A COMPARISON OF OPTIMIZATION SOFTWARE FOR MESH SHAPE-QUALITY IMPROVEMENT PROBLEMS[1]

Lori Freitag,[†] Patrick Knupp,[*] Todd Munson,[†] and Suzanne Shontz[‡]

[†] *Mathematics and Computer Science Division*
Argonne National Laboratory, Argonne, IL 60439
freitag,tmunson@mcs.anl.gov.

[*] *Parallel Computing Sciences Department*
Sandia National Laboratories, Albuquerque, NM 87185
pknupp@sandia.gov

[‡] *Center for Applied Mathematics*
Cornell University, Ithaca, NY 14853
shontz@cam.cornell.edu

ABSTRACT

Simplicial mesh shape-quality can be improved by optimizing an objective function based on tetrahedral shape measures. If the objective function is formulated in terms of all elements in a given mesh rather than a local patch, one is confronted with a large-scale, nonlinear, constrained numerical optimization problem. We investigate the use of six general-purpose state-of-the-art solvers and two custom-developed methods to solve the resulting large-scale problem. The performance of each method is evaluated in terms of robustness, time to solution, convergence properties, and scalability on several two- and three-dimensional test cases.

Keywords: mesh smoothing, optimization, mesh quality improvement, mean ratio

1. INTRODUCTION

Node point movement methods are effective at improving both structured and unstructured meshes.

[1]THE WORK OF THE FIRST, THIRD, AND FOURTH AUTHORS WAS SUPPORTED BY THE MATHEMATICAL, INFORMATION, AND COMPUTATIONAL SCIENCES DIVISION SUBPROGRAM OF THE OFFICE OF ADVANCED SCIENTIFIC COMPUTING RESEARCH, U.S. DEPARTMENT OF ENERGY, UNDER CONTRACT W-31-109-ENG-38. THE WORK OF THE SECOND AUTHOR WAS SUPPORTED BY THE DEPARTMENT OF ENERGY'S MATHEMATICS, INFORMATION, AND COMPUTATIONAL SCIENCES PROGRAM (SC-31) AND WAS PERFORMED AT SANDIA NATIONAL LAB ORATORIES. SANDIA IS A MULTIPROGRAM LABORATORY OPERATED BY SANDIA CORPORATION, A LOCKHEED MARTIN COMPANY, FOR THE U.S. DEPARTMENT OF ENERGY UNDER CONTRACT DE-AC-04-94AL85000. THE FOURTH AUTHOR WAS SUPPORTED BY ARGONNE WHILE VISITING AS A RESEARCH INTERN AND BY THE NATIONAL PHYSICAL SCIENCE CONSORTIUM.

These methods use an optimization algorithm or heuristic to improve the quality of the elements adjacent to vertices being repositioned. The most commonly used mesh-smoothing technique is Laplacian smoothing, which relocates a single point to the geometric center of its adjacent vertices. This technique is computationally inexpensive and simple to implement, but it can produce meshes with invalid or poor-quality elements [11]. To address these problems, researchers have developed optimization-based methods that guarantee mesh quality improvement. These methods are formulated in terms of the design variables (one or more vertices to be repositioned), an improvement goal (the quality metric, objective function, and constraints), and the algorithm used to calculate an optimal solution.

Most work in optimization-based smoothing repositions one vertex at a time. A number of sweeps over the adjustable vertices are performed to achieve overall improvement in the mesh. The quality metrics optimized range from *a priori* geometric criteria [27, 12] and algebraic quality metrics [16, 17] to *a posteriori* metrics that minimize solution error indicators [2]. An objective function is then defined based on a quality metric to meet various improvement goals. For example, to improve the average quality of mesh elements, one uses an ℓ_1 or ℓ_2 norm [17]; and to improve the worst quality element, one uses an ℓ_∞ norm [12]. The optimization methods employed include conjugate gradient techniques [19], simplex methods [13], and active-set algorithms [12].

Similar techniques can be used to reposition many vertices simultaneously. However, a solution to the resulting nonlinear, constrained optimization problem is more difficult to calculate as the number of design variables increases. Several methods have been used to solve this problem for both structured and unstructured grids.

For structured mesh generation using direct optimization techniques [3, 6] researchers have employed conjugate gradient [26] and truncated Newton [9] methods. Several strategies have been investigated for unstructured meshes. For example, White and Rodrigue use a potential energy function defined on the grid nodes to push them away from each other by using a steepest descent optimization procedure [30]. The method incorporates edge swapping, retriangularization, and Laplacian smoothing to achieve a final mesh. Amezua et al. developed a length constraint method in which a user-defined density function determines the ideal length of elements within a given region [1]. An error function providing the difference between the ideal and actual edge lengths for a patch of nodes is defined and minimized by using a quasi-Newton approach. Parthasarthy and Kodiyalam optimize the ℓ_2 norm of the element aspect ratios and constrain each element to maintain positive volume [25]. The minimization is performed by using a modified feasible direction method to find a search direction at each iteration. Knupp optimizes the condition number of tetrahedral and hexahedral element meshes using a conjugate gradient method [19]. Other researchers have used steepest descent [31], quadratic programming [7], and conjugate gradients [15] to solve related problems.

