A COMPARISON OF INEXACT NEWTON AND COORDINATE DESCENT MESH OPTIMIZATION TECHNIQUES

Lori Freitag Diachin[1] Patrick Knupp[2] Todd Munson[3] Suzanne Shontz[4]

[1]*Lawrence Livermore National Laboratory, Livermore, CA U.S.A. diachin2@llnl.gov*
[2]*Sandia National Laboratories, Albuquerque, NM U.S.A pknupp@sandia.gov*
[3]*Argonne National Laboratory, Argonne, IL U.S.A. tmunson@mcs.anl.gov*
[4]*Cornell University, Ithaca, NY U.S.A. shontz@cam.cornell.edu*

ABSTRACT

We compare inexact Newton and coordinate descent methods for optimizing the quality of a mesh by repositioning the vertices, where quality is measured by the harmonic mean of the mean-ratio metric. The effects of problem size, element size heterogeneity, and various vertex displacement schemes on the performance of these algorithms are assessed for a series of tetrahedral meshes.

Keywords: mesh quality, mesh improvement, mesh smoothing

1. INTRODUCTION

Mesh vertex repositioning algorithms have been used for many years to improve solution accuracy and efficiency; see, for example, [1, 2, 3, 4]. Repositioning techniques vary widely in the time required to implement and modify the algorithm and in the computational cost and effectiveness when applying the algorithm, usually with a trade-off between these criteria. Laplacian smoothing, for example, is easy to implement and inexpensive to apply but can produce tangled meshes. Moreover, this method is limited to the creation of smooth meshes, while vertex repositioning can address other meshing needs such as equidistribution of volumes [5], shape improvement [6], or adaptive R-refinement [7]. These more complex tasks can usually be posed as numerical optimization problems in which an objective function is defined that measures one or more mesh properties. This objective function can then be optimized by repositioning the vertices, leading to improvement in the mesh properties measured.

When one approaches the vertex repositioning problem from an optimization perspective, it is natural to formulate a single objective function to measure global mesh quality. This objective function typically accumulates contributions from local measures of mesh quality into a scalar function of the positions of all free vertices in the mesh.[1] We consider two approaches to numerically optimizing the global objective function. In the *all-vertex* approach, all of the free vertices are moved simultaneously within a single iteration. We employ an *inexact Newton* method as our all-vertex optimization algorithm. In the *single-vertex* approach, only one vertex at a time is modified. This approach is an instance of a *coordinate descent* method, and we apply an exact Newton algorithm to solve the coordinate descent subproblems. The main goal in this paper is to determine when one of these methods is preferable to the other, where preference can include the ease with which the method can be implemented and modified, the computational and memory requirements for applying the method, and the accuracy and

[1]An important alternative to mesh optimization, often used by the unstructured mesh quality community, employs a series of local objective functions.

quality of the mesh produced, perhaps as a function of CPU time. A complete answer should consider all these characteristics.

The preferred method may differ depending on the circumstances. For example, the coordinate descent method may be better suited to quickly finding an approximate solution, while the inexact Newton method may be more suitable for calculating a highly accurate solution. Factors that may be significant in determining the preferred approach include the objective function, quality metric, the desired accuracy in the resulting mesh, the mesh type (structured vs. unstructured), dimension (planar vs. volume), element type (simplicial vs. nonsimplicial), problem size, degree of mesh heterogeneity and anisotropy, and the degree and manner in which the initial mesh differs from the optimal mesh. Algorithm implementations also have a significant impact, since a simple implementation can be much slower than a more sophisticated version.

In this paper we report the results of an initial exploration of these factors to determine the circumstances in which the inexact Newton method or the coordinate descent method may be preferred. To make the study manageable, we limit the number of free parameters and consider a fixed mesh type, quality metric, and objective function template. In particular, we use tetrahedral meshes, the mean-ratio quality metric for isotropic elements, and a template that targets average quality improvement as opposed to worst-case quality improvement. The free parameters in this study are the problem size, element homogeneity, initial mesh configuration, and desired degree of accuracy in the resulting mesh.

2. PROBLEM STATEMENT

2.1 Element and Mesh Quality

An unstructured mesh $\mathcal{M} = (\mathcal{V}^{\mathcal{M}}, \mathcal{E}^{\mathcal{M}})$ consists of a finite set of vertices $\mathcal{V}^{\mathcal{M}}$ and elements $\mathcal{E}^{\mathcal{M}}$, where $V^{\mathcal{M}}$ denotes the number of vertices and $E^{\mathcal{M}}$ the number of elements. The set of boundary vertices for the mesh is denoted by $\mathcal{V}_{\mathcal{B}}^{\mathcal{M}}$, while the set of interior vertices of the mesh, that is, those not on the boundary, is denoted by $\mathcal{V}_{\mathcal{I}}^{\mathcal{M}}$. Let $x_v^{\mathcal{M}}$ be the coordinates for vertex $v \in \mathcal{V}^{\mathcal{M}}$, where $x_v^{\mathcal{M}} \in \Re^d$. For surface and volume meshes $d = 3$, while for planar meshes $d = 2$. Moreover, $x^{\mathcal{M}} \in \Re^{d \times V^{\mathcal{M}}}$ refers to all of the vertex coordinates in the mesh. Each element $e \in \mathcal{E}^{\mathcal{M}}$ consists of a small subset of the vertices and the edges between these vertices, where $|e|$ is the number of vertices referenced by element e, $\mathcal{V}^e$ refers to the vertices referenced by e, and $x_{\mathcal{V}^e}^{\mathcal{M}} \in \Re^{d \times |e|}$ denotes the matrix of coordinates for the vertices. Where the context is clear, we suppress the superscripts that refer to the mesh.

The mesh $\mathcal{M}$ can be decomposed into one or more submeshes. In particular, let $\mathcal{E}^{\mathcal{N}} \subseteq \mathcal{E}^{\mathcal{M}}$, and let $\mathcal{V}^{\mathcal{N}}$ be the set of vertices referenced by the elements of $\mathcal{E}^{\mathcal{N}}$. Then, $\mathcal{N} = (\mathcal{V}^{\mathcal{N}}, \mathcal{E}^{\mathcal{N}})$ is called a *submesh* of $\mathcal{M}$. Each submesh of $\mathcal{M}$ has its own set of boundary and interior vertices. The most important submeshes in this paper are the *local patches* $\mathcal{P}_v$. For each $v \in \mathcal{V}_{\mathcal{I}}$, the set $\mathcal{E}^{\mathcal{P}_v}$ consists of the elements in $\mathcal{E}$ containing vertex v, while $\mathcal{V}^{\mathcal{P}_v}$ consists of the vertices in $\mathcal{V}$ referenced by the elements in $\mathcal{E}^{\mathcal{P}_v}$. For each $\mathcal{P}_v$, v is the only interior vertex in the local patch.

