Noname manuscript No.
(will be inserted by the editor)

Performance characterization of nonlinear optimization methods for mesh quality improvement[*][†]

Shankar Prasad Sastry · **Suzanne M. Shontz**

Received: date / Accepted: date

Abstract We characterize the performance of gradient- and Hessian-based optimization methods for mesh quality improvement. In particular, we consider the steepest descent and Polack-Ribiere conjugate gradient methods which are gradient based. In the Hessian-based category, we consider the quasi-Newton, trust region, and feasible Newton methods. These techniques are used to improve the quality of a mesh by repositioning the vertices, where the overall mesh quality is measured by the sum of the squares of individual elements according to the aspect ratio metric. The effects of the desired degree of accuracy in the improved mesh, problem size, initial mesh configuration, and heterogeneity in element volume on the performance of the optimization solvers are characterized on a series of tetrahedral meshes.

Keywords Mesh quality improvement · optimization solvers · performance characterization

[*] The work of the first author was funded in part by an Institute for CyberScience grant from The Pennsylvania State University. The work of the second author was funded in part by NSF grant CNS 0720749 and a Grace Woodward grant from The Pennsylvania State University.

[†] A preliminary version of a portion of these results appeared in shortened form in the Proceedings of the 2009 International Meshing Roundtable.

Shankar Prasad Sastry
The Pennsylvania State University
University Park, PA 16802
E-mail: sps210@cse.psu.edu

Suzanne Shontz
The Pennsylvania State University
University Park, PA 16802
E-mail: 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.

The three major classes of unstructured mesh quality improvement methods are as follows: (1) point insertion/deletion methods which refine/coarsen the mesh in order to improve the local length scale of the mesh [8–11]; (2) local reconnection methods which change the topology of the mesh through edge or face swapping of a specified set of vertices [11–14], and (3) mesh smoothing methods which relocate vertices in order to improve mesh quality while preserving the topology of the mesh [15–17]. For this research, we focus on mesh quality improvement methods in the third class.

Despite the large number of papers on mesh smoothing methods (e.g., [18–21,15–17]), 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. 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 [22], a mesh quality improvement toolkit, to perform a numerical study comparing the performance of several local and global mesh quality improvement methods to improve the global objective function representing the overall mesh quality. We investigate the performance of the following gradient-based methods: steepest descent [23] and Polack-Ribiere conjugate gradient [23], and the following Hessian-based methods: quasi-Newton [23], trust region [23], and feasible Newton [24]. 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 as measured by the aspect ratio mesh quality metric [25]. We took a "black-box" approach and used the Mesquite solvers in their native state and with the default parameters. This approach was taken since this is how the majority of the practitioners use Mesquite and other such software packages [26]. 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 desired degree of accuracy in the improved mesh, problem size, initial mesh configuration, and heterogeneity in element volume.

The rest of this paper is organized as follows. In Section 2, we describe the mesh quality improvement problem. In Section 3, we describe the mesh quality improvement algorithms as implemented in Mesquite. In Section 4, we describe our numerical experiments to determine the most efficient solver in various contexts. Finally, we give some conclusions and describe our plans for future reasearch in Section 5.

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.

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 consideration 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. However, in order to minimize the number of free parameters, we consider only the aspect ratio quality metric [6].

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:

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

where $l_i, i = 1, 2, \ldots, 6$ represent the six edge lengths, and vol represents its volume [27]. The range of values for this metric is from 1 to ∞. The optimal value of the metric is 1.

2.3 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. All optimization problems in this paper are formulated as (1).

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 [23] 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 [28], 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. When the Armijo condition is not met, the step length is reduced by a factor, namely the reduction parameter. Also, when the step results in a tangled mesh, the step length is reduced by a factor, namely the backtracking parameter.

3.2 Conjugate Gradient Method

The conjugate gradient method [23] 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 Polack-Ribiere conjugate gradient method implemented in Mesquite computes

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

Care is taken in the Armijo 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 [23] 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 [22] 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 [23] 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 using a standard backtracking Armijo linesearch. A block diagonal preconditioner is employed in Mesquite; the preconditioner gets modified if the diagonal block is not positive definite. 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 [24] is a specialized method for mesh quality improvement. In particular, it uses an inexact Newton method [29, 23] with an Armijo line search [28] 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 [29]. The solver also obtains good locality of reference by vertex and element reordering steps performed as a preprocessing step using a breadth-first search.

4 Numerical Experiments

In this section, we report results from a set of three numerical experiments designed to determine when each of the five solvers are preferred according to their quality improvement as a function of time. We consider both local and global mesh quality improvement methods separately. All solvers are implemented in Mesquite 2.0, the Mesh Quality Improvement Toolkit [22], and were run with their default parameter values (as shown in Table 1). We solve the optimization problem (1) on a series of tetrahedral meshes generated with the CUBIT [30] and Tetgen [31] mesh generation packages. We consider the following geometries: distduct, gear [26], and cube. In addition, we consider meshes on

Solver	Parameter	Value
Steepest descent	Backtracking parameter	0.2
	Reduction parameter	0.5
Conjugate gradient	Backtracking parameter	0.2
	Reduction parameter	0.5
Quasi-Newton and Feasible Newton	Backtracking parameter	0.5
	Reduction parameter	0.5
Trust region	Backtracking parameter	0.5
	Reduction parameter	0.25
	Initial trust region radius	1000
	Maximum trust region radius	10^{20}

Table 1 Default parameter values for the various solvers as implemented in Mesquite.

the foam geometry in our Conclusions and Future Work section. Sample meshes are shown in Figure 1. In the first three experiments, we study the effects of three different problem parameters on the improvement of the objective function. The problem parameters of interest are: the problem size, the initial mesh configuration, and the grading of mesh elements. Table 2 shows the the parameters that are held constant and those that are varied for each experiment.

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 [32, 33]. 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.

Because the objective functions used for our experiments are nonconvex, 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. Unless indicated otherwise, the optimization methods converged to the same optimal mesh.

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 aspect ratio quality metric), and parameter values of interest (such as the magnitude of the 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 majority of our experiments were run for 300 seconds before they were terminated in order to obtain the results; this was an appropriate length of time for the experiments in that most meshes with up to 500,000 elements can be smoothed in under 300 seconds using the various solvers. When the fastest solvers were employed, meshes with 500,000 elements were smoothed in approximately 100 seconds. For the first experiment which focuses on problem size scaling, we also smoothed meshes containing approximately 1 million and 3 million elements. These meshes took approximately 600 and 1800 seconds to converge to an optimal mesh with the fastest solvers. Hence, we ran this experiment for 3000 seconds before termination to ensure that all solvers had enough time to converge to the optimal mesh in the various contexts. The machine employed for this study is equipped with an Intel P4 processor (3.33 GHz). The 32-bit machine has 4GB of RAM, a 2MB L2 cache, and runs Linux.

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. A series of meshes were generated for the distduct and gear geometries shown in Figures 1(a) and 1(b); for each series of meshes, the number of elements is increased from approximately 5,000 to 3,000,000 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 3 shows the initial qualities of the distduct meshes. It also shows the final mesh qualities after the conjugate gradient and feasible Newton methods were used to perform local and global mesh quality improvement, respectively, on each mesh.

Table 4 shows the times and numbers of iterations required for the local and global mesh quality improvement solvers described above to converge to the optimal meshes on the distduct geometry. Such improvements in mesh quality were typical of the results seen in this experiment.

Besides determining the most efficient solver for a given context, we also determine the order of convergence for each solver as a function of problem size for this experiment. The order of convergence, α, is given by the following formula:

$$t = k * n^{\alpha},$$

where t is the time to convergence, n is the number of mesh vertices, and $k > 0$ is a constant. In order to determine α, a least squares fit can be computed by taking the logarithm of both sides.

We now summarize the results of performing local and global mesh smoothing for this experiment.