While several techniques have been advocated to solve the problem of simultaneously repositioning many vertices to improve mesh quality, it is impossible to compare them because different merit functions are used on varying test cases. To address this issue, we conducted a formal study of several existing optimization methods using a consistent problem formulation similar to that of Parthasarthy and Kodiyalam [25]. The problem is described in more detail in Section 2. We then consider eight solvers: six publicly available, general-purpose software packages and two methods designed specifically for the mesh quality improvement problem. In Section 3, we give an overview of the optimization techniques used in our study. In Section 4, we present numerical results and analyze the effectiveness of each solver. In particular, we examine the effect of initial mesh quality on algorithm performance and evaluate the ability of the algorithms to solve several two- and three-dimensional test cases. For the most promising algorithms, we study the convergence histories to determine if early termination is an option to reduce computational costs and investigate their scalability as problem size increases. In Section 5, we summarize our findings and rank the algorithms from most promising to least effective.

2. PROBLEM FORMULATION

Meshes can be improved with respect to any number of quality metrics including shape, size, alignment, solution error, or combinations of these. To keep the research reported in this paper of manageable size, we investigate the behavior of the optimization solvers using a shape metric and leave investigation of solver behavior using the other types of metrics for future work. Shape metrics are important because they can be used to control one of the most important properties of a finite element mesh, namely element skew and aspect ratio. Among the various shape metrics, we have selected the mean ratio metric [22]. Since a variety of shape measures have been shown to be equiva-

lent in the sense that all are zero when the tetrahedral element is flat and approach unity for an equilateral tetrahedron [21], it is likely that the behavior of the optimization solvers in this comparison is representative of the solver behavior if other shape metrics were used, but we have not verified this. We further limit our investigation to the optimization of meshes for the purpose of creating isotropic elements. This determines a fixed weight matrix W in the formulation of the mean ratio below. The behavior of the solvers in this study may differ when applied to optimize anisotropic meshes using a different W. However, preliminary experiments with the identity weight matrix and FeasNewt algorithm described in Section 3 indicate a minimal impact on performance when compared to the same method with the weight matrix below.

To define the mean ratio metric, let ϵ be a tetrahedral element with vertex coordinates x_0, x_1, x_2, and x_3, and define a matrix A such that the three edge vectors emanating from vertex zero form the columns of the matrix. That is, $A = [x_1 - x_0, \quad x_2 - x_0, \quad x_3 - x_0]$. The mean ratio measure is formulated in terms of A as

$$\eta = \frac{3(\alpha)^{\frac{2}{3}}}{\| A \|_F^2},$$

where $\alpha = det(A)$ and $\| \cdot \|_F$ signifies the Frobenius matrix norm. The mean ratio approaches zero for nearly flat elements and is unity for a right-angled tetrahedron.

Following the ideas in [11], we reformulate the mean ratio metric proposed in [22] so that it attains the maximum value for an equilateral tetrahedron. To do so, we introduce a weight matrix W,

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

which is formed from the vertex coordinates of a unit equilateral tetrahedron. We note that the use of a weight matrix, W, creates a flexible metric which can be referenced to any ideal element; for example, anisotropic elements commonly found in boundary layer flows.

Let T be the matrix defined by $T = AW^{-1}$ so that T is the identity matrix when the element is equilateral and $\tau = det(T)$. The reformulated mean ratio measure is then

$$\mu = \frac{3(\tau)^{\frac{2}{3}}}{\| T \|_F^2}.$$

This measure ranges from zero to unity, with zero indicating a "flat" element and unity an equilateral tetrahedron. We note that this measure is equivalent to the weighted condition number measure [18].

One can also derive a weighted mean ratio measure for triangular elements referenced to an equilateral triangle,

$$\mu = \frac{2\tau}{\| T \|_F^2},$$

where the weight matrix is

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

For triangles, the mean ratio measure is identical to the condition number of T because for 2×2 matrices the Frobenius norm of T^{-1} equals the Frobenius norm of T divided by the absolute value of the determinant of T.

The simplest ℓ_2 objective function one can construct from this measure is formed by taking the inverse mean ratio so that each term in the objective function ranges from unity to infinity. By doing so, we create a "barrier" against mesh inversion. The optimization algorithms presented in this paper thus seek to find the set of free node positions (x_j, y_j, z_j), $j = 1, \ldots, J$, that minimize

$$F(\ldots, x_j, y_j, z_j, \ldots) = \sum_{\epsilon_k} \mu^{-1}(\epsilon_k),$$

where the sum extends over all of the elements ϵ_k in the mesh. Boundary nodes are assumed to be fixed. The objective function is nonlinear because it consists of sums of terms of quadratic functions over polynomial functions.

We define the feasible region to be the set of node locations for which all the tetrahedra in the mesh have positive volume, and we assume this region

is nonempty. The resulting optimization problem

$$\begin{array}{ll} \min & \sum_{\epsilon_k} \mu^{-1}(\epsilon_k) \\ \text{subject to} & \tau(\epsilon_k) \geq \beta \text{ for all } \epsilon_k \end{array}$$

where $\beta > 0$ is sufficiently small, is given to the optimization software along with a feasible point where the constraints are not active. Because all nodes on the boundary of the mesh are fixed, there are $2V_I$ and $3V_I$ degrees of freedom in the optimization problem for two and three-dimensional meshes respectively, where V_I is the number of internal vertices.

The objective function is continuous and bounded below on the nonempty, closed feasible region. Therefore, if the feasible region is also bounded, we can assert the existence of an optimal solution to the problem. The feasible region will be bounded, for example, if all of the mesh nodes are in a bounded set.

3. OPTIMIZATION METHODS

The optimization problem we want to solve is both nonconvex and nonlinearly constrained, properties that can pose difficulty for optimization methods. Therefore, we evaluate a variety of algorithms to determine their robustness and speed. The differences among the optimization methods are in how they handle constraints, calculate improving directions, and accept new iterates. The algorithms considered in this paper can be roughly classified into sequential quadratic programming, interior-point, and augmented Lagrangian methods. We refer the reader to [24] for more detailed information on numerical optimization methods.