Associated with the mesh is a continuous function $q : \Re^{d \times |e|} \rightarrow \Re$ that measures one or more of the geometric properties of an element as a function of the vertex positions.[2] In particular, $q(x_e)$ measures the *quality* of element e, where we assume larger values of $q(x_e)$ indicate higher quality. A specific function q is referred to as an element *quality metric*. Many functions can serve as quality metrics, so the quality of an element is not uniquely defined. For example, there are different metrics to measure the shape, size, and orientation of an element. In general, *useful* quality metrics possess other properties in addition to continuity, but a discussion of this topic is beyond the scope of the present study [8]. The overall quality of mesh $\mathcal{M}$ is measured by a function $\mathcal{Q}^{\mathcal{M}} : \Re^E \rightarrow \Re$ that takes as input the vector of element quality metrics, $\prod_{e \in \mathcal{E}} q(x_e)$, where $\prod$ denotes the Cartesian product. The mesh quality depends on both the choice of the specific element quality metric q and the particular template function $\mathcal{Q}$ used to combine them. Useful template functions can be constructed from the arithmetic or other means.

2.2 The Mean-Ratio Metric

An important variable in this study is the choice of quality metric. In general, we expect that study results could vary significantly depending on whether or not one were to chose shape metrics as opposed to size, smoothness, combined, or other metrics. For this initial study on solver comparisons, we focus on the mean-ratio element shape quality metric. It seems likely that other shape metrics such as aspect ratio or condition number would give similar timing results; we plan to verify this in future work, as well as consider metrics which are not explicitly focused on shape.

Let $S_{d \times d}$ be a matrix with $\det(S) > 0$. Then the mean ratio μ of S is the scalar

$$\mu(S) = \frac{\det(S)^{2/d}}{\|S\|_F^2},$$

[2]For hybrid meshes, the exact definition of q can change depending on the element type. However, we assume that the quality metric, shape, for example, is the same for every element.

where $\|S\|_F = \sqrt{tr(S^t S)}$ is the Frobenius norm. One can readily show that $0 < \mu(S) \leq \frac{1}{d}$.

To apply the mean ratio to element quality, assume that each vertex of the element is connected to d edges (and therefore d other vertices) belonging to the element.[3] Let x_i be the coordinates of the ith vertex, and let x_k be the coordinates of one of the other vertices in the element to which v_i is connected by an edge. Construct the matrix $A^{(i)}_{d\times d}$ whose columns are the vectors $x_k - x_i$. The columns are ordered in such a way as to preserve element orientation so that if $\det(A^{(i)}) \leq 0$ for any vertex in the element, the element has locally nonpositive volume; we call such elements "inverted." Let $W_{d\times d}$ be a reference matrix that determines the ideal element shape (e.g., an equilateral reference triangle is often used for triangular elements). The reference matrix is found by constructing W from the ideal element in the same way A is constructed from the mesh element. Finally, let $T^{(i)} = A^{(i)}W^{-1}$. For each element vertex i let $\mu_i = \mu(T^{(i)})$ be the mean ratio at the ith element vertex. Let these mean ratios at the element vertices be averaged in some way to form a mean ratio $\mu_j = \mu_{e_j}$ for the jth element in the mesh that is symmetric in the element vertex indices.[4] The shape quality of the jth element is then

$$q_j = \mu_j d.$$

As one would expect, the element shape quality metric is both scale and rotation invariant. Furthermore, $0 < q_j \leq 1$ with $q_j = 1$ only when the element attains the ideal reference shape. We do not define the mean ratio for matrices with nonpositive determinants. Therefore, the shape quality of "inverted" elements is not defined. For further details on shape metrics see [9].

2.3 Quality Improvement Problem

To improve the overall quality of $\mathcal{M}$ when using $\mathcal{Q}$, we compute an $x^* \in \Re^{d\times V^{\mathcal{M}}}$ such that x^* is an optimal solution to

$$\max_{x_{\mathcal{V}_\mathcal{I}}} \quad \mathcal{Q}\left(\prod_{e\in\mathcal{E}} q(x_e)\right) \tag{1}$$

subject to the constraint that $x_{\mathcal{V}_\mathcal{B}} = \bar{x}_{\mathcal{V}_\mathcal{B}}$, where $\bar{x}_{\mathcal{V}_\mathcal{B}}$ are the coordinates for the boundary vertices. Note that additional constraints can be added if the locations for some of the interior vertices also need to be fixed. If the objective function for this optimization problem is uniformly concave as a function of $x_{\mathcal{V}_\mathcal{I}}$, that is, the Hessian matrix, $\nabla^2_{x_{\mathcal{V}_\mathcal{I}},x_{\mathcal{V}_\mathcal{I}}}\mathcal{Q}(\cdot)$, is uniformly negative definite, then an x^* solving this optimization problem exists and is unique. If the objective function is not concave, then one can only hope to find a local maximizer for the optimization problem and may instead compute a critical point.

We use the harmonic mean template for all our numerical results. This template produces the objective function

$$\frac{E}{\sum_{e\in\mathcal{E}} \frac{1}{\mu_e d}}.$$

This objective function is maximized precisely when the denominator is minimized. Therefore, the optimization problem we solve is

$$\min_{x_{\mathcal{V}_\mathcal{I}}} \quad \mathcal{Q}_{\text{hm}}(x) := \frac{1}{E}\sum_{e\in\mathcal{E}} \frac{1}{\mu_e d} \tag{2}$$

subject to the same boundary constraints as in (1). The objective function in this case is below by 1 and continuous for the set of noninverted meshes. We note that this objective function minimizes the average inverse mean ratio. We assume that we start at a feasible point, that is, a noninverted mesh. We also require the improved mesh to be noninverted, which translates to the implicit constraint $\det(A^{(i)}) > 0$ for every element vertex. There is no need to implement these constraints explicitly, however, because the denominator in the mean ratio acts as an effective barrier to element inversion.

3. IMPROVEMENT ALGORITHMS

Many algorithms can be applied to compute a solution to the quality improvement problem (2).[5] In this paper, we consider the block coordinate descent and inexact Newton methods [10, 11]. The block coordinate descent method optimizes the location of a single vertex at a time by applying an optimization algorithm to a restricted problem in which only the coordinates for the given vertex are allowed to move. This optimization step is repeated for each of the other vertices in the mesh. This iterative repositioning stops when the norm of the gradient of the global objective function is small. The inexact Newton method, on the other hand, constructs a quadratic approximation to the global objective function at the current iterate and computes a solution to this quadratic program by solving a large system of equations. A new iterate is then found for which the objective function has decreased.

The block coordinate descent algorithm solves a sequence of small optimization problems to improve the global objective function but has a slow asymptotic

[3]This approach excludes elements such as pyramids but includes triangles, tetrahedra, wedges, quadrilaterals, and hexahedra.