4.1.1 Local Smoothing

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

Experiment #	Experiment name	Parameters held constant	Parameters varied
4.1	Increasing problem size	average mesh quality	mesh size
4.2.1	Initial mesh configuration - random perturbation	mesh size	number of vertices perturbed
4.2.2	Initial mesh configuration - translation	mesh size	magnitude of vertex translation
4.3	Graded meshes	mesh size	magnitude of grading of elements

Table 2 Parameters that were held constant and varied in each of the experiments. For experiment 4.1, unperturbed CUBIT-generated gear and distduct meshes were used. We verified that the average mesh quality was approximately the same for these meshes. For experiments 4.2.1 and 4.2.2, perturbed versions of these meshes were employed. Tetgen-generated cube meshes were employed for experiment 4.3.

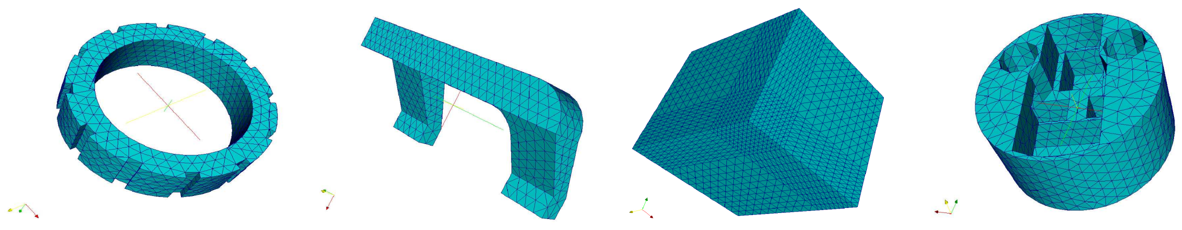

(a) Gear mesh (b) Distduct mesh (c) Cube mesh (d) Foam mesh

Fig. 1 Sample meshes on the gear, distduct, and cube geometries. Geometries (a), (b), and (d) were provided to us by Dr. Patrick Knupp of Sandia National Laboratories [26].

distduct geometry. The behavior of the solvers was identical on the gear geometry; thus, additional figures have been omitted. It was also verified that the solvers converged to the same optimal meshes for a given geometry. This is expected as the number of iterations required to converge is more or less a constant, and the time per iteration increases linearly with the number of elements used for local mesh smoothing. This is because the tetrahedral meshes are generated using CUBIT and hence are already close to optimal. There are instances where a deviation from linear scaling is seen when smoothing larger meshes. This is because they took one to two additional iterations to attain the desired quality. By computing the least squares fit of the data, we found that the value of α for local smoothing is very close to 1 for fully-converged meshes. Thus, we conclude that local smoothing scales linearly with the mesh size.

We now examine the behavior of the various solvers on the distduct meshes. 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 for minimizing (1). We define a fully-converged solution to be one which is accurate to 6 digits. For each solver, we consider smoothing with 85%, 90%, and 100%-converged solutions; the results are shown in Figure 2. These percentages specify the amount of mesh smoothing performed and is defined as

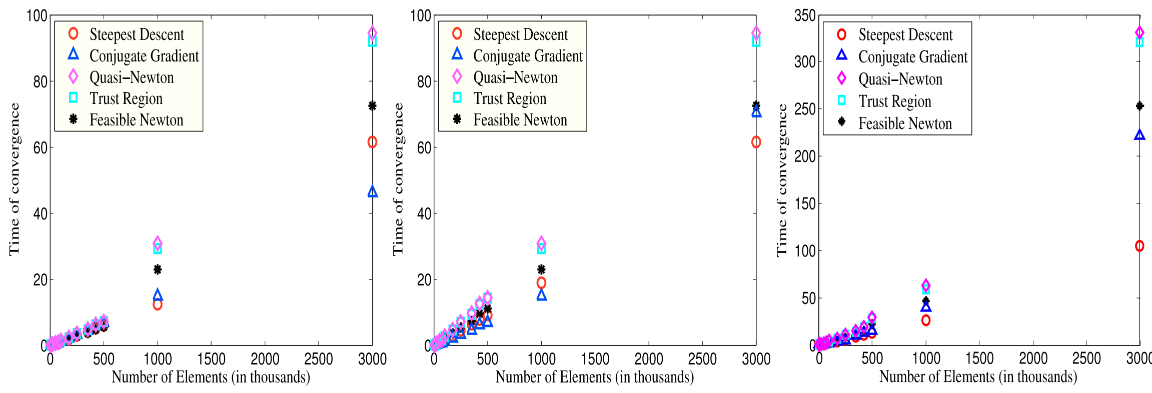
$$x = \frac{Q_i - Q_x}{Q_i - Q_{conv}},$$

where x is the percentage of smoothing which has been completed, Q_i is the initial value of the objective function, Q_x is the value of the objective function after x% of smoothing, and Q_{conv} is the value of objective function for the fully-converged mesh. Note that although inaccurate mesh smoothing can be performed, in order to determine the percentage of improvement, Q_{conv} must be determined through an initial run of accurate mesh smoothing, i.e., it is not known *apriori*. We chose 85% and 90% smoothing for inaccurate smoothing because the optimization methods are able to improve the mesh quality by the desired amount in just 2 or 3 iterations.

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 the quickest method since it took fewer iterations than the other methods. Although each iteration took a greater amount of time than for the other solvers, the amount of time to convergence was less than that required by the other solvers. The ranking of all solvers in order of fastest to slowest on the largest mesh is: conjugate gradient $<$ steepest descent $<$ feasible Newton $<$ trust region $<$ quasi-Newton. For the small 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) due to their lower computational complexity. As the number of mesh elements increased to 1 million and beyond, the gap between the steepest descent and other solvers increased. Clearly the rank-

Distduct Mesh		Smoothing	Mesh Quality (Aspect Ratio)				
# Vertices	# Elements		min	avg	rms	max	std. dev.
		Initial	1.00557	1.33342	1.35118	2.71287	0.218363
1,262	5,150	Final (local)	1.00077	1.27587	1.28932	2.83607	0.185684
		Final (global)	1.00077	1.27587	1.28932	2.83604	0.185686
		Initial	1.00156	1.31327	1.33023	5.23155	0.211758
2,347	10,181	Final (local)	1.00015	1.20092	1.21025	10.31880	0.150045
		Final (global)	1.00315	1.24561	1.25762	4.60388	0.173392
		Initial	1.00146	1.30004	1.31688	6.81820	0.209909
5,292	24,860	Final (local)	1.00281	1.22567	1.23718	6.81820	0.168317
		Final (global)	1.00280	1.22567	1.23717	6.81820	0.168334
		Initial	1.00282	1.28901	1.30397	7.84040	0.196944
10,326	50,649	Final (local)	1.00110	1.21313	1.22273	7.84040	0.152960
		Final (global)	1.00111	1.21310	1.22271	7.84040	0.152984
		Initial	1.00033	1.28333	1.29898	9.06301	0.201012
14,858	74,641	Final (local)	1.00054	1.20605	1.21582	9.06301	0.153753
		Final (global)	1.00063	1.20595	1.21572	9.06301	0.153828
		Initial	1.00070	1.28014	1.29531	10.3188	0.197718
19,602	99,895	Final (local)	1.00065	1.21742	1.22755	4.86240	0.157424
		Final (global)	1.00009	1.20054	1.20991	10.31880	0.150259
		Initial	1.00111	1.27886	1.29358	11.91700	0.194551
33,128	1,72,479	Final (local)	1.00025	1.20569	1.21472	3.44530	0.147858
		Final (global)	1.00023	1.19746	1.20601	11.91700	0.143376
		Initial	1.00090	1.27189	1.28648	13.20510	0.193164
47,361	249,975	Final (local)	1.00008	1.19900	1.20730	3.47083	0.141801
		Final (global)	1.00035	1.22353	1.23745	6.43000	0.065702
		Initial	1.00087	1.27160	1.28636	14.71870	0.194342
64,685	345,114	Final (local)	1.00038	1.19077	1.19900	13.36920	0.140402
		Final (global)	1.00027	1.18908	1.19737	13.36820	0.140705
		Initial	1.00036	1.27132	1.28586	17.51110	0.192819
80,025	429,578	Final (local)	1.00014	1.19101	1.19924	17.51110	0.140236
		Final (global)	1.00014	1.18853	1.19679	17.51110	0.140397
		Initial	1.00009	1.27055	1.28513	18.55920	0.193054
92,316	498,151	Final (local)	1.00004	1.18949	1.19770	18.55920	0.139968
		Final (global)	1.00009	1.18757	1.19580	18.55920	0.140090
		Initial	1.00014	1.27476	1.28717	5.56563	0.178268
184,006	995,308	Final (local)	1.00010	1.20567	1.21391	2.76560	0.141226
		Final (global)	1.00009	1.20567	1.21392	2.76551	0.141230
		Initial	1.00004	1.24995	1.26111	3.53242	0.167390
532,789	2,951,272	Final (local)	1.00003	1.19667	1.20448	3.00381	0.136930
		Final (global)	1.00004	1.19646	1.20428	2.96445	0.136953

