we investigated the performance of the various methods on the distduct, foam, gear, hook, and cube meshes by repeating a subset of the above experiments for the inverse mean ratio and vertex condition number quality metrics.

The results of performing the scaling experiment for the inverse mean ratio and vertex condition number quality metrics are the same as those for the aspect ratio mesh quality metric described above.

Performing the random perturbation experiment for the inverse mean ratio and vertex condition number quality metrics yielded results that were qualitatively the same, i.e., the results could be classified into one of the above three cases depending upon how large were the perturbations.

The results of performing the affine perturbation experiment for the inverse mean ratio and vertex condition number mesh quality metric yielded results similar to those when the aspect ratio mesh quality metric was used.

Performing the element heterogeneity experiment for the inverse mean ratio mesh quality metric yielded results that were the same as those observed earlier for the aspect ratio mesh quality metric. However, the results are different for the vertex condition number mesh quality metric. When the vertex condition number metric is employed for mesh smoothing in the context of the graded mesh experiment, we observe a small rise in the objective function after a significant initial decrease as seen in Figure 6. The plots in this figure are for the cube meshes with vertices in all three axes distributed nonuniformly to create graded meshes with elements of heterogeneous volume. Although such behavior is rare, it is possible, as local mesh smoothing is being applied with a global objective function. Further investigation into the cause of such behavior for the meshes in this experiment is needed.

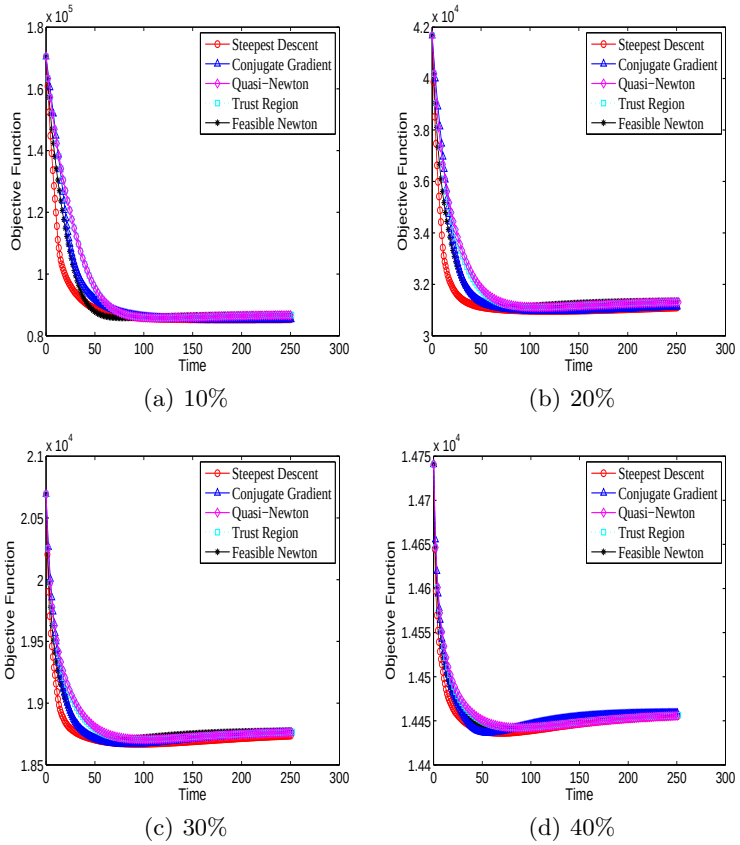


Fig. 6. Mesh smoothing results for the cube meshes with heterogeneous element volumes using the vertex condition number mesh quality metric. The percentages indicate the percentage of volume used by the first half of the vertices in all three axes within the cube domain.

In conclusion, the scaling experiment results were insenstivive to the choice of mesh quality metric. However, the perturbation and element heterogeneity results were indeed sensitive to the choice of mesh quality metric. Further research is needed to identify additional contexts where the choice of mesh quality metric influences optimization solver behavior.

5 Future Work

The results in this study are specific to local mesh quality improvement of unstructured tetrahedral meshes via five optimization solvers, namely, the steepest descent, Fletcher-Reeves conjugate gradient, quasi-Newton, trust-region,

and feasible Newton methods, with mesh quality measured according to the three specified quality metrics, namely the aspect ratio, inverse mean ratio, and vertex condition number. The results we obtained may vary dramatically if global mesh quality improvement methods were used instead of the local ones studied here [28, 29]; hence, we plan to investigate global versions of these solvers in future work. In addition, vertex ordering has been shown to play an important role in convergence of the Feasnewt solver when used for local mesh optimization [30]; thus, we will also investigate the effect of vertex ordering in the future. We also plan to investigate the role that other non-shape quality metrics have on the mesh optimization methods with the goal of identifying other contexts where quality metrics influence optimization solver behavior. Finally, we plan to investigate the use of hybrid solvers to improve optimization solver performance.