[4]We show in [6] that averaging is unnecessary in the case of triangular or tetrahedral elements provided the reference element is equilateral.

[5]Recall that (2) minimizes the inverse mean-ratio objective function, so the stated algorithms use minimization terminology. However, the same algorithms can be used for the maximization problem (1).

convergence rate, while the inexact Newton method solves a large quadratic optimization problem at every iteration but has a fast asymptotic convergence rate. If the global objective function in (2) is uniformly convex in the free variables, then both algorithms converge to the same solution x^* [10]. However, if the objective function is not convex, as is often the case in mesh optimization, we can only say that if the block coordinate descent method converges to x^*, then x^* is a critical point for the optimization problem (2) and x^* may not be a local minimizer. Moreover, the optimization subproblems in the nonconvex case may have either no solution or many local minimizers.

However, for the inverse mean-ratio metric, even though the global objective function is not convex everywhere, one can prove that the objective function for each subproblem of the block coordinate descent method is strictly convex [12, 13] and the feasible region is compact. Therefore, each of these subproblems has a unique solution. Note that the inexact Newton and block coordinate decent algorithms may not converge to the same critical point.

3.1 Block Coordinate Descent Method

The block coordinate descent method modifies a single vertex at a time by applying one iteration of an exact Newton method to the subproblem obtained by fixing the rest of the vertices at their current coordinates. That is, we computed a direction d where for vertex $v \in \mathcal{V}_{\mathcal{I}}$, the vth component of d is obtained by solving the system of equations

$$\nabla^2_{v,v} \mathcal{Q}_{\text{hm}}(x^k) d_v = -\nabla_v \mathcal{Q}_{\text{hm}}(x^k).$$

The remaining components of d are set to zero. Note that we need only the Hessian matrix with respect to vertex v, so the complete Hessian matrix for the global objective function does not need to be computed. The direction is obtained by computing the LDL^T factorization of the $d \times d$ local Hessian matrix and applying it to the right-hand side of the problem. An iterative method is not needed in this case because the system of equations is very small. For the metric used, the Hessian matrix is positive definite, so the factorization can always be computed.

Having obtained a search direction, we then use an Armijo linesearch [14] to obtain a new iterate with improved quality. In particular, we compute the smallest nonnegative integer m such that

$$\mathcal{Q}_{\text{hm}}(x^k + \beta^m d) \le \mathcal{Q}_{\text{hm}}(x^k) + \sigma \beta^m \nabla \mathcal{Q}_{\text{hm}}(x^k)^T d.$$

When searching along the direction, all points where the resulting mesh is degenerate or inverted are rejected; the objective function value is treated as positive infinity in these cases. To judge progress, we need consider the quality of the elements only in local patch $\mathcal{P}_v$, since the quality of the elements outside of this patch do not change when the location of vertex v is modified. Hence, the Armijo linesearch is computationally inexpensive. To make both the Armijo linesearch and Hessian matrix computations efficient for the block coordinate descent method, we precompute a mapping from each vertex to the elements referencing the vertex.

We then update $x^k = x^k + \beta^m d$ and choose a different vertex to optimize. The order in which the vertices are traversed in each pass is determined by a reverse breadth-first search starting from the vertex farthest from the origin.

An iteration of the block coordinate descent method consists of a single repositioning of each of the interior vertices. Once each of the interior vertices has been repositioned, we proceed to the next iteration, by setting $x^{k+1} = x^k$, and repeat the local improvement process until the gradient of the global objective function is less than some tolerance.

3.2 Inexact Newton Method

The inexact Newton method computes a direction d by solving the system of equations

$$\nabla^2_{x_{\mathcal{V}_{\mathcal{I}}}, x_{\mathcal{V}_{\mathcal{I}}}} \mathcal{Q}_{\text{hm}}(x^k) d = -\nabla_{x_{\mathcal{V}_{\mathcal{I}}}} \mathcal{Q}_{\text{hm}}(x^k)$$

by applying the conjugate gradient method with a block Jacobi preconditioner [15], where x^k is the current iterate. Having obtained a search direction, we then use an Armijo linesearch [14] to obtain a new iterate with a sufficiently improved quality. This linesearch is the same as the block coordinate descent linesearch but must consider the improvement in the global objective function instead of the improvement in the local patch.

The gradient and Hessian of the objective function are calculated by assembling the gradients and Hessians for each of the element functions into a vector and symmetric sparse matrix. Only the upper triangular part of the Hessian matrix is stored in a block compressed sparse row format. Each block corresponds to a coordinate in the original problem. The gradient and Hessian elements corresponding to fixed vertices on the boundary of the domain are ignored.

The preconditioner consists of the Hessian with respect to the (i, i) coordinates, resulting in a block diagonal preconditioner, where each block consists of a $d \times d$ matrix. An LDL^T factorization of each diagonal matrix is performed when calculating the preconditioner. The preconditioner is applied by setting $y = L^{-T}(D^{-1}(L^{-1}x))$. We store D^{-1} so that the middle product consists of a few multiplications, instead of a few divisions. Each diagonal block of the Hessian matrix is positive definite even though the overall Hessian is indefinite in general [12, 13], so the preconditioner can always be computed.

3.3 Implementation Characteristics

Our implementations of the coordinate descent method and the inexact Newton method have been coded with a bias toward achieving high performance with minimal memory requirements. Both codes use analytic gradient and Hessian evaluations, since finite difference approximations for the inverse mean-ratio metric are inefficient by comparison.

In order to minimize the number of floating-point operations performed per iteration, the coordinate descent algorithm has separate evaluation routines for taking the gradient and Hessian with respect to each vertex in the inverse mean-ratio metric. Each routine is obtained by applying an even permutation to the input data (coordinates for both the trial and reference elements), computing W^{-1} for the permuted reference element, using rotation matrices on the resulting W^{-1} to produce an upper triangular matrix $\bar{W}^{-1}$, and taking the gradient and Hessian with respect to the desired vertex of this equivalent function definition. All of these operations are performed offline for the given weight matrix, and the resulting code can be further refined to reduce operation counts. In particular, for the equilateral weight matrix, $\bar{W}^{-1}$ is the same for all four routines. For a different weight matrix, however, four different versions of the weight matrix may be required. The savings attributed to this approach are significant compared to a simple implementation that uses a single Hessian evaluation routine for the entire element; for equilateral tetrahedral elements, the cost per iteration of the simple implementation is over three times that of the efficient implementation.

One of the main computational tasks associated with the inexact Newton method is obtaining an efficient evaluation for the Hessian of the global objective function. This computation requires obtaining the Hessian for each of the individual element functions. The code for calculating the gradient of the element function uses the reverse mode of differentiation [16] on the element quality metric. The Hessian calculation uses the forward mode of differentiation on the gradient evaluation and matrix-matrix products for efficiency. The other significant computational task is the matrix-vector products required by the conjugate gradient method to compute the search direction.