Table 3 Initial and final mesh quality after smoothing the distduct meshes with the conjugate gradient method for local smoothing and the feasible Newton method for global smoothing. Results from different solvers were chosen for the local and global smoothing contexts, as these two methods represent the fastest solvers in the two contexts, respectively.



(a) 85%-converged solution (b) 90%-converged solution (c) 100%-converged solution

Fig. 2 Local Smoothing: Smoothing of the gear meshes to various convergence levels: (a) 85%-converged solution; (b) 90%-converged solution; (c) 100%-converged solution.

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 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 conjugate gradient method is superior to the steepest descent method since it uses gradient history to determine the optimal vertex position. The remaining methods also use the Hessian of the objective function to move wach 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 difference between the fastest two solvers is approximately 16%, whereas the difference between the fastest and the slowest solvers is approximately 66%.

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 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. For example, for the distduct mesh with 500,000 elements and approximately 100,000 vertices, each additional vector has an approximate length of 300,000. Since each component of the vector consumes four bytes of memory, storing one additional vector corresponds to a 1.1MB increase in memory. In addition, the number of iterations remained the same for all solvers. This is because we are moving only one vertex at a time. All of the solvers move the vertex in almost the same direction and magnitude. Hence, the additional complexity and accuracy is not necessary in the context of local smoothing. The difference between the fastest two solvers is approximately 125%, whereas the difference between the fastest and the slowest solvers is approximately 225%.

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.

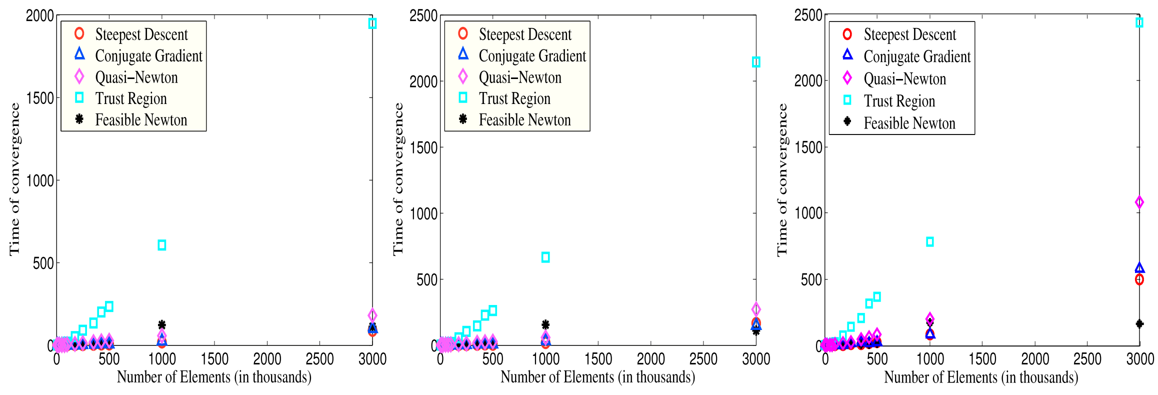
4.1.2 Global Smoothing

There are a few differences between the results from the local and global smoothing contexts. Figure 3 shows that the method scales superlinearly with respect to the problem size for all the solvers except the trust region method. The value of α for the least squares fit was close to 1.2. This can be attributed to the inherent computational complexity of the solvers. As the number of unknowns increases, the solvers scale superlinearly. In global smoothing, the number of unknown variables is proportional to the number of interior vertices in the mesh. The trust region solver is an exception to this trend as the movement of the vertices is constrained by the trust region. Additional trust-region experiments were performed which determined that as the initial trust region radius was increased, the convergence of the trust region solver to the optimal mesh was significantly faster. This demonstrates that the trust region radius is limiting the progress of the optimization method, and, hence, the solver takes smaller steps toward the optimal mesh resulting in linear scaling of the method.

In Figure 3, global mesh smoothing results for smoothing a gear mesh are shown; such results are typical for global mesh smoothing. As in the local case, we examine the nature of solvers at various amounts of smoothing. When only 85% smoothing is required, the rank ordering of solvers from fastest to slowest is steepest descent $<$ conjugate gradient $<$ feasible Newton $<$ quasi-Newton $<$ trust region. Typically, we see that the fesible Newton method, although slow in the begining, ends up being the fastest solver closer to convergence. This can be seen in the plots for 90% smoothing (see Figure 3(b). The rank ordering remains the same except for feasible Newton which emerges as the fastest solver in almost all cases. The feasible Newton method uses the Hessian to compute the direction. Although it is more computationally complex than the gradient-based methods, it is also more accurate. Despite the fact that each iteration takes a greater amount of time, only a few iterations are necessary for convergence. Hence the total amount of time required to convergence is lower than that of the gradient-based methods. The rank ordering in this case is feasible Newton $<$ steepest descent $<$ conjugate gradient $<$ quasi-Newton $<$ trust region. When 100% smoothing is required, feasible Newton continues to be the fastest solver. The difference between the fastest two solvers is approximately 150%, whereas the difference between the fastest and slowest solvers is approximately 2500%. Figure 3 illustrates that the trust region method takes significantly longer than the other four solvers to converge to the optimal mesh.

Vertices	Elements	Smoothing	T_{85}		T_{90}		T_{100}	
			Time	Iterations	Time	Iterations	Time	Iterations
1,262	5,150	Local	0.02	2	0.03	3	0.03	3
		Global	0.03	2 (14)	0.07	3 (22)	0.07	3 (22)
2,347	10,181	Local	0.04	2	0.06	3	0.06	3
		Global	0.10	2 (25)	0.20	3 (30)	0.20	3 (30)
5,292	24,860	Local	0.13	2	0.19	3	0.41	5
		Global	0.30	2(19)	0.60	3 (29)	0.60	3 (29)
10,326	50,649	Local	0.27	2	0.27	3	0.86	5
		Global	0.65	2 (25)	1.30	3 (37)	1.30	3 (37)
14,858	74,641	Local	0.77	3	0.77	3	1.17	5
		Global	0.85	2 (26)	1.70	3 (39)	1.70	3 (39)
19,602	99,895	Local	1.23	3	1.23	3	1.88	5
		Global	11.40	2 (29)	2.77	3 (43)	2.77	3 (43)
33,128	172,479	Local	2.16	3	2.16	3	3.31	5
		Global	2.52	2 (36)	5.02	3 (52)	5.02	3 (52)
47,361	249,975	Local	3.30	3	3.30	3	5.05	5
		Global	3.92	2 (40)	7.77	3 (61)	7.77	3(61)
64,685	345,114	Local	4.56	3	4.56	3	9.47	5
		Global	5.32	2 (23)	10.57	3 (35)	10.57	3 (35)
80,025	429,578	Local	6.19	3	6.19	3	12.93	6
		Global	6.98	2 (23)	13.84	3 (30)	13.84	3 (30)
92,316	498,151	Local	6.92	3	7.98	4	14.35	6
		Global	8.05	2 (21)	15.96	3 (31)	15.96	3 (31)
184,006	995,308	Local	46.17	3	14.91	3	38.94	6
		Global	121.25	8 (61)	155.63	10 (72)	172.17	11 (79)
532,789	2,951,272	Local	46.17	3	70.42	4	221.86	10
		Global	111.72	3 (31)	111.72	3 (31)	165.48	4 (42)