Sequential quadratic programming methods iteratively solve optimization problems containing a quadratic approximation to the objective function and a linear approximation of the constraints to determine a direction. Many variations on this theme exist and have been implemented. We consider two packages in this category: FilterSQP and SNOPT. FilterSQP [20] uses an exact Hessian and incorporates a trust region to restrict the length of the calculated direction. If the full direction is not acceptable, a new direction is calculated by tightening the length restriction. The acceptance criterion uses the notion of a filter and allows nonmonotonic behavior in the objective function and norm of the constraint violation. SNOPT [14] uses an approximation to the Hessian and a linesearch along the calculated direction to find an improving iterate. Both codes use an active set method to solve the quadratic subproblems generated.

Interior-point methods reformulate the original inequality-constrained optimization problem into one containing only equality constraints by adding slack variables and then removing the bounds on the slack variables by incorporating them into the objective with a log-barrier penalty function. For the problem considered, the resulting reformulation is

$$\begin{array}{ll} \min & \sum_{\epsilon_k} \mu^{-1}(\epsilon_k) - \nu \sum_{\epsilon_k} \ln s_{\epsilon_k} \\ \text{subject to} & \tau(\epsilon_k) = s_{\epsilon_k} + \beta \text{ for all } \epsilon_k \end{array}$$

where s_{ϵ_k} are the slack variables and ν is the penalty parameter. The reformulation is then solved for ν converging to zero. LOQO and KNITRO are the two codes that we consider in this category. LOQO [29] solves the equality-constrained problem using Newton's method to calculate a direction and then finds a new iterate using a linesearch along the direction. KNITRO [5] uses a sequential quadratic programming method with a trust region to calculate the solution to the equality-constrained problem for a fixed ν.

Augmented Lagrangian methods reformulate the inequality-constrained problem into a problem with only simple bounds by adding slack variables and incorporating the resulting equality constraints into the objective function. LANCELOT [8] is in this category. It solves the resulting bound constrained problem using a trust-region algorithm. MINOS [23] is similar in that it uses an augmented Lagrangian approach. MINOS, however, also includes a linearization of the nonlinear constraints in the subproblems and solves the linearly constrained problem with an active set method. See [24] for a complete description of augmented Lagrangian methods.

We also consider two methods specifically written to solve the inequality-constrained optimization problem using known information. In particular, we are guaranteed that we will start from a feasible point and that none of the nonlinear constraints will be satisfied as equalities at a solution[2]. The latter condition means that the constraints are redundant and can be removed, provided we safeguard the algorithm to prevent element inversions.

[2]Untangling methods can be used to create an initially valid mesh [13] and β can always be chosen sufficiently small to guarantee the second condition.

The first method, NLCG [19], uses a Polak-Ribière nonlinear conjugate gradient method [24] with an inexact linesearch. The second, FeasNewt, is a feasible Newton method that solves a quadratic approximation of the objective function to find a direction and performs a linesearch along this direction to find an improved point. The step size is reduced whenever an inverted element is found. The direction is calculated using conjugate gradients. The use of conjugate gradients is important because we want to calculate either a minimizer of the quadratic approximation or a direction of negative curvature. In most cases, the conjugate gradient method applied to the quadratic approximation of the objective function provides such directions. For the test cases reported, an appropriate direction was always found with the conjugate gradient method. This approach is similar to the Dembo and Steihaug method [9] for unconstrained optimization.

4. NUMERICAL RESULTS

Because we want to solve mesh quality optimization problems in multiple dimensions and for a number of different geometries, element types, and mesh sizes, our two primary concerns when selecting an algorithm are robustness and speed. We also desire a method that converges monotonically and maintains mesh validity throughout the optimization process. Such a method can be terminated early with a guaranteed improvement in the shape-quality metric. The method chosen should also effectively use the initial point provided as input to the optimization routine as it is typically near a solution. This property would also ensure effective restarts are possible from a partially converged solution. Finally, because today's meshes contain a very large number of elements, the chosen method should be scalable with respect to the number of elements and parallelizable.

For this study, the optimization problem was implemented in the AMPL modeling language [10]. In general, modeling languages provide an easy way to algebraically represent optimization problems, can deal with large quantities of data, and automatically calculate the derivative and Hessian information needed by the solvers. We chose AMPL because it is commonly used and seven of the eight optimization packages considered in this paper accept problems written with it. We note that some efficiency is lost in function, gradient, and Hessian evaluations when using a modeling language. For our AMPL implementation of the optimization problem, hand-coded versions of the function, gradient, and Hessian evaluation routines were more than 10 times faster than those generated by AMPL. However, writing the required derivative and Hessian routines is time consuming and prone to error. Using a consistent approach in their computation allows us to qualitatively compare the different methods.

To test the performance of the optimization methods with respect to our desired characteristics, we developed a series of test cases in both two and three dimensions. We examined algorithm effectiveness as the initial mesh quality degrades, robustness on both two- and three-dimensional meshes, convergence properties, and scalability. The optimization packages use different termination criteria related to feasibility and optimality. In an attempt to have more uniform results, we used a tolerance of 10^{-6} for the measures used by the individual optimization methods. All tests were run on Solaris UltraSPARC workstations. The solvers available in AMPL were run on a 296 MHz workstation while NLCG was run on a 400 MHz workstation.