In order to obtain good locality of reference in the Hessian evaluation and matrix-vector products, the vertices and elements in the initial mesh are reordered by applying a breadth-first search. The ordering starts by selecting the (boundary) vertex farthest from the origin. A breadth-first search of the vertices in the mesh is then performed, and the order they are visited is tracked. We then reverse the order in which the vertices were visited to obtain the reordering for the problem. Once the vertices are reordered, the elements are then reordered according to when they are visited by the Hessian evaluation. This reordering is used by both the coordinate descent and inexact Newton methods.

Each iteration of the coordinate descent method consists of computing the gradient and Hessian for each subproblem, obtaining the direction, and computing each improving point. The gradient and Hessian evaluation is the most expensive operation and requires a total of 508 floating-point operations per element in the tetrahedral mesh. Each iteration of the inexact Newton method consists of computing the gradient and Hessian evaluation of the global objective function, obtaining the direction by the conjugate gradient method, and computing the improving point. The gradient and Hessian evaluation in this case requires 689 floating-point operations per element in the mesh. Just looking at the cost of obtaining the gradient and Hessian information, we can see that the coordinate descent method should be faster per iteration than the inexact Newton method.

In addition to the computational effort, we are also interested in evaluating the memory footprint of each method as the problem size increases. Our implementation of the coordinate descent method for tetrahedral elements has a steady-state memory requirement of approximately $23V + 12E$ integer values, where V is the number of vertices and E is the number of elements in the mesh. The formula for memory usage of the inexact Newton method is more complicated due to the storage requirements for the Hessian matrix and is given by $64V + 18E + 19NB$ integer values, where NB denotes the number of off-diagonal blocks in the Hessian matrix. On the tetrahedral meshes tested, the number of off-diagonal blocks is bounded above by the number of elements in the mesh. Therefore, the memory usage is approximately $64V + 37E$ integer values for the inexact Newton method, about three times the storage required for the coordinate descent method.

Conclusion 1: The coordinate descent method is faster per iteration and consumes less memory than the inexact Newton method but has a slow asymptotic convergence rate. Furthermore, the inexact Newton method requires a higher initial investment in coding routines to assemble the global Hessian matrix from the element Hessian matrices, construct the preconditioner, and perform the preconditioned conjugate gradient method to compute the direction. Once this infrastructure has been built, however, changing to a

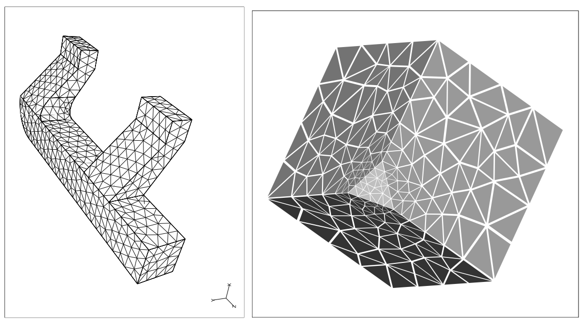


Figure 1: Sample meshes on the duct and clipped cube geometries.

new metric requires only an efficient computation of the element Hessian matrix. To change the metric for the coordinate descent method, we need to implement four different gradient and Hessian evaluation routines. Moreover, if the new metric is not rotationally invariant, then the rotation operations used with the inverse mean-ratio metric cannot be performed, and a different technique must be devised to obtain an efficient coordinate descent method.

4. NUMERICAL EXPERIMENTS

In this section, we perform numerical tests to evaluate the block coordinate descent and inexact Newton techniques to determine if and when one method is preferred to the other using a subset of the criteria given in Section 1. We solve the optimization problem (2) on a series of tetrahedral meshes generated with the CUBIT [17] and GRUMMP [18] mesh generation packages. We consider two different computational domains, duct and clipped cube, and show sample meshes on these geometries in Figure 1. In this paper, we study the effects of three different problem parameters on the time taken to reach x^*: problem size, element heterogeneity, and initial mesh configuration. For each parameter studied, we create a suite of test meshes in which we isolate the parameter of interest, allowing it to vary, while simultaneously holding the other parameters as constant as possible.

In each of the following subsections, we give the problem characteristics of the test suite in terms of the number of vertices and elements, initial mesh quality as evaluated by the inverse mean-ratio metric, and specific parameter values used such as the perturbation of the initial mesh. We then provide the results for both the inexact Newton and coordinate descent mesh quality improvement techniques. For the inexact Newton method, the maximum number of solver iterations is 500, and the maximum number of CG subiterations is 100, while for the coordinate descent method, the maximum number of iterations, that is, passes through the free vertices, is 1000. In both cases, the solution is considered to be optimal when the norm of the gradient is less than 1.0×10^{-6}.

4.1 Increasing Problem Size

For the duct geometry shown in Figure 1, we used CUBIT to generate tetrahedral meshes with uniform quality and element size but with an increasing number of vertices. In Table 1, we give the number of vertices, V, and elements, E, along with the average, median, and standard deviation normalized by the average value, denoted σ_n, for the inverse mean-ratio metric, $\frac{1}{\mu d}$, and element volume, e_{vol}. One can see that within each mesh, we achieve roughly uniform element size and shape quality distributions while increasing the problem size from 4,104 elements to 965,759 elements. In addition, the element quality characteristics are similar across this suite of initial meshes as the problem size increases. In particular, the initial mesh quality is quite good, with an average inverse mean-ratio value of 1.2 (optimal is 1) and a maximum value ranging from 2.2 to 5.

We optimized each test mesh until a very accurate solution was computed, and in Table 2, we give the number of iterations, I, and time, T_{100}, required to achieve the optimal solution. In all cases, both I and T_{100} are significantly smaller for the inexact Newton solver because of its superior asymptotic convergence rate. As the problem size increases, the disparity in time to solution increases monotonically from a factor of 6.4 to a factor of 40.

However, a highly accurate solution is often not required in mesh smoothing applications. Therefore, the time required to reach a partially improved mesh is also of interest. As a particular example, we include the time required to achieve 50% of the optimal solution as defined by the global objective function value, T_{50}, in Table 2. For every mesh in this test suite, the coordinate descent method takes less time than the inexact Newton method to reach this suboptimal solution, typically by a factor of 1.5.

To examine this behavior more closely, we recorded the objective function and gradient values at each iteration. A typical time history is shown for the Duct15 mesh in Figure 2. Because the inexact Newton method converges to the optimal solution much more quickly than the coordinate descent method, we show the complete time history of the inexact Newton solver and only the corresponding portion of the coordinate descent method. Because the initial mesh quality is very good, both methods make significant progress toward the optimal solution in their first few iterations. However, significant setup overhead is associated with computing the sparsity pattern of the Hessian matrix for

Table 1: Initial mesh characteristics for increasing problem size on the duct geometry.