Table 4 Timing results for the fastest local (i.e., conjugate gradient) and global (feasible Newton) mesh smoothing methods for inaccurate and accurate mesh smoothing. The table shows the time taken to smooth the distduct meshes for the respective solvers. T_{85}, T_{90}, and T_{100} are the times taken to achieve 85%, 90%, and 100% of the locally optimal solution. The number of iterations represent the number of outer iterations for each solver. The numbers in parentheses represent the total number of linear conjugate gradient iterations (i.e., the number of inner iterations) for the feasible Newton solver. For local smoothing using the conjugate gradient method, the number of inner iterations is always one and hence is not shown.



(a) 85% Smoothing (b) 90% Smoothing (c) 100% Smoothing

Fig. 3 Global Smoothing: Smoothing of the gear meshes to various convergence levels: (a) 85%-converged solution; (b) 90%-converged solution; (c) 100%-converged solution.

4.2 Initial mesh configuration

In order to investigate the effect that the initial mesh configuration (as measured by the distance from the optimal mesh) had on the performance of the five solvers, a series of perturbed meshes, based on the 500,000 element distduct and gear meshes from the previous experiment, were designed. In particular, the meshes were completely smoothed initially. 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 randomly chosen subsets of vertices of size 5%, 10%, 25%, and 100% of the number of 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. α_v was chosen such that it did not tangle the mesh. If the random value generated for α_v was too large, it was gradually decreased until the mesh was untangles. α_v values range from 0.001 to 1.55.

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 $s = [1\ 0\ 0]^T$. In addition, 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.

4.2.1 Random Perturbations

For this experiment, the vertices of the meshes to be were perturbed from the CUBIT-generated meshes. Thus, the initial meshes are of poorer quality. Starting with these poor quality meshes, i.e., far away from an optimal mesh, had a very significant impact on the performance of the solvers. Table 5 shows the results of smoothing a highly perturbed disduct mesh with the various solvers.

4.2.1.1 Local smoothing

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. However, the conjugate gradient method performs better than the steepest descent method when the quality of the input mesh is reasonably good.

When poor quality initial meshes are smoothed, the initial movement of the vertices is large. However, once several iterations of smoothing have been performed, the vertex movement is small and can be obtained efficiently obtained through the use of Hessian-based solvers. The gradient-based solvers are less accuarate, and hence more iterations are required to converge to the optimal mesh. As a result, they usually take more time than Hessian-based solvers do to converge. In most cases, because the perturbation was large, vertices had to move by large distance. As a result, the performance of steepest descent was the best (which was also due to the lower complexity of the algorithm). When the perturbations are small, the fine-scale smoothing requirements imply that the Hessian-based methods will converge faster. This was indeed seen in the small perturbation case. 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 4(a) shows typical objective function versus time plots for our experiments.

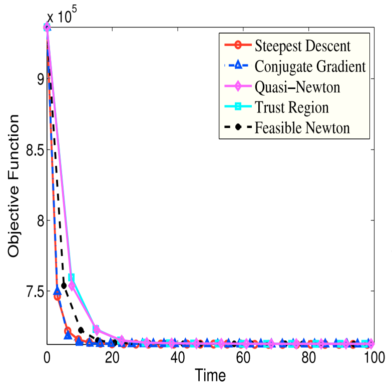
The behavior of the trust region method was distinctly different than that of the other algorithms. For small perturbations from the optimal mesh, the behavior of the trust region method almost coincided with that of the other methods in the quality versus time plots. Figure 4(b) below illustrates an example of such behavior. The difference between the fastest two solvers is approximately 12%, whereas the difference between the fastest and slowest solvers is approximately 60%.

However, when the perturbations were large, the trust region method was much slower than the other methods in terms of time to convergence. This behavior is due to the constraint of the trust region bounding the maximum acceptable step length at each iteration. For large perturbations, we ran the steepest descent method for longer and confirmed that it does not converge to the same optimal mesh as the other methods. This is a nonconvex optimization problem; an optimization method may converge to any one of the local minima. In particular, it converged to an optimal mesh with a higher objective function value than the meshes obtained when other algorithms were used. The plot shown in Figure 4(c) is a good example of the poor performance of the trust region and steepest descent methods in the large perturbation case. The difference between the fastest two solvers is approximately 32%, whereas the difference between the fastest and slowest solvers is approximately 90%.

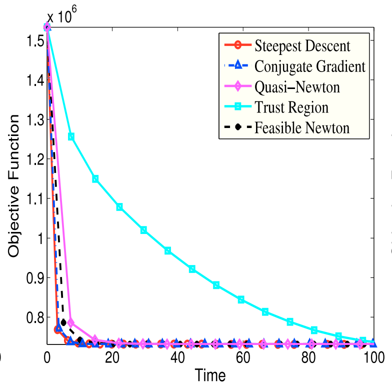
In conclusion, the rank-ordering of the optimization solvers depends upon the amount of random perturbations applied to the initial meshes in the context of local mesh smoothing. In particular, all five methods performed competitively for the small perturbation case; however, the steepest descent and 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 lesser quality.

4.2.1.2 Global smoothing

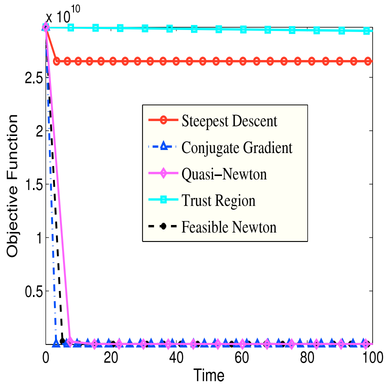
The results from global smoothing are similar to the those from local smoothing in that they can be classified into three main categories: small, medium, and large perturbation. In



(a) Small perturbations



(b) Medium perturbations

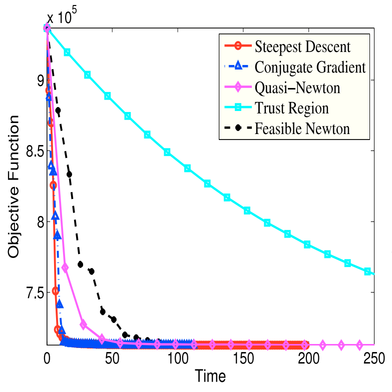


(c) Large perturbations

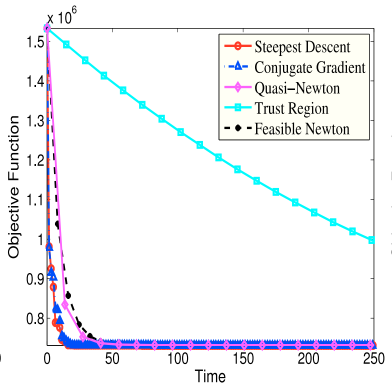
Fig. 4 Local Smoothing: Typical results from the random perturbation experiment. Results were obtained by smoothing the 500,000 element meshes. (a) Gear mesh with 10% of its vertices perturbed (Small perturbations); (b) Distduct mesh with 10% of its vertices perturbed (Medium perturbations); (c) Distduct mesh with 5% of its vertices perturbed. Observe that the scaling of the vertical axis is different in each plot (Large perturbations).