4.1 Effect of Initial Mesh Quality

We first evaluate the effectiveness and performance of the various optimization methods as the quality of the initial mesh degrades. For this test we use a simple honeycomb mesh containing 2040 equilateral triangles (see the leftmost mesh in Figure 1) and create a series of increasingly poor quality meshes by perturbing the vertices by a percentage of the initial mesh edge length in a random direction. The perturbed mesh is checked to ensure that there are no inverted elements. For this series, the perturbation percentages are 0, 50, 70, 90, and 99. The meshes corresponding to 50 and 99 percent perturbation are shown in the two rightmost meshes in Figure 1, respectively. Table 1 reports the maximum and average value of the mean ratio metric for the five initial meshes.

Table 1. Maximum and average mean ratio metric for the honeycomb series at the initial mesh

P	MR_{max}	MR_{avg}
0	1.00	1.00
50	1.44	1.04
70	2.10	1.08
90	4.58	1.16
99	52.2	2.60

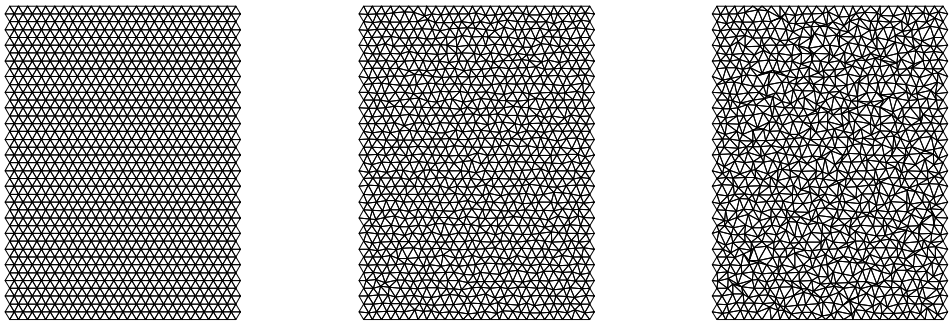

Figure 1. The honeycomb mesh series

Seven optimization solvers were run on this series: the six general-purpose codes and the FeasNewt method. We are not able to report results for NLCG on the two-dimensional test cases as it is currently available only for three-dimensional CUBIT [4] meshes. Figure 2 shows the time to solution as a function of the perturbation percentage. Because the cost of the methods varies dramatically, we plot the results using a logarithmic scale in time. No point is plotted in the graph if the algorithm was unable to calculate an optimal solution.

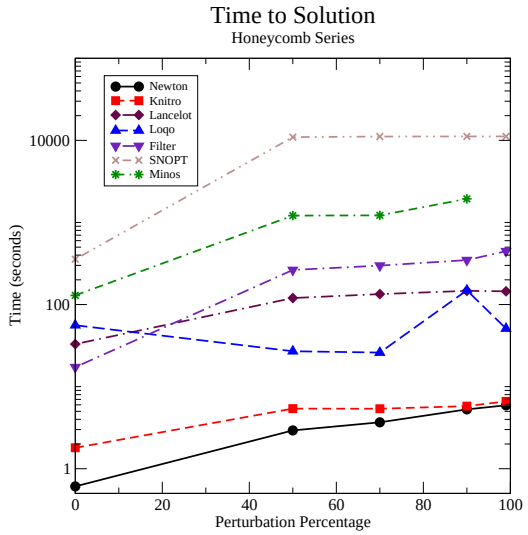

Figure 2. The time to solution for the 7 methods as a function of the perturbation percentage

The FeasNewt and KNITRO solvers reach the optimal mesh in the shortest amount of time for this series of problems. In both cases, the solvers effectively use a near-optimal initial point; the zero perturbation case is not optimal because the initial mesh points were truncated to four decimal places in the AMPL data file. The true optimum is obtained in one and two iterations, and 0.61 and 1.80 seconds, respectively. As the initial quality of the mesh worsens, these two solvers require a slowly, monotonically increasing amount of time to find the optimal solution. The FeasNewt solver is slightly faster than KNITRO, requiring a maximum of 5.93 seconds to solve the hardest of the five test cases compared with 6.62 seconds needed by KNITRO.

In the second performance tier, we place the LANCELOT, LOQO, and FilterSQP solvers. Each of these methods successfully solved all of the test cases but required significantly more time than the FeasNewt and KNITRO solvers. In particular, these methods required 4, 50, and 12 iterations, and 30.91, 54.44, and 17.28 seconds to solve the unperturbed problem, respectively. As the perturbation increased, the solvers required tens to hundreds of seconds to calculate a solution. One of the methods, LOQO, does not increase monotonically in cost. The initial decrease in time is likely due to the nature of interior-point methods, which tend to perform better when the initial point is not too close to the solution. As the initial quality worsens, we see a general increase the number of iterations. In the 90 percent perturbation case, LOQO appears to invert some elements of the mesh and must then return to the feasible region, accounting for the spike in the time.

SNOPT and MINOS perform the worst on this test series. Both are prohibitively expensive; MINOS required as much as 20 minutes to solve the problem with a 90 percent perturbation and was unable to solve the problem with a 99 percent perturbation. SNOPT required 3 hours to solve the 99 percent perturbation case. The longer running times were expected because these methods use only first-order information and approximate the needed Hessian matrices.