Mesh	V	E	Inverse Mean Ratio				Element Volume		
			avg	median	σ_n	max	avg	median	σ_n
Duct20	1067	4104	1.208	1.176	.115	2.2	1167	1176	.285
Duct15	2139	9000	1.210	1.179	.116	2.1	532	519	.304
Duct12	4199	19222	1.209	1.182	.111	2.1	249	237	.327
Duct10	7297	35045	1.120	1.170	.106	2.2	136	128	.310
Duct8	13193	65574	1.19	1.162	.105	2.4	73	68	.320
DuctBig	177887	965759	1.18	1.160	.109	4.9	4.1	2.91	.587

Table 2: Number of iterations, total time, and time to achieve a 50% optimal solution as problem size increases.

	Newton			Coordinate Descent		
V	I	T_{100}	T_{50}	I	T_{100}	T_{50}
1067	4	.05	.015	33	.32	.005
2139	5	.13	.025	46	1.1	.011
4199	5	.34	.056	74	4.2	.037
7297	5	.69	.106	105	11.6	.081
13193	5	1.4	.213	146	31.0	.152
177887	8	44.3	4.52	548	1738	2.47

the inexact Newton method because the edges in the mesh need to be sorted with a radix sort. During this setup time, the coordinate descent method is able to complete one iteration through the mesh, which is sufficient to achieve 46% of optimality. Clearly, there is a point at which the inexact Newton solution is more accurate than the coordinate descent solution. We call this point the *crossover point*, and it is highlighted with an asterisk in Figure 2. In the Duct15 case, the mesh is 96% optimized when the crossover point occurs.

We address the questions: "What is the percent accuracy achieved at the crossover point?" and "What is the time required by each method to achieve a certain level of optimality?" To answer these questions, for each method we plot the percent accuracy obtained, the number of coordinate descent iterations, and the percentage of time spent in setup by each solver at the crossover point as a function of an increasing problem size in the top graph in Figure 3. In all cases, the mesh is nearly optimal at the crossover point even though the number of coordinate descent iterations completed is quite small, typically less than five. As the problem size increases, the setup time for the inexact Newton solver is greater than 25% of the time to reach the crossover point and typically less than 10% for the coordinate descent method. In this case, the setup time is the primary factor in determining which solver reaches suboptimal solutions faster.

In the bottom graph in Figure 3, we show the ratio

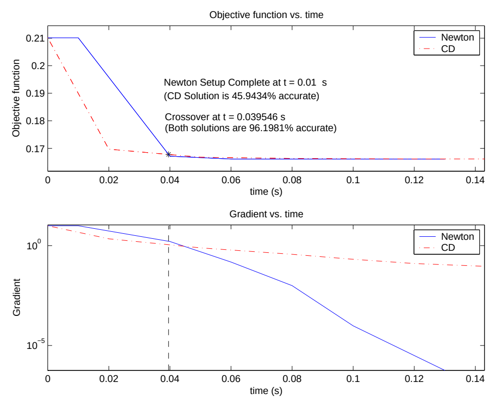
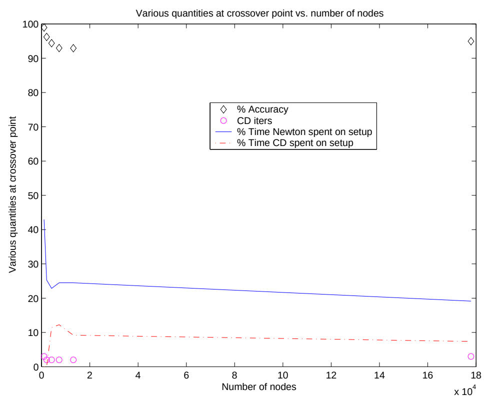


Figure 2: Objective function value and gradient norm as a function of time for the Duct15 mesh.

of the time required by the inexact Newton solver and coordinate descent solver to achieve certain levels of accuracy. Each line in the graph represents a different problem size, and the flat line at 1 represents the point at which the preferred method changes. Data above this line indicates that the coordinate descent method is faster; data below indicates the opposite. In this case, we see that the smaller problem sizes are more affected by the setup time differences, but as the problem size exceeds 20,000 elements, the methods behave similarly. In particular, it takes roughly twice as long to compute suboptimal meshes using the inexact Newton approach for a wide range of desired accuracies. As the accuracy level increases, the inexact Newton method becomes more competitive, but only when nearly optimal meshes are desired does the inexact Newton method outperform the coordinate descent method.

Conclusion 2: For good-quality, uniformly sized tetrahedral meshes, the inexact Newton method outperforms the coordinate descent method if an optimal mesh is desired. If a suboptimal mesh is sufficient, the coordinate descent method typically outperforms the inexact Newton method because of the high setup costs associated with computing the sparsity pattern of the Hessian matrix, which cannot then be amortized



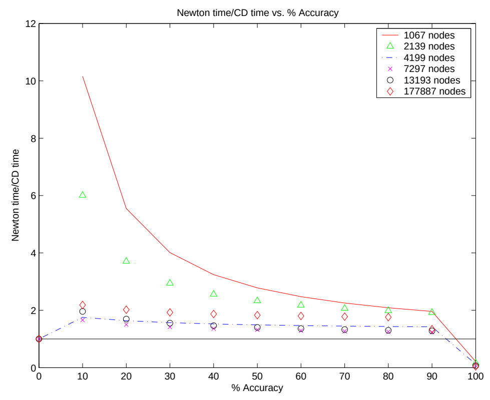


Figure 3: Various quantities of interest at the crossover point as the problem size increases (top) and the ratio of the times required by the inexact Newton and coordinate descent solvers to achieve certain levels of accuracy (bottom).

over a large number of iterations because we start from a nearly optimal solution. This conclusion is true for a wide range of problem sizes, including the largest.

We emphasize that the conclusion in this section has been demonstrated only for the mean-ratio metric and harmonic mean template function on initial meshes with (nearly) uniformly sized, well-shaped tetrahedral elements. Whether it extends to other situations remains to be seen. In the remainder of this section, we examine the behavior of the solvers under different conditions.

4.2 Element Size Heterogeneity

Our second test suite was generated using GRUMMP with the aim of testing the effect of element size (volume) heterogeneity on the two solvers. A simple geometry consisting of the unit cube with a small tetrahedral volume clipped from one corner was used to create graded meshes with grid points clustered around that corner. GRUMMP parameters that determined the smallest element size and gradation of the mesh were manipulated to create a series of meshes with roughly the same number of vertices and element quality distribution but with different ranges of element sizes. In Table 3, we give the statistics for these meshes in terms of numbers of vertices, elements, shape quality distribution, and element volume. The numbers of vertices, although not identical, are all within 6% of 10,470. The normalized standard deviation of the element volumes and the ratio of the maximum-sized element to the minimum-sized element show that the element size varies dramatically within a given mesh. The shape quality distributions across the meshes in the test suite are similar; the average shape quality is nearly the same as in the uniform element size test cases, but the normalized standard deviation is higher, indicating a wider range of individual element qualities. In particular, the maximum mean ratio of the heterogeneous element size meshes exceeds a value of 15 in all cases, whereas it is approximately 2.5 in the uniform element size test suite.