Smoothing	Algorithm	min	avg	rms	max	std. dev.
	Initial	1.00021	1.95294	244.23100	162,866.0000	244.223000
Local	Steepest Descent*	1.00009	1.51455	230.75800	162,866.0000	230.753000
	Conjugate Gradient	1.00010	1.18758	1.19581	18.5592	0.140066
	Feasible Newton	1.00014	1.18759	1.19582	18.5592	0.140035
	Trust Region**	1.00018	1.69459	239.91600	161,087.0000	239.910000
	Quasi-Newton	1.00018	1.18761	1.19583	18.5592	0.140006
Global	Steepest Descent*	1.00021	1.95294	244.23100	162,866.0000	244.223000
	Conjugate Gradient	1.00018	1.18773	1.19594	18.5592	0.139916
	Feasible Newton	1.00009	1.18757	1.19580	18.5592	0.140090
	Trust Region**	1.00021	1.95078	243.57000	162,440.0000	243.562000
	Quasi-Newton	1.00019	1.18796	1.19622	18.5592	0.140374

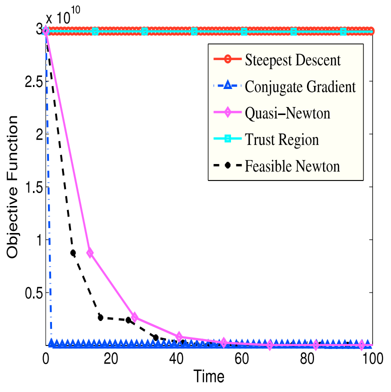
Table 5 Mesh quality results obtained by smoothing a 500,000 vertex distduct mesh with 5% of its vertices perturbed by a large amount. A '*' denotes convergence to a bad quality mesh, and a '**' denotes that the solver did not converge in 300 seconds.



(a) Small perturbations



(b) Medium perturbations



(c) Large perturbations

Fig. 5 Global Smoothing: Typical results from the random perturbation experiment. Results were obtained by smoothing the 500,000 element meshes. (a) Gear mesh with 10% of its vertices perturbed (Small perturbations); (b) Distduct mesh with 10% of its vertices perturbed (Medium perturbations); (c) Distduct mesh with 5% of its vertices perturbed. Observe that the scaling of the vertical axis is different in each plot (Large perturbations).

each case, results similar to those from local smoothing were obtained.

Typical results for global smoothing of the randomly perturbed meshes are shown in Figure 5. In comparing these results with those for local smoothing (Figure 5), we see that the results are very similar except for couple of differences. First, the trust region solver converges to the solution very slowly even in the case of a small perturbation. It can be deduced from the plots that the trust region radius restriction has a greater impact when performing global smoothing than when performing local smoothing. This is due to the increased impact that scaling of the objective function has on the performance of the trust region solver. The second difference is that the steepest descent method converges to a different minimum when global smoothing is performed. We ran the trust region solver on all meshes and verified that solver converges to same optimal meshes as did the other solvers.

4.2.2 Translation

In order to determine the effect that translation 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 and gear meshes were smoothed. Refer to Table 6 for results of smoothing highly translated disduct mesh with various solvers.

4.2.2.1 Local smoothing

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. 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 of the solvers is observed in the plots shown in Figure 6. 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. Eventually, more vertex movement occurs, and the objective function values become less predictable. After 20 iterations, the objective function values are completely different for the various solvers. However, all solvers 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. The difference between the fastest two solvers is approximately 100%, whereas the difference between the fastest and the slowest solvers is approximately 400%.

In conclusion, the optimization solvers exhibited a distinct rank ordering. In particular, the rank-ordering was as follows: steepest descent $<$ conjugate gradient $<$ feasible Newton $<$ trust region $<$ quasi-Newton.

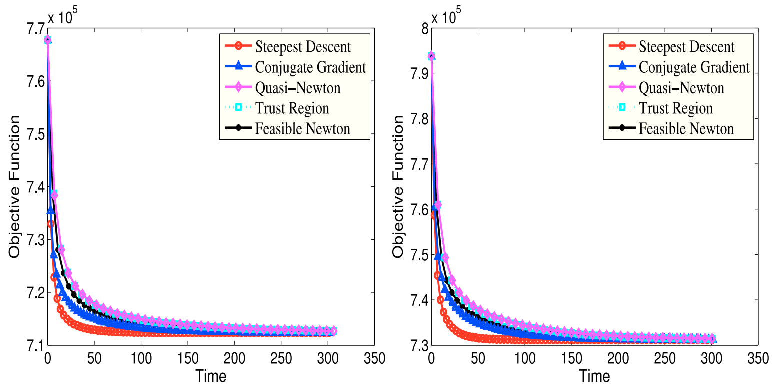
4.2.2.2 Global smoothing

For the translation meshes, the global smoothing results are completely different from the local smoothing results. In the local smoothing context, we saw that there was a definite hierarchy in the rank ordering of the solvers. In the global smoothing context, the hierarchy is not present. Instead, we see that feasible Newton eventually overtakes the other solvers and becomes the fastest solver. For larger translations, it takes longer before the feasible Newton method overtakes the other methods. The conjugate gradient and the steepest descent algorithms are almost identical in performance most of the time. The performance of the quasi-Newton and the trust region methods are in are general not as good the performances of the other methods.

Figure 7 shows the results that were explained above. You can clearly see the absence of a fixed hierarchy. Instead, during the initial iterations, the rank ordering of the solvers is: feasible Newton $<$ steepest descent $<$ conjugate gradient $<$ quasi-Newton $<$ trust region. In subsequent iterations, steepest descent trades places with conjugate gradient, and the order becomes: feasible Newton $<$ conjugate gradient $<$ steepest descent $<$ quasi-Newton $<$ trust region.

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



(a) Distduct mesh (smoothing translated vertices) (b) Gear mesh (smoothing translated vertices)

Fig. 6 Local Smoothing: Typical results for the affine perturbation experiment for local mesh smoothing. The results are for smoothing the distduct and gear meshes with 500,000 elements after all interior vertices were affinely perturbed.

Smoothing	Algorithm	min	avg	rms	max	std. dev.
	Initial	1.00024	1.22465	1.24143	41.6410	0.203427
Local	Steepest Descent	1.00008	1.18757	1.19580	18.5592	0.140091
	Conjugate Gradient	1.00009	1.18759	1.19583	18.5592	0.140108
	Feasible Newton	1.00010	1.18769	1.19593	18.5592	0.140153
	Trust Region	1.00012	1.18782	1.19606	18.5592	0.140206
	Quasi-Newton	1.00012	1.18783	1.19608	18.5592	0.140213
Global	Steepest Descent	1.00024	1.18970	1.19802	18.5592	0.140887
	Conjugate Gradient	1.00029	1.18872	1.19700	18.5592	0.140555
	Feasible Newton	1.00009	1.18757	1.19580	18.5592	0.140090
	Trust Region	1.00016	1.19787	1.20645	18.5592	0.143625
	Quasi-Newton	1.00024	1.19038	1.19873	18.5592	0.141267

Table 6 Typical mesh quality results obatined from smoothing a 500,000 vertex translated distduct mesh with the vertices translated the maximum distance possible without mesh tangling.

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, were used for the graded mesh experiment. Figure 1(c) shows 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. Table 7 shows the results of smoothing a highly graded cube mesh with the various solvers.

4.3.1 Local Smoothing

The results obtained from this experiment are shown in Figure 8. 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.