4.2 Two-Dimensional Test Cases

To further evaluate the performance of these seven solvers, we analyze their ability to solve two two-dimensional test cases. Both test cases are generated using the Triangle mesh generation package [28]. The first test case, ANL, has a concave geometry. The mesh generated by Triangle was modified by moving the interior vertices to deliberately create poor-quality elements along the boundary. The second test case, Rand, is generated by randomly choosing points in the unit square and triangulating them using a Delaunay criterion. In Table 2, we give the total number of vertices and elements, V_T and E, the number of interior vertices, V_I, and the maximum mean ratio and average mean ratio, MR_{max} and MR_{avg} for each test case. The meshes are shown in Figure 3.

Table 2. The size and initial quality of the two-dimensional test cases

Mesh	V_T	E	V_I	MR_{max}	MR_{avg}
ANL	312	456	184	9.82	1.73
Rand	1152	2170	937	32.6	1.84

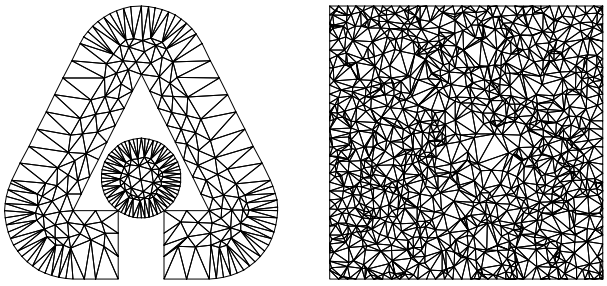

Figure 3. The two-dimensional test cases: ANL (left), Rand (right)

The results of running each of the seven optimization solvers on the two-dimensional meshes are given in Table 3. For each solver, we report the termination status, S, of the solver as an "S" or an "F" for succeed or fail, respectively. If the method fails, we give the termination message reported by the solver. If the method succeeds, we report the resulting mesh quality in terms of the maximum and average mean ratio values, MR_{max} and MR_{avg}, respectively; the cost in seconds, T; and the number iterations, I.

Table 3. The mesh quality, time, and iteration count for the optimization solvers applied to the two-dimensional test cases

Method	S	MR_{max}	MR_{avg}	T	I
ANL					
FeasNewt	S	1.62	1.11	1.95	15
KNITRO	F	Trust region radius too small			
LANCELOT	F	No decrease in constraint violation			
LOQO	S	1.62	1.11	1.59	18
FilterSQP	S	1.62	1.11	9.10	19
SNOPT	S	1.62	1.11	257	231
MINOS	F	Problem unbnded or badly scaled			
Rand					
FeasNewt	S	2.20	1.12	20.0	25
KNITRO	F	Trust region radius too small			
LANCELOT	F	No decrease in constraint violation			
LOQO	S	2.20	1.12	28.2	33
FilterSQP	S	2.20	1.12	4918	264
SNOPT	F	Killed after 26 hours			
MINOS	F	Cannot calculate improving point			

The quality of the mesh at the optimal solution for both test problems is considerably improved; for ANL, MR_{max} has been reduced from 9.82 to 1.62, and MR_{avg} has been reduced from 1.73 to 1.11. For Rand, MR_{max} has been reduced from 32.6 to 2.20, and MR_{avg} has been reduced from 1.84 to 1.12. On these two problems, the FeasNewt and LOQO solvers are the best performers; they solve both test problems successfully and are comparable in terms of time. FilterSQP, although it solves both test cases, is orders of magnitude more expensive than either the FeasNewt or LOQO solvers. KNITRO, LANCELOT, and MINOS are unable to solve either of the two test problems and produce tangled meshes. Once an element became inverted, these algorithms proceeded to decrease the objective function to negative infinity, but never returned to the feasible region. SNOPT solved only the relatively easy ANL test case but required 257 seconds to do so whereas the other successful methods required at most 10 seconds. Given the lack of robustness and expense of KNITRO, LANCELOT, SNOPT, and MINOS, we do not consider them further.

4.3 Three-Dimensional Test Cases

We now use the remaining solvers, FeasNewt, LOQO, and FilterSQP, to improve the quality of four tetrahedral meshes generated using the CUBIT mesh generation package [4]. We also report the results for the NLCG solver described in Section 3. For each test case, we give in Table 4 the total number of vertices and elements, V_T and E; the number of interior vertices, V_I; and the maximum mean ratio and average mean ratio, MR_{max} and MR_{avg}. The meshes are shown in Figure 4.

Table 4. The size and initial quality of the three-dimensional test cases

Mesh	V_T	E	V_I	MR_{max}	MR_{avg}
Duct	1106	4267	382	3.00	1.26
Gear	866	3116	260	2.84	1.37
Foam	1337	4847	289	4.06	1.34
Hook	1190	4675	400	3.36	1.32

As with the two-dimensional cases, we report in Table 5 the termination status, final mesh quality, time to solution, and iteration count for the three-dimensional cases.

In all cases, the average mean ratio is improved at the optimal solution, reflecting the goal of the objective function formulation. We note that the improvement is not as dramatic as it was in the two-dimensional cases, because the initial mesh quality is good. In three of the four test cases, the maximum mean ratio is also improved, even though it is not explicitly the goal of the optimization procedure; in the Gear test problem it is slightly worsened. All four methods are able to solve all of the test cases; these are, listed in order from fastest to slowest: FeasNewt, LOQO, NLCG, and FilterSQP. The FeasNewt solver is a factor of 2.2 to 4.5 times faster than its nearest competitor in all test cases. While FilterSQP successfully solved all test cases, it is a factor 5.6 to 19 times slower than FeasNewt.

4.4 Convergence Histories

In many cases, the exact optimal solution is not required from a mesh improvement technique. Rather, a very good solution is desired quickly.