In Table 4, we give the number of iterations and the times to reach the optimal and 50% accurate solutions as the element heterogeneity, measured by the ratio of maximum element volume to minimum element volume, increases. As with the uniform element distribution test suite, the inexact Newton method is significantly faster than the coordinate descent method when the optimal solution is desired. If a 50% accurate solution is desired, however, the coordinate descent method outperforms the inexact Newton method by a factor that ranges from 1.5 to 4, compared to the factor of 1.5 for the uniform distribution case of the previous subsection.

We examine the convergence history of one particular case, Hetero4, to obtain insight into this result. Figure 4 shows the value of the objective function and gradient as a function of time for both the inexact Newton and coordinate descent methods. The coordinate descent method maintains a steep initial decrease in the objective function value, while the inexact Newton method has more difficulty. In particular, many of the initial iterations of the inexact Newton method encounter directions of negative curvature because the global objective function is not convex at that iterate. Typically, a small step is taken when such directions are found by the conjugate gradient method. Thus, the superior asymptotic convergence rates of the inexact Newton method are not evident until approximately two seconds have elapsed.

Because the inexact Newton method has difficulty with these problems in the initial iterations, the coordinate descent method has the time to take several iterations, and the accuracy of the solution is near 100% at the crossover point in all cases. As before, the inexact Newton method uses twice as much time

Table 3: Initial mesh characteristics for heterogeneous element size distributions.

Mesh	V	E	Inverse Mean Ratio				Element Volume			
			avg	median	σ_n	max	avg	median	σ_n	max/min
Hetero1	10318	54132	1.271	1.171	.330	17.1	$1.84{\cdot}10^{-5}$	$1.10{\cdot}10^{-5}$	1.11	$5.5{\cdot}10^{3}$
Hetero2	9883	56184	1.274	1.172	.371	30.2	$1.77{\cdot}10^{-5}$	$6.04{\cdot}10^{-6}$	1.46	$2.3{\cdot}10^{5}$
Hetero3	10926	58610	1.275	1.173	.413	58.6	$1.70{\cdot}10^{-5}$	$3.89{\cdot}10^{-7}$	1.74	$7.2{\cdot}10^{7}$
Hetero4	11057	59985	1.272	1.173	.322	16.1	$1.66{\cdot}10^{-5}$	$1.59{\cdot}10^{-6}$	2.08	$4.2{\cdot}10^{6}$

Table 4: Number of iterations, total time, and time to achieve 50% optimal solution as the element heterogeneity increases.

Mesh	Newton			Coordinate Descent		
	I	T_{100}	T_{50}	I	T_{100}	T_{50}
Hetero1	16	4.24	.386	674	122	.128
Hetero2	13	3.53	.345	708	132	.192
Hetero3	21	5.41	.432	505	93	.207
Hetero4	15	4.22	.641	554	109	.168

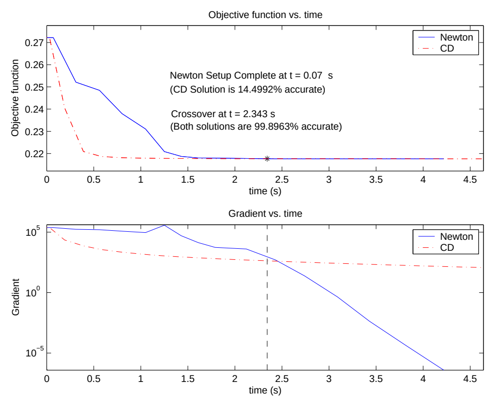


Figure 4: Objective function value and gradient norm as a function of time for the Hetero4 mesh.

in setup as does the coordinate descent method. Unlike the uniform element size test suite, however, this is not the dominant factor in determining the crossover time because both methods use less than 5% of their total time in setup.

For this test suite, in Figure 5 we show the ratio of the time required by the inexact Newton method and the time required by the coordinate descent method to reach a number of different accuracies. For comparison, we also show the curves for the two uniform element size test cases, Duct10 and Duct8, which tightly bracket the number of elements in the clipped cube meshes. The normalized standard deviation values for the Duct 10 and Duct8 meshes are .310 and .320, respectively. In almost all cases, the coordinate descent

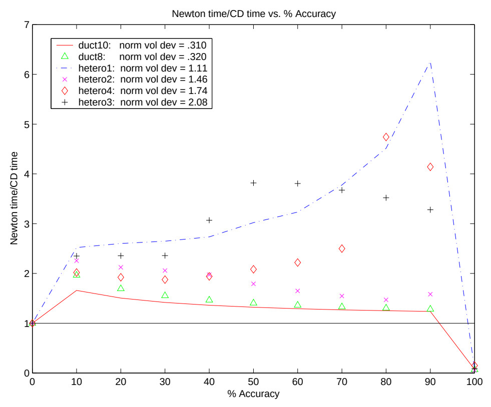


Figure 5: Ratio of the inexact Newton and coordinate descent times for various levels of accuracy in the objective function.

method is two to three times faster than the inexact Newton method to reach a desired level of accuracy. In fact, in several cases, the ratio of the times required to achieve higher levels of accuracy actually increases rather than decreases as a result of the steep initial convergence of the coordinate descent method. Furthermore, the coordinate descent method is more competitive than the inexact Newton method on the heterogeneous element size meshes than on the uniformly sized element problems.

Conclusion 3: As element size heterogeneity increases and mesh quality decreases for tetrahedral meshes, the coordinate descent method is increasingly attractive for suboptimal solutions. If the optimal solution is desired, the inexact Newton method is preferred.

4.3 Initial Mesh Configuration

The final mesh test suite was designed to investigate the effect of the degree and manner in which the initial mesh differs from the optimal mesh. To address this issue, our approach was to apply systematic or random perturbations of the optimal positions of the interior mesh vertices. We started with the optimized DuctBig mesh and applied three different perturbation schemes that involved random, translational, and oscillatory movement of the mesh vertices. In all three cases, we consider perturbations applied to all of the vertices and to randomly chosen vertex subsets of size 1000, 2000, and 5000. The formulas for the perturbations are as follows:

Random: $x_v = x_v + a\mathbf{r}$, where $\mathbf{r}$ is a three-vector containing random numbers generated using the function `rand` and a is a multiplicative factor controlling the degree of perturbation. For this test suite, we chose a= .001, .005, .01, and .05.