The similarity in results from local smoothing of the translated and graded meshes is seen because graded meshes we generated are similar to the translated meshes. To create a translated mesh, all the vertices were moved a certain distance in a fixed direction. To create a graded mesh, half of the vertices were moved to a corner, and fewer vertices were left behind in the opposite corner. This is similar to translating vertices from a uniform mesh from one corner to another.

In conclusion, the rank ordering of the conjugate gradient method varied as a function of time as the graded meshes were smoothed. However, the rank-ordering of the remaining four optimization solvers was as follows: steepest descent $<$ feasible-Newton $<$ trust region $<$ quasi-Newton. The difference between the fastest two solvers is approx-

Smoothing	Algorithm	Min	Avg	Rms	Max	Std. Dev.
	Initial	1.76933	3.01439	3.25715	4.57034	1.233900
Local	Steepest Descent	1.04097	2.14383	2.23283	5.76333	0.624099
	Conjugate Gradient	1.04098	2.14384	2.23283	5.76326	0.624068
	Feasible Newton	1.04098	2.14384	2.23283	5.76313	0.624061
	Trust Region	1.04098	2.14384	2.23283	5.76313	0.624061
	Quasi-Newton	1.04098	2.14384	2.23283	5.76316	0.624061
Global	Steepest Descent	1.08081	2.16411	2.24952	5.79640	0.613983
	Conjugate Gradient	1.04130	2.14489	2.23315	5.76358	0.621615
	Feasible Newton	1.04098	2.14384	2.23283	5.76313	0.624061
	Trust Region	1.04098	2.14384	2.23283	5.76313	0.624061
	Quasi-Newton	1.04103	2.14449	2.23300	5.76323	0.622458

Table 7 Mesh quality results obatined from smoothing a cube mesh with graded elements; 50% of vertices are present in 10% of the space.

imately 300%, whereas the difference between the fastest and slowest solvers is approximately 400%.

4.3.2 Global Smoothing

We saw in the context of local smoothing that the results for graded meshes were similar to those of translated meshes. This is also true within the global smoothing context. However in this case, all five methods were competitive. Feasible Newton outperformed the others, but among the remaining solvers, there was no clear winner. In some instances, steepest descent or conjugate gradient is the best, whereas in other instances, quasi Newton or trust region end up being the fastest solver.

There are some common trends that occurred between the translation and graded meshes experiment. As additional grading was introduced, feasible Newton took more iterations to surpass the other solvers. The performance of the trust region solver was directly related to the extent to which the meshes were graded. Highly graded meshes were observed to deteriorate the performance of the solver, as the movement of the vertices were constrained by the trust region.

Figure 9 shows the results from this experiment. Again, we observe the following rank ordering of the solvers. During the initial iterations, the rank ordering is: feasible Newton $<$ steepest descent $<$ conjugate gradient $<$ quasi-Newton $<$ trust region. In subsequent iterations, as in the affine perturbation case, steepest descent trades places with conjugate gradient. Hence the order is: feasible Newton $<$ conjugate gradient $<$ steepest descent $<$ quasi-Newton $<$ trust region. Again, the trust region method is surpasses all solvers except feasible newton as the fastest solver. The difference in time for converging to an optimal mesh between the fastest two solvers is approximately 75%.

4.4 Effect of Parameter Changes

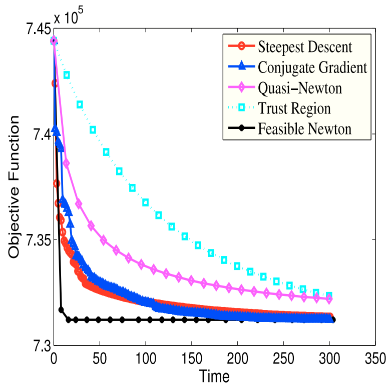
As noted in Sections 4.1-3, the trust region solver performed well on some mesh optimization problems and poorly on others. Thus, in this section, we investigate the effect of changing the default parameters that are shown in Table 1 in attempt to improve the performance of the trust region solver. In addition, we investigate the sensitivity of the other solvers to changes in the default parameter values.

Solver	Parameter	Value
Trust region	Backtracking parameter	0.5
	Reduction parameter	0.25
	Initial trust region radius	10^{7}
	Maximum trust region radius	10^{20}

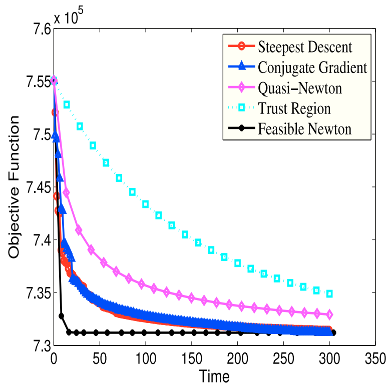
Table 8 The most efficient combination of trust region parameters for global smoothing for the majority of the meshes considered.

For each of the solvers, we changed the default values to determine the ones that resulted in the fastest convergence time. We found that the performance of the steepest descent, conjugate gradient, feasible Newton, and quasi-Newton solvers were rather insensitive to these changes in the parameter values. The plots of objective function versus time were nearly coincident with the corresponding plots based on the default parameter values. However, the performance of the trust region method was very sensitive to the parameter changes. In particular, the rankings of the various solvers were significantly influenced within the context of global smoothing. In the remainder of the section, we provide additional details as to the changes made to the default parameters and how they influenced the trust region solver performance.

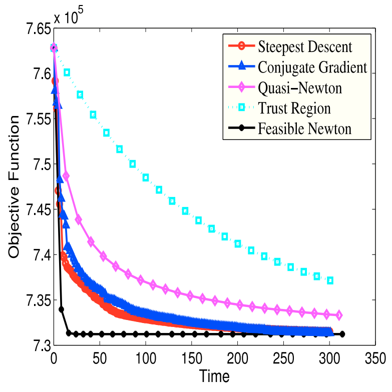
Previously, the trust region solver performed poorly for several large perturbation test cases. Although this was the case for both local and global mesh smoothing, it was especially the case for global smoothing. This is because for global mesh smoothing, all of the element mesh qualities are squared and added together to obtain the objective function. Thus, the value of the objective function increases as the number of mesh elements increases (subject to the quality of individual elements remaining the same). Since the objective function is not scaled according to the number of mesh elements, the optimization solver needs to take the scaling of



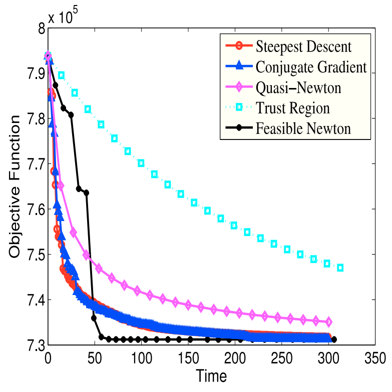
(a) Small affine perturbation



(b) Medium affine perturbation



(c) Large affine perturbation



(d) Huge affine perturbation

Fig. 7 Global Smoothing: Typical results for the affine perturbation experiment for global mesh smoothing. The results are for smoothing the gear meshes with 500,000 elements after all interior vertices were affinely perturbed. Observe that the scaling of the vertical axis is different for each plot.