Table 5. The results of the optimization solvers on the three-dimensional test cases

Method	S	MR_{max}	MR_{avg}	T	I
Duct					
FeasNewt	S	2.92	1.24	14.9	6
FilterSQP	S	2.92	1.24	282	82
LOQO	S	2.92	1.24	75.2	24
NLCG	S	2.92	1.24	67.5	50
Gear					
FeasNewt	S	3.28	1.33	9.8	5
FilterSQP	S	3.28	1.33	54.7	21
LOQO	S	3.28	1.33	21.8	11
NLCG	S	3.28	1.33	40.1	41
Foam					
FeasNewt	S	3.52	1.33	10.9	5
FilterSQP	S	3.52	1.33	86.2	30
LOQO	S	3.52	1.33	35.3	17
NLCG	S	3.52	1.33	76.3	58
Hook					
FeasNewt	S	2.91	1.30	15.9	5
FilterSQP	S	2.91	1.30	276	67
LOQO	S	2.91	1.30	52.6	16
NLCG	S	2.91	1.30	83.0	62

Thus, we now evaluate the convergence history of each method to determine the feasibility of early termination to reduce computational costs. The two characteristics that would make this possible are monotonic convergence to the solution and significant early progress toward the optimal point. We plot the value of the objective function as a function of time for each solver on the three-dimensional test cases and show the results in Figure 5. Note that the horizontal axis is scaled to highlight the early convergence behavior of the methods, and the complete time history of some of the methods is not shown.

The FeasNewt, FilterSQP, and NLCG solvers all converge to the optimal solution monotonically, making them candidates for early termination. LOQO does not always converge monotonically as is illustrated by the Duct, Hook, and Foam test cases. This nonmonotonic behavior is caused by the penalty function, which allows LOQO to visit points far from the solution before converging to the optimal point. Thus, even though its total

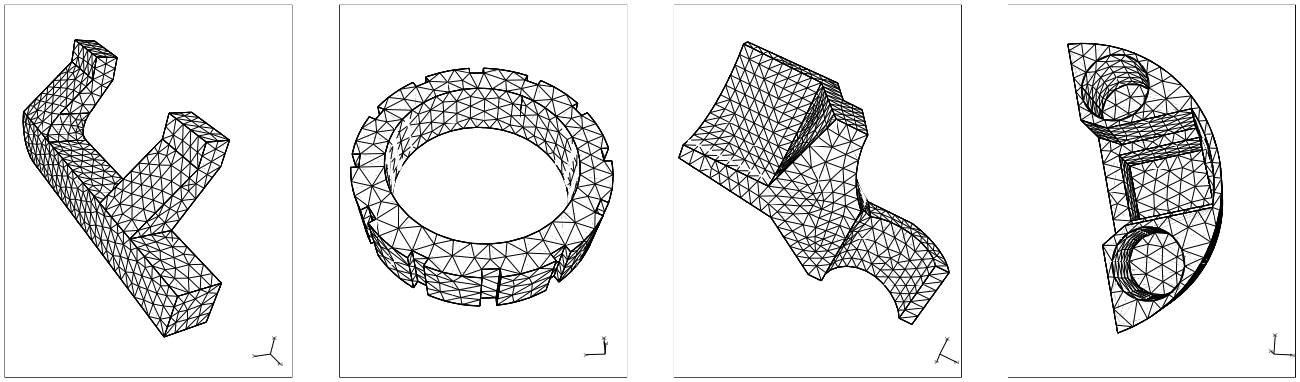

Figure 4. The four tetrahedral mesh test cases for duct, gear, hook, and foam geometries

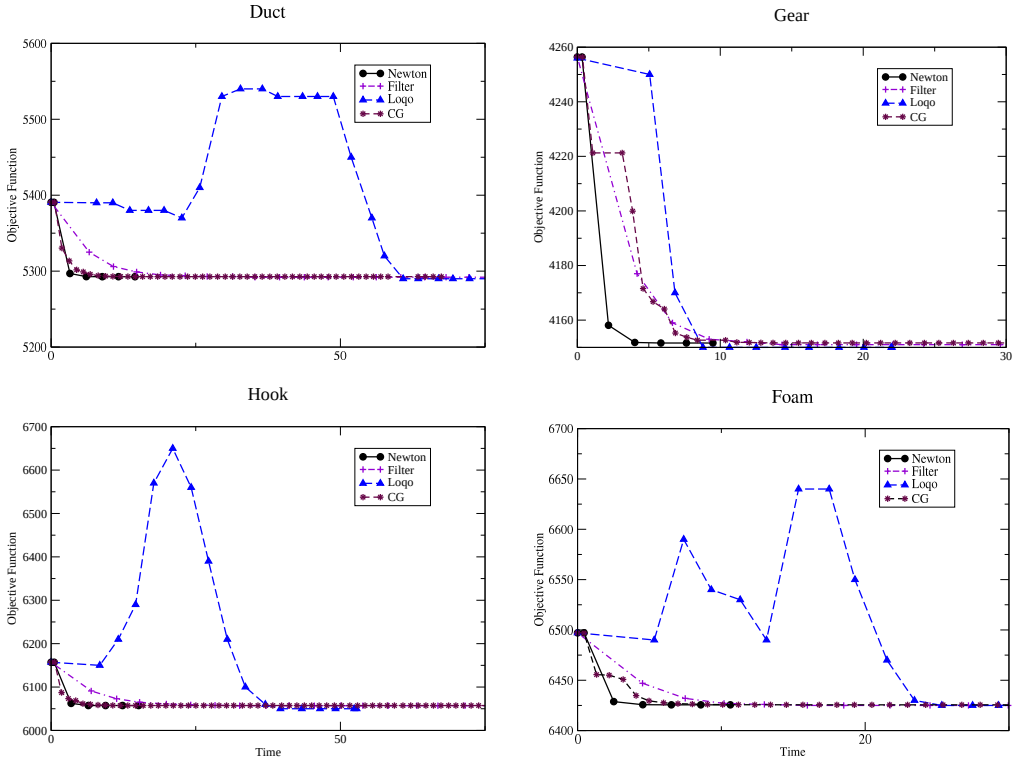

Figure 5. The convergence histories for FeasNewt, FilterSQP, and LOQO on the four tetrahedral mesh test cases