Translational: $x_v = x_v + a\mathbf{s}$, where $\mathbf{s}$ is a direction vector giving the coordinates to be shifted and a is again a multiplicative factor controlling the degree of perturbation. In this case we considered a "right" shift (R) with $\mathbf{s}_1 = [1\ 0\ 0]^T$ and a "northeast" shift (NE) with $\mathbf{s}_2 = [1\ 1\ 0]^T$ and chose $a = .1, .5$, and .7.

Oscillatory: $x_v = x_v + b\sin(ax_v)$, where the scalars a and b control the frequency and amplitude of the perturbation, respectively. For this test suite, we considered two different amplitudes, b=.01 and .05, and, for each amplitude, three different frequencies $a = .01$, .05, and .1. The wavelength of the perturbation can be computed from the frequency by $w = 2\pi/a$, and we note that for this test suite w ranges from 63 to 628, which exceeds the average edge length of the mesh of approximately 3.6.

The perturbation amounts were chosen to avoid inverting any of the elements of the initial mesh. The resulting quality characteristics of the meshes as the perturbations increase do not vary significantly, and we do not include the details. In particular, the average inverse mean ratio value is approximately 1.13, the ratio of the standard deviation to the average is approximately .093, and the maximum inverse mean ratio is approximately 2.21 in all cases.

In Table 5, we give the iterations and time to reach the 100% and 50% accurate solutions for the cases in which we perturb all of the vertices in the mesh according to formulas and parameters given above. As with the other test suites, if a highly accurate solution to the optimization problem is sought, the inexact Newton method outperforms the coordinate descent method in every case. If the 50% solution is sought, the coordinate descent method outperforms the inexact Newton method for the random and translational test cases. For these cases, the solutions are greater than 95% accurate at the crossover points. The setup time is the primary factor determining the crossover point and requires about 30% of the inexact Newton solution time compared to approximately 10% for coordinate descent. If a 50% accurate solution is sought for the oscillatory test suite, the results are mixed. For both amplitudes considered, as the wavelength increases, the inexact Newton method outperforms the coordinate descent method.

Table 5: Number of iterations and total time to achieve 100% and 50% of the optimal solution as the element heterogeneity increases.

		Newton			Coordinate Descent		
Pert.	a	I	T_{100}	T_{50}	I	T_{100}	T_{50}
Rand	.001	3	18.0	3.17	176	586	2.23
	.005	4	25.0	3.18	220	729	2.28
	.01	4	25.0	3.18	245	792	2.31
	.05	6	33.0	3.17	305	995	2.26
Trans (R)	.1	3	17.5	3.10	139	439	2.04
	.5	3	17.5	3.11	142	445	2.02
	.7	3	17.5	3.10	145	459	1.97
Trans (NE)	.1	3	17.6	3.12	138	437	2.02
	.5	3	17.8	3.19	146	455	1.97
	.7	3	17.7	3.21	150	469	1.94
Osc. (b=.01)	.01	3	19.8	3.64	268	885	2.06
	.05	3	21.1	4.35	360	1124	5.34
	.1	3	21.5	4.39	309	989	14.2
Osc. (b=.05)	.01	3	20.7	4.12	329	1072	2.42
	.05	3	22.4	4.82	421	1324	17.3
	.1	4	28.2	4.41	363	1173	15.6

In Figure 6, we examine the ratio of the time required by the inexact Newton method and the coordinate descent method as the desired degree of optimality increases. In all cases considered, as a more accurate solution is sought, the inexact Newton method looks increasingly attractive. For the random and translational test suites, however (top left and middle figures, respectively), the coordinate descent method always outperforms it up to 90% accuracy. As the degree of perturbation increases in the translational test suite, the relative performance of the coordinate descent method improves, whereas the relative performance is unaffected as the degree of perturbation increases in the random test suite. For the oscillatory test cases, we separate the plots for the two different amplitudes (top right). For both amplitudes, as the wavelength increases, the inexact Newton method is the method of choice for all levels of accuracy for high amplitude or long wavelength perturbations. In this case, the local nature of the coordinate descent method can only slowly eliminate the long wavelength errors introduced by the perturbation scheme. In contrast, the inexact Newton method has access to global information and is able to overcome this difficulty.

To determine whether the number of vertices that we perturb affects the relative performance of the two solvers, we give the ratio of the inexact Newton method and coordinate descent method times to achieve various levels of accuracy (see the bottom row

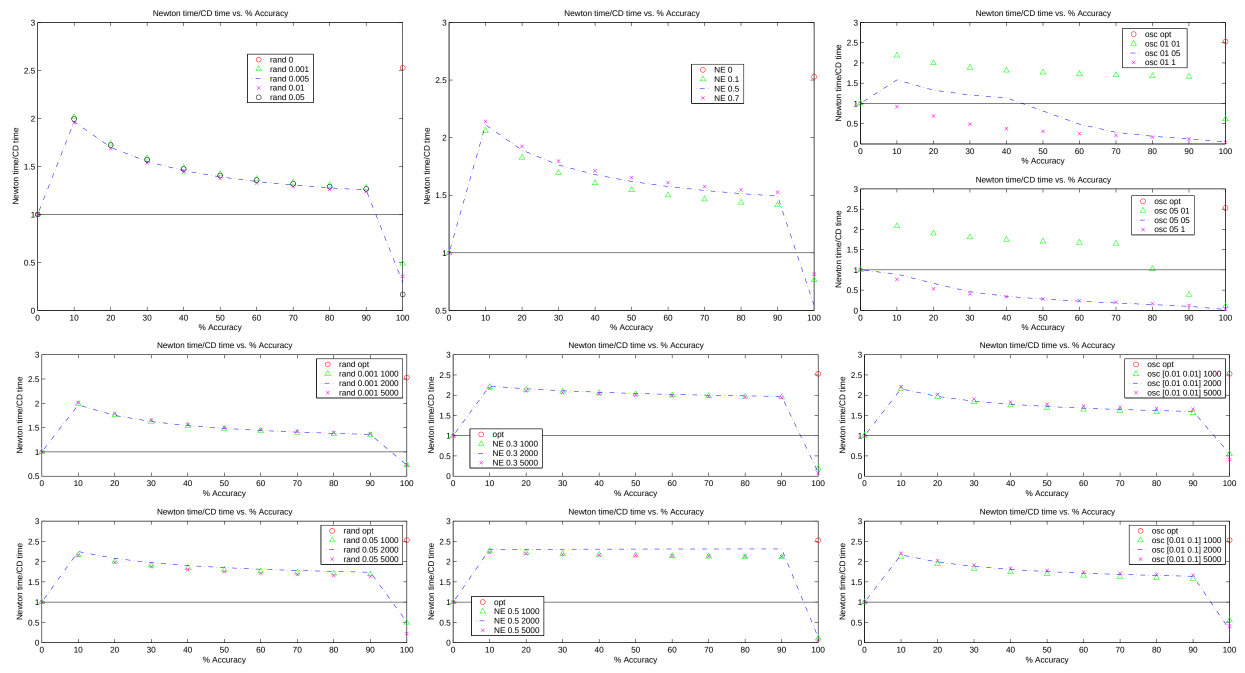


Figure 6: Ratio of the times required by the inexact Newton and coordinate descent solvers to achieve certain levels of accuracy for the random (top left), NE translational (top middle), and oscillatory (top right) perturbations and for the cases where a subset of vertices are perturbed randomly (bottom left), NE translational (bottom middle), and oscillatory (bottom right).