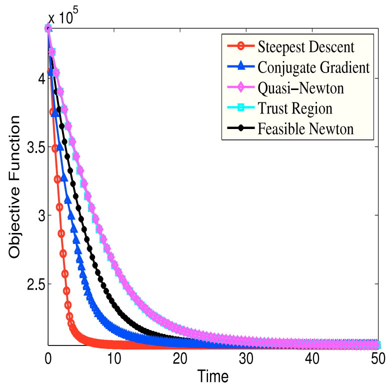


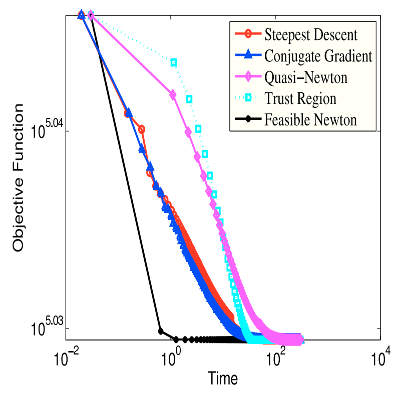
Fig. 8 Local Smoothing: A typical result for smoothing a cube mesh with graded elements; 50% of its vertices are present in 10% of the space.

the objective function into account. However, the initial trust region radius in Mesquite is not chosen based on the problem scaling. Thus, we performed additional experiments in which we varied the value of the initial trust region radius from 1000 to 10^7 (as shown in Table 8). In addition, we varied the number of inner iterations from 1 to 5 to allow the trust region to increase or decrease on each iteration. (Note this was necessary due to the way in which the trust region solver is coded in Mesquite.)

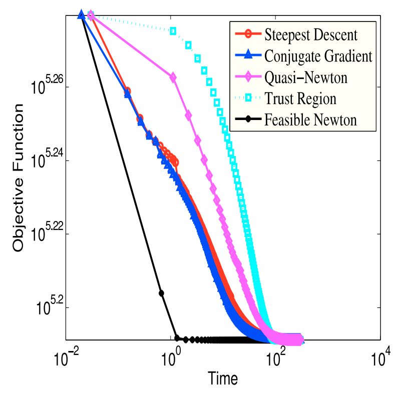
Figs. 10, 11, and 12 illustrate the resulting performance of the trust region solver as compared to the performances of the other solvers for global mesh smoothing in several contexts. As shown in Fig. 10, the trust region solver quickly converged to an optimal mesh after the changes were made to the trust region solver when used on a randomly-perturbed distduct mesh. Fig. 11 shows that the trust region solver smoothed the translated gear mesh quickly when the new parameters were used. As shown in Fig. 12, the trust region method smoothed the graded cube mesh faster than the other methods. These results were selected because they demonstrate the greatest improvement possible for the trust region solver when these parameters were varied and set equal to the values in Table 8. In many cases, the performance of the trust region solver was as good as the performance of the feasible Newton solver.

5 Conclusions and Future Work

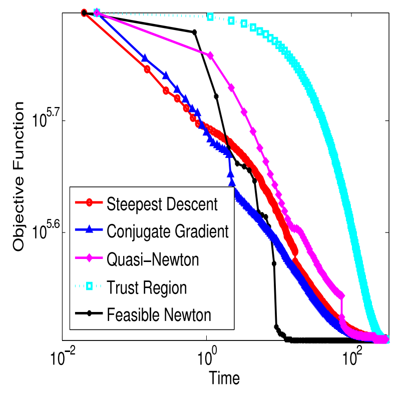
The main results of this study are as follows: (1) the behavior of the optimization solvers, i.e., their rank ordering, is influenced by the degree of accuracy desired in the solution and the size of the mesh; (2) most of the time, the gradient-based local mesh optimization solvers exhibited superior performance compared to that of the Hessian-based local mesh optimization solvers; (3) for global mesh smoothing, the class of optimization solver (i.e., gradient or Hessian) - which ex-



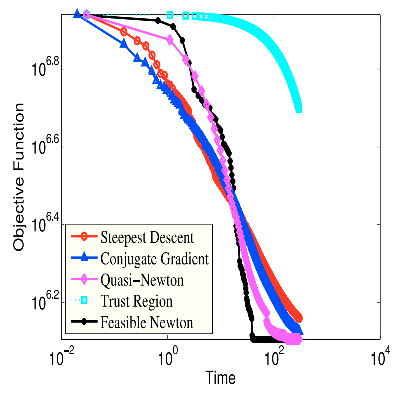
(a) Low grading



(b) Medium grading



(c) Fairly high grading



(d) High Grading

Fig. 9 Global Smoothing: Typical mesh smoothing results for the graded meshes. Note that when the mesh is highly graded, trust region method doesn't converge as fast as other methods. Observe that both axes are on a logarithmic scale.

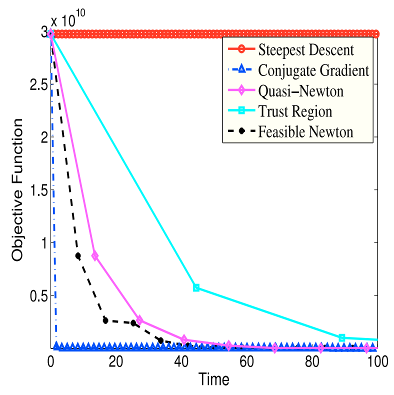


Fig. 10 Global Smoothing: Result for smoothing a distduct mesh with 5% of its vertices randomly perturbed. Notice how the performance of the trust region method is improved after the default values of the initial trust region radius and the number of inner iterations are changed to those in Table 8. Compare this figure with Fig. 5(c).

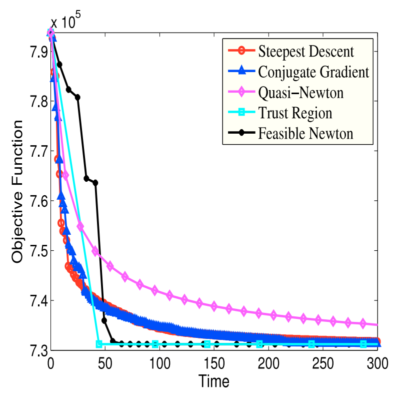


Fig. 11 Global Smoothing: Result for smoothing a translated gear mesh after the trust region method parameter values were changed as shown in Table 8. Compare this figure with Fig. 7(d).

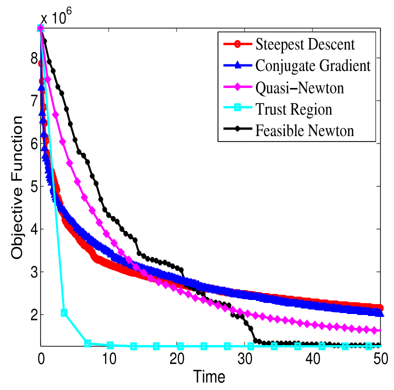


Fig. 12 Global Smoothing: Result for smoothing a cube mesh with graded elements (with 50% of its vertices present in 10% of the space) after the trust region method parameter values were changed as shown in Table 8. Compare this figure with Fig. 9(d).

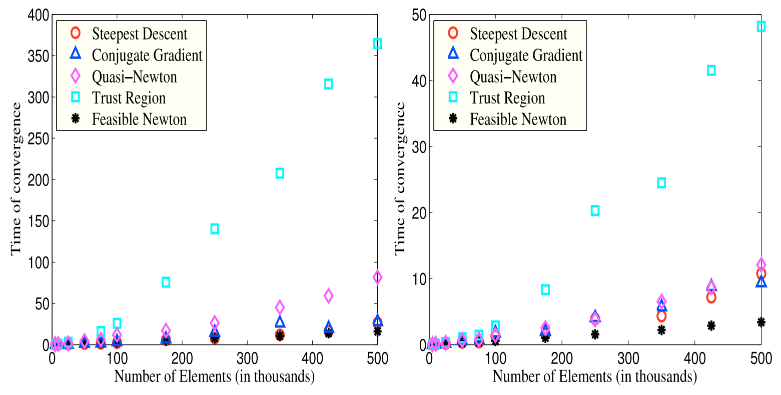
hibited superior performance is context-dependent; (4) the rank-ordering of the optimization solvers depends on the amount of random perturbation applied; (5) the rank-ordering of the local mesh optimization solvers exhibits a definite hierarchy on the affine perturbation meshes; (6) feasible Newton exhibited superior performance when compared to the other global mesh optimization solvers on the affine perturbation meshes; (7) the rank ordering of the majority of the local mesh optimization solvers is the same for graded meshes; however, the rank of the conjugate gradient method is a function of time; (8) feasible Newton exhibited superior performance when compared to the other global mesh optimization solvers on the graded meshes.