time to solution as reported in Table 5 is less than FilterSQP in all cases and is less than NLCG in three cases, it typically takes much longer to approach the optimal solution. Thus, it is not a good candidate for early termination.

Of the three methods that converge monotonically, FeasNewt approaches the optimal solution the quickest in all four cases. In two of the four cases, the NLCG method is very competitive with FeasNewt in terms of how it approaches the optimal solution, even though it takes much longer to converge to the exact optimal point. In the other two cases, it takes about twice as long to approach the optimal point. FilterSQP takes two to three times longer than FeasNewt to approach the optimal point in all cases. Furthermore, FilterSQP is not guaranteed to monotonically decrease the objective function, even though this behavior was observed on the test problems.

4.5 Scalability

To obtain a sense of how these methods will scale as the problem size increases, we examine their

performance on the Duct geometry as the number of elements increases. The mesh sizes and quality information are given in Table 6.

Table 6. The size and initial quality of a series of meshes created on the duct geometry to test scalability

Mesh	V_T	E	V_I	MR_{max}	MR_{avg}
Duct1	1067	4104	382	2.34	1.21
Duct2	2139	9000	965	3.55	1.21
Duct3	4199	19222	2302	3.29	1.21
Duct4	7297	35045	4480	2.71	1.20
Duct5	13193	65574	8738	4.30	1.19

The total time to solution and iteration counts for the FeasNewt, FilterSQP, LOQO, and NLCG methods are given in Table 7. A dash entry indicates that the method was unable to solve the problem in a reasonable amount of time. As the problem size increases, the FeasNewt method is consistently the fastest method and maintains a nearly constant number of iterations. Each iteration becomes more expensive as the problem size increases, but of all the methods considered, this method's total time to solution grows the most slowly. Although LOQO and NLCG require about the same amount of time to solve the Duct1 problem, the NLCG method is more scalable in that its time to solution grows more slowly as the the problem size increases. In particular, the Duct5 to Duct1 ratio for NLCG is 43.8 and for LOQO is 91.8. The FilterSQP method does not perform well as the problem size increases, requiring over 5 hours to solve the Duct3 problem. We did not investigate its behavior on the Duct4 and Duct5 problems.

5. CONCLUSIONS

We have conducted a series of numerical experiments to determine which of several selected optimization methods are most suitable for solving the mesh shape quality optimization problem where all of the vertices are simultaneously repositioned to improve average quality. We compared eight different solvers: six state-of-the-art solvers and two custom solvers we developed. In Table 8 we summarize our findings in terms of the methods' robustness, time to solution, flexibility to be used with an early stopping criterion, and scalability. In each category, we score the methods as excellent (X), good (G), average (A), or poor (P). A dashed line indicates that the method was not analyzed for a given characteristic.

The two methods that performed the best were, not surprisingly, those written specifically for the mesh quality improvement problem. In particular, the custom-developed FeasNewt method is the best performer in all categories. It solves every test problem and was consistently the fastest technique, particularly in three dimensions. It monotonically converges to a solution and can therefore be used with a flexible stopping criterion. Furthermore, it effectively uses a good initial starting point and the computational cost grows as the quality of the initial mesh degrades. A complete description of this solver is planned for another paper. Following FeasNewt, the NLCG method is also considered to be a top performer. Although we could not test it on the two-dimensional test cases, it solved every three-dimensional problem, and the method is well suited for quickly finding good solutions. Further, only FeasNewt and NLCG are guaranteed to remain feasible with respect to the nonlinear constraints in the optimization problem.

The LOQO and FilterSQP methods, although robust, are not as good as either of the other two methods. LOQO's convergence properties, both as the initial mesh becomes more difficult to solve and within a given run, make it an unpredictable solver that cannot be terminated early. FilterSQP was observed to converge monotonically but is prohibitively expensive unless early termination is considered. Both LOQO and FilterSQP would be more competitive with NLCG if we were to remove our reliance upon AMPL and write custom interfaces. The other four solvers, KNITRO, LANCELOT, SNOPT, and MINOS, were unable to solve one or more of the two-dimensional problems; they were not considered in the three-dimensional test cases.

Now that promising methods have been identified for solving the mesh shape quality optimization problem, a number of interesting extensions to this comparison can be considered. First, we note that our tests of the solvers used the mean ratio metric. Although this represents a typical shape metric, in the future it will be necessary to perform similar tests on other shape metrics, anisotropic meshes, other types of mesh quality optimization metrics, and different objective functions. This comparison of optimization software packages is not exhaustive. However, since it appears better to use

Table 7. The time to solution in seconds, T, and cost in iterations, I, for the four solvers as mesh size in the Duct geometry increases

	FeasNewt		FilterSQP		LOQO		NLCG	
Mesh	T	I	T	I	T	I	T	I
Duct1	12.2	4	360	96	68.7	20	66.0	48
Duct2	37.1	5	2842	257	181	20	169	51
Duct3	90.6	5	18750	630	807	31	518	64
Duct4	151	4	–	–	1777	24	984	67
Duct5	401	5	–	–	6309	24	2889	101

solvers tailored to the shape optimization problem as opposed to general purpose solvers, inclusion of other examples of the latter in future studies may not be worthwhile.