of Figure 6) when a subset of the vertices is perturbed. The number of randomly selected vertices perturbed is 1,000, 2,000, and 5,000. We show results for only a subset of the cases analyzed; the results for the cases not shown are qualitatively the same. In all cases, the inexact Newton method is unable to outperform the coordinate descent method for suboptimal solution values. This result is particularly interesting for the oscillatory perturbations (bottom right). In this case, because only a subset of the vertices were perturbed, the long wavelength errors that affected the performance of the coordinate descent method are no longer evident, and the coordinate descent method is again able to quickly approach the optimal solution. In the case of the translational perturbations (bottom middle), we see that as we increase the number of vertices perturbed, the relative performance of the coordinate descent method decreases.

Conclusion 4: For random and translational perturbations of uniform tetrahedral meshes, the coordinate descent method outperforms the inexact Newton method if suboptimal solutions to improve shape are sought. Large wavelength errors, such as those introduced by the oscillatory perturbations of all the vertices, present difficulties for the coordinate descent method, and the inexact Newton method is often preferred.

5. FUTURE WORK

The numerical results show that the block coordinate descent method is most useful for making fast improvement in the shape quality of tetrahedral meshes. The inexact Newton method is the most appropriate when a very accurate solution to the optimization problem is necessary. We note that a 50% optimization level may be somewhat misleading in that it does not indicate the degree of the accuracy in the solution. For most of the Duct meshes, the iterate at a 50% optimization level has four or five digits of accuracy in the objective function value. If the initial quality of the mesh is very poor, however, then the iterate at a 50% optimization level may not have any digits of accuracy in the objective function value.

Because of the size limitations of this paper, the impact of several important factors could not be investigated. In particular, we limited ourselves to shape improvement of tetrahedral meshes and ignored triangular, quadrilateral, and hexahedral meshes. Preliminary experience with planar quadrilateral meshes shows that the crossover point for our two codes generally occurs earlier, but further study is warranted.

Furthermore, because the global objective function is not convex, a trust-region method for the inexact Newton code may be perform better since it can handle directions of negative curvature more rigorously. A limited-memory quasi-Newton method may also be better than the coordinate descent method at obtaining an approximate solution in a small amount of time. Efficient implementations of these methods can be based on the infrastructure developed for the inexact Newton and coordinate descent methods.

In conclusion, there are many factors which can affect whether or not one should use a coordinate descent solver or an inexact Newton method for mesh optimization. The present work identifies many of the potentially important factors and develops a methodology for investigations on this topic. Future work will consider many of the remaining open questions.

ACKNOWLEDGMENTS

The initial version of the analytic gradient for the inverse mean-ratio metric for tetrahedral elements was provided by Paul Hovland. The clipped cube mesh image was provided by Carl Ollivier-Gooch.

The work of the first and third authors was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contracts W-7405-Eng-48 (UCRL-CONF-205150) and W-31-109-Eng-38, respectively. The work of the fourth author was supported by Sandia National Laboratories, Cornell University, and the National Physical Science Consortium.

References

[1] Bank R., Smith B. “Mesh Smoothing Using A Posteriori Error Estimates.” *SIAM J. Num. Anal.*, vol. 34, 979–997, 1997

[2] Canann S.A., Stephenson M.B., Blacker T. “Optismoothing: An Optimization-Driven Approach to Mesh Smoothing.” *Finite Elements in Analysis and Design*, vol. 13, 185–190, 1993

[3] Parthasarathy V.N., Kodiyalam S. “A Constrained Optimization Approach to Finite Element Mesh Smoothing.” *Finite Elements in Analysis and Design*, vol. 9, 309–320, 1991

[4] Zavattieri P., Dari E., Buscaglia G. “Optimization Strategies in Unstructured Mesh Generation.” *Int'l. J. Numer. Meth. Engr.*, vol. 39, 2055–2071, 1996

[5] Castillo J. “A Discrete Variational Grid Generation Method.” *SIAM J. Sci. Stat. Comp.*, vol. 12, 454–468, 1991

[6] Freitag L., Knupp P. “Tetrahedral Mesh Improvement via Optimization of the Element Condition Number.” *Intl. J. Num. Meth. Engr.*, vol. 53, 1377–1391, 2002

[7] Anderson D. “Grid Cell Volume Control with an Adaptive Grid Generator.” *Appl. Math. and Comp.*, vol. 35, 209–217, 1990

[8] Knupp P. “Algebraic Mesh Quality Metrics.” *SIAM Journal on Scientific Computing*, vol. 23, 193–218, 2001

[9] Knupp P. “Matrix Norms and the Condition Number.” *Proceedings of 8th International Meshing Roundtable*, pp. 13–22. 1999

[10] Bertsekas D.P. *Nonlinear Programming.* Athena Scientific, Belmont, Massachusetts, second edn., 1999

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

[12] Munson T.S. “Mesh Shape-Quality Optimization Using the Inverse Mean-Ratio Metric.” Preprint ANL/MCS-P1136-0304, Argonne National Laboratory, Argonne, Illinois, 2004

[13] Munson T.S. “Mesh Shape-Quality Optimization Using the Inverse Mean-Ratio Metric: Tetrahedral Proofs.” Technical Memorandum ANL/MCS-TM-275, Argonne National Laboratory, Argonne, Illinois, 2004

[14] Armijo L. “Minimization of Functions Having Lipschitz-Continuous First Partial Derivatives.” *Pacific Journal of Mathematics*, vol. 16, 1–3, 1966

[15] Saad Y. *Iterative Methods for Sparse Linear Systems.* SIAM, Philadelphia, Pennsylvania, second edn., 2003

[16] Griewank A. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* SIAM, Philadelphia, Pennsylvania, 2000

[17] Sandia National Laboratories, Albuquerque, New Mexico. *CUBIT 8.1 Mesh Generation Toolkit*, 2003

[18] Ollivier-Gooch C.F. “GRUMMP — Generation and Refinement of Unstructured Mixed-Element Meshes in Parallel.” http://tetra.mech.ubc.ca/GRUMMP, 1998–2002