Table 9 provides a brief summary of results, which can be used a guideline for choosing the most efficient smoothing method. For a typical user, we recommend use of the conjugate gradient solver if local mesh smoothing is to be performed. Though the steepest descent solver is faster than other solvers in many cases, it is more likely to converge to a local minimum corresponding to a poorer quality mesh. If local smoothing of a graded mesh is desired, the steepest descent method is recommended instead. For graded meshes, this technique does not converge to a mesh of poorer quality. If, on the other hand, global mesh smoothing is desired, the feasible Newton solver is recommended. For graded meshes, the feasible Newton and trust region methods are the fastest. However, the feasible Newton method requires the Hessian of the objective function which might be difficult to obtain. It also requires a longer set-up time. We also recommend that the Mesquite developers initialize the trust region radius as a function of the mesh size, especially for global mesh smoothing.

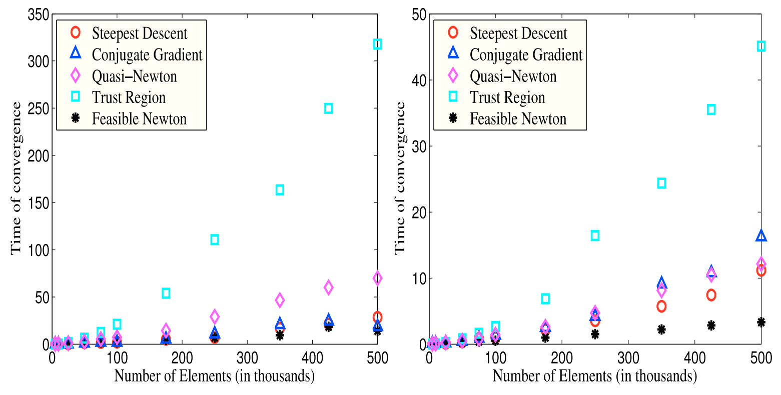
The results in this study are specific to mesh quality improvement of unstructured tetrahedral meshes via five optimization solvers, namely, the steepest descent, Polack-Ribiere conjugate gradient, quasi-Newton, trust-region, and feasible Newton methods, with mesh quality measured according to the aspect ratio quality metric. Note that we have employed a black-box approach to mesh smoothing in this paper. We have used the default parameters in Mesquite to run our numerical experiments. It is possible that the results of this study would change if the parameter values of the solvers were allowed to change. Because, vertex ordering has been shown to play an important role in convergence of the Feasnewt solver when used for local mesh optimization [34], we plan to investigate the effect of vertex ordering in the future. We also plan to examine 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. Figure 10 shows an example where the choice of quality metric influences the results. In particular, when the inverse mean ratio mesh quality metric is instead used to smoothe the mesh, the rank ordering of the solver changes. Figures 13(a) to 13(d) show that the choice of the mesh quality metric influenced the global smoothing results on the gear and foam meshes. In particular, it influenced the relative ranking of the quasi-Newton solver.

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. R. Bank and A. Sherman and A. Weiser, *Refinement algorithms and data structures for regular local mesh refinement*, In: R. Stepleman et al. (ed.) Scientific Computing. pp. 3–17. IMACS, Amsterdam (1983).
9. C. Ollivier-Gooch, *Multigrid acceleration of an upwind Euler solver on unstructured meshes*, AIAA Journal, 33: 1822-1827, 1995.
10. M. Rivara, *Mesh refinement processes based on the generalized bisection of simplices*, SIAM J. Numer. Anal., 21: 604-613, 1984.
11. E. de L'isle and P. George, *Optimization of tetrahedral meshes*, In: I. Babuska and W. Henshaw and J. Oliger and J. Flaherty and J. Hopcroft and T. Tezduyar (eds.) Modeling, Mesh Generation and Adaptive Numerical Methods for PDEs, vol. 72, pp. 97–127. Springer (1995).
12. H. Edelsbrunner and N. Shah, *Incremental topological flipping works for regular triangulations*, in Proceedings of the 8th ACM Symposium on Computational Geometry, pp. 43-52, 1992.
13. B. Joe, *Three-dimensional triangulations from local transformations*, SIAM J. Sci. Stat. Comp., 10: 718-741, 1989.
14. B. Joe, *Construction of three-dimensional improved-quality triangulations using local transformations*, SIAM J. Sci. Comput., 16: 1292-1307, 1995.
15. 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.
16. S. Canann and M. Stephenson and T. Blacker, *Optismoothing: An optimization-driven approach to mesh smoothing*, Finite Elem. Anal. Des., 13: 185-190, 1993.
17. V. Parthasarathy and S. Kodiyalam, *A constrained optimization approach to finite element mesh smoothing*, Finite Elem. Anal. Des., 9: 309-320, 1991.
18. P. Knupp and L. Freitag, *Tetrahedral mesh improvement via optimization of the element condition number*, Int. J. Numer. Meth. Eng., 53: 1377-1391, 2002.
19. L. Freitag and P. Plassmann, *Local optimization-based simplicial mesh untangling and improvement*, Int. J. Numer. Meth. Eng., 49: 109-125, 2000.



(a) Gear mesh, aspect ratio quality metric (b) Gear mesh, inverse mean ratio quality metric



(c) Foam mesh, aspect ratio quality metric (d) Foam mesh, inverse mean ratio quality metric

Fig. 13 Global Smoothing: Results from smoothing the gear and foam meshes of various sizes. The choice of mesh quality metric influences the smoothing results. In (b) and (d), the quasi-Newton solver ranks better than in (a) and (c). When the inverse mean ratio metric is used to measure the quality of a mesh, the quasi-Newton solver performs relatively better.

Experiment #	Experiment	Local Smoothing Fastest Method	Global Smoothing Fastest Method
4.1	Increasing problem size	conjugate gradient	steepest descent (inaccurate) feasible Newton (accurate)
4.2.1	Initial mesh configuration - random perturbation	steepest descent (small pert.) conjugate gradient (large pert.)	steepest descent (small pert.) conjugate gradient (large pert.)
4.2.2	Initial mesh configuration - translation	steepest descent	feasible Newton
4.3	Graded meshes	steepest descent	feasible Newton

Table 9 Summary of key results from each of the experiments. Note the results from the translation and graded mesh experiments are very similar. ’Inaccurate’ refers to inaccurate smoothing, and ’accurate’ refers to accurate smoothing; ’small pert.’ refers to a small perturbation, whereas ’large pert.’ refers to a large perturbation.

20. 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.
21. P. Zavattieri, *Optimization strategies in unstructured mesh generation*, Int. J. Numer. Meth. Eng., 39: 2055-2071, 1996.
22. 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.
23. J. Nocedal and S. Wright, *Numerical Optimization*, Springer-Verlag, 2nd Edition, 2006.
24. T. Munson, *Mesh Shape-Quality Optimization Using the Inverse Mean-Ratio Metric*, Mathematical Programming, 110: 561-590, 2007.
25. 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.
26. P. Knupp, Sandia National Laboratories, Personal communication, 2009.
27. P. Knupp, *Algebraic mesh quality metrics*, SIAM J. Sci. Comput., 23:193-218, 2001.

28. L. Armijo, *Minimization of functions having Lipschitz-continuous first partial derivatives*, Pacific Journal of Mathematics, 16:1-3, 1966.
29. C.T. Kelley, *Solving Nonlinear Equations with Newton's Method*, SIAM, Philadelphia, Pennsylvania, 2003.
30. Sandia National Laboratories, CUBIT Generation and Mesh Generation Toolkit, http://cubit.sandia.gov/.
31. H. Si, TetGen - A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator, http://tetgen.berlios.de/.
32. 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.
33. 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.
34. 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.