REFERENCES

[1] E. Amenzua, M. V. Hormaza, A. Hernández, and M. B. G. Ajuria. A method for the improvement of 3d solid finite-element meshes. *Advances in Engineering Software*, 22(4):45–53, 1995.

[2] R. E. Bank and R. K. Smith. Mesh smoothing using a posteriori error estimates. *SIAM Journal on Numerical Analysis*, 34(3):979–997, June 1997.

[3] P. Barrera and L. Castellanos. Curvilinear coordinate system generation over plane irregular regions. *Faculty Report, U.N.A.M., Mexico City, Mexico*, 1992.

[4] T. D. Blacker and et al. CUBIT mesh generation environment. Technical Report SAND94-1100, Sandia National Laboratories, Albuquerque, NM, 1994.

[5] R. H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large scale nonlinear programming. *SIAM J. Optimization*, 9(4):877–900, 1999.

[6] J. E. Castillo. A discrete variational grid generation method. *SIAM J. Sci. and Stat. Comp.*, 12(2):454–468, 1991.

[7] C. L. Chen, K. Y. Szema, and S. R. Chakravarthy. Optimization of unstructured grids. *AIAA 95-0217, 33rd Aerospace Sciences Meeting and Exhibit*, 1995.

[8] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A), Springer Series in Computational Mathematics 17.* Springer-Verlag, 1992.

[9] R. S. Dembo and T. Steihaug. Truncated Newton algorithms for large-scale unconstrained optimization. *Math. Prog.*, pages 190–212, 1983.

[10] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming.* Boyd & Fraser Publishing Company, Danvers, MA, 1993.

[11] L. Freitag and P. Knupp. Tetrahedral mesh improvement via optimization of the element condition number. *Intl. J. Numer. Meth. Engr.*, 53:1377–1391, 2002.

[12] L. Freitag and C. Ollivier-Gooch. A comparison of tetrahedral mesh improvement techniques. In *Proceedings of the Fifth International Meshing Roundtable*, pages 87–100. Sandia National Laboratories, 1996.

[13] L. Freitag and P. Plassmann. Local optimization-based simplicial mesh untangling and improvement. *Intl. J. Num. Methods in Engr.*, 49:109–125, 2000.

[14] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. Technical Report SOL 97-3, Stanford University, Palo Alto, CA, 1997.

[15] O.-P. Jacquotte. Grid optimization methods for quality improvement and adaptation. *Chapter 33 in the Handbook of Grid Generation, J.F. Thompson et. al. editors*, 1999.

Table 8. Summary of the capabilities of the methods

Method	Robust	Fast	Flexible	Scalable
FeasNewt	X	X	X	X
NLCG	X	G	X	G
LOQO	X	G	P	A
FilterSQP	X	A	G	P
KNITRO	A	G	–	–
LANCELOT	A	A	–	–
SNOPT	G	P	–	–
MINOS	A	P	–	–

[16] P. Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities, Part I - A framework for surface mesh optimization. *Intl. J. Num. Methods in Engr.*, 48(3):401–420, 2000.

[17] 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. *Intl. J. Num. Methods in Engr.*, 48:1165–1185, 2000.

[18] P. Knupp. Algebraic mesh quality metrics. *SIAM J. Sci. Comput.*, 23(1):193–218, 2001.

[19] P. Knupp. A method for hexahedral mesh shape optimization. *Intl. J. Num. Methods Engr.*, to appear, 2002.

[20] S. Leyffer and R. Fletcher. Nonlinear programming without a penalty function. *Math. Prog.*, to appear, 2002.

[21] A. Liu and B. Joe. On the shape of tetrahedra from bisection. *Math. Comp.*, 63:141–154, 1994.

[22] A. Liu and B. Joe. Relationship between tetrahedron quality measures. *BIT*, 34:268–287, 1994.

[23] B. A. Murtagh and M. A. Saunders. MINOS 5.4 user's guide. Technical Report SOL83-20R, Stanford University, Palo Alto, CA, 1983, (Revised 1995).

[24] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 1999.

[25] V. N. Parthasarathy and S. Kodiyalam. A constrained optimization approach to finite element mesh smoothing. *Finite Elements in Analysis and Design*, 9:309–320, 1991.

[26] D. F. Shanno. Conjugate gradient methods with inexact line searches. *Mathematics of Operations Research*, 3:244–256, 1978.

[27] M. Shephard and M. Georges. Automatic three-dimensional mesh generation by the finite octree technique. *Intl. J. Num. Methods in Engr.*, 32:709–749, 1991.

[28] J. Shewchuk. Triangle: Engineering a 2d quality mesh generator and Delaunay triangulator. In *Proceedings of the First Workshop on Applied Computational Geometry*, pages 124–133, Philadelphia, 1996. ACM.

[29] R. J. Vanderbei and D. F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. Technical Report SOR-97-21, Princeton University, Princeton, NJ, 1997.

[30] D. White and G. Rodrigue. Improved vector FEM solutions of Maxwell's equations using grid pre-conditioning. *Int'l. J. Numer. Meth. Engr.*, 40:3815–3837, 1997.

[31] P. Zavittieri. Optimization strategies in unstructured mesh generation. *Intl. J. Num. Meth. Engr.*, 39:2055–2071, 1996.