References

1. I. Babuska and M. Suri, *The p and h-p versions of the finite element method, basic principles, and properties*, SIAM Review, 35: 579-632, 1994.
2. M. Berzins, *Solution-based mesh quality for triangular and tetrahedral meshes*, in Proceedings of the 6th International Meshing Roundtable, Sandia National Laboratories, pp. 427-436, 1997.
3. M. Berzins, *Mesh quality - Geometry, error estimates, or both?*, in Proceedings of the 7th International Meshing Roundtable, Sandia National Laboratories, pp. 229-237, 1998.
4. I. Babuska and A. Aziz, *On the angle condition in the finite element method*, SIAM J. Numer. Anal., 13: 214-226, 1976.
5. E. Fried, *Condition of finite element matrices generated from nonuniform meshes*, AIAA Journal, 10: 219-221, 1972.
6. J. Shewchuk, *What is a good linear element? Interpolation, conditioning, and quality measures*, in Proceedings of the 11th International Meshing Roundtable, Sandia National Laboratories, pp. 115-126, 2002.
7. L. Freitag and C. Ollivier-Gooch, *A cost/benefit analysis for simplicial mesh improvement techniques as measured by solution efficiency*, Internat. J. Comput. Geom. Appl., 10: 361-382, 2000.
8. P. Knupp and L. Freitag, *Tetrahedral mesh improvement via optimization of the element condition number*, Int. J. Numer. Meth. Eng., 53: 1377-1391, 2002.
9. L. Freitag and P. Plassmann, *Local optimization-based simplicial mesh untangling and improvement*, Int. J. Numer. Meth. Eng., 49: 109-125, 2000.
10. N. Amenta and M. Bern and D. Eppstein, *Optimal point placement for mesh smoothing*, In Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms, pp. 528-537, 1997.
11. P. Zavattieri, *Optimization strategies in unstructured mesh generation*, Int. J. Numer. Meth. Eng., 39: 2055-2071, 1996.
12. E. Amezua and M. Hormaza and A. Hernandez and M. Ajuria, *A method of the improvement of 3D solid finite element meshes*, Adv. Eng. Softw., 22: 45-53, 1995.

13. S. Canann and M. Stephenson and T. Blacker, *Optismoothing: An optimization-driven approach to mesh smoothing*, Finite Elem. Anal. Des., 13: 185-190, 1993.
14. V. Parthasarathy and S. Kodiyalam, *A constrained optimization approach to finite element mesh smoothing*, Finite Elem. Anal. Des., 9: 309-320, 1991.
15. M. Brewer and L. Freitag Diachin and P. Knupp and T. Leurent and D. Melander, *The Mesquite Mesh Quality Improvement Toolkit*, In Proceedings of the 12th International Meshing Roundtable, Sandia National Laboratories, pp. 239-250, 2003.
16. J. Nocedal and S. Wright, *Numerical Optimization*, Springer-Verlag, 2nd Edition, 2006.
17. T. Munson, *Mesh Shape-Quality Optimization Using the Inverse Mean-Ratio Metric*, Mathematical Programming, 110: 561-590, 2007.
18. J. Cavendish and D. Field and W. Frey, *An approach to automatic three-dimensional finite element mesh generation*, Int. J. Num. Meth. Eng., 21: 329-347, 1985.
19. A. Liu and B. Joe, *Relationship between tetrahedron quality measures*, BIT, 34: 268-287, 1994.
20. P. Knupp, *Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities, Part II - A framework for volume mesh optimization and the condition number of the Jacobian matrix*, Int. J. Numer. Meth. Eng., 48: 1165-1185, 2000.
21. P. Knupp, *Matrix norms and the condition number*, In Proceedings of the 8th International Meshing Roundtable, Sandia National Laboratories, pp. 13-22, 1999.
22. P. Knupp, *Algebraic mesh quality metrics*, SIAM J. Sci. Comput., 23:193-218, 2001.
23. L. Armijo, *Minimization of functions having Lipschitz-continuous first partial derivatives*, Pacific Journal of Mathematics, 16:1-3, 1966.
24. C.T. Kelley, *Solving Nonlinear Equations with Newton's Method*, SIAM, Philadelphia, Pennsylvania, 2003.
25. Sandia National Laboratories, CUBIT Generation and Mesh Generation Toolkit, http://cubit.sandia.gov/.
26. H. Si, TetGen - A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator, http://tetgen.berlios.de/.
27. P. Knupp, Personal communication, 2009.
28. L. Freitag and P. Knupp and T. Munson and S. Shontz, *A comparison of inexact Newton and coordinate descent mesh optimization techniques*, In Proceedings of the 13th International Meshing Roundtable, Sandia National Laboratories, pp. 243-254, 2004.
29. L. Diachin and P. Knupp and T. Munson and S. Shontz, *A comparison of two optimization methods for mesh quality improvement*, Eng. Comput., 22:61-74, 2006.
30. S.M. Shontz and P. Knupp, *The effect of vertex reordering on 2D local mesh optimization efficiency*, In Proceedings of the 17th International Meshing Roundtable, Sandia National Laboratories, pp. 107-124, 2008.