ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

Optimizing the Quality of Mesh Elements

Todd Munson

Mathematics and Computer Science Division

Preprint ANL/MCS-P1260-0605

June 2005

[1]This work 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 Contract W-31-109-Eng-38. Any opinions or errors are the responsibility of the authors and not the sponsoring agency.

Optimizing the Quality of Mesh Elements

Todd S. Munson
Mathematics and Computer Science Division,
Argonne National Laboratory, Argonne, IL 60439
(tmunson@mcs.anl.gov).

1. Introduction

Discretization methods are common techniques for computing approximate solutions to partial differential equations [7, 10, 30]. These methods decompose the given domain into a finite set of elements, triangles or tetrahedrons, for example, to produce a mesh used within the approximation scheme. Several factors affect the accuracy of the solution obtained: the degree of the approximation scheme and the number of elements in the mesh [2], and the quality of the mesh [4, 5]. Optimizing the quality of the mesh prior to computing the approximate solution can improve the condition number of the linear systems solved [29], reduce the time taken to compute the solution [15], and increase the numerical accuracy.

The savings in computational time from using the optimized mesh can be substantial. One application we investigated was to solve the Navier-Stokes equations for a fluid with a moderate Reynolds number containing a dilute suspension of particles [34]. The approximate solution was obtained by applying a spectral element method to a hexahedral mesh. The top portion of Figure 1 shows the original mesh constructed by applying the meshing technique developed by Lin Zhang, while the bottom depicts the optimized mesh. The original mesh has many regular elements, while the optimized mesh loses much of this structure. However, the spectral element method applied required 29 hours to compute a solution when using the original mesh, but only 20 hours when using the optimized mesh, a 30% reduction in time. The optimization problem was modeled in AMPL [11] and solved by applying KNITRO [8, 33]. This instance had 18,135 variables, 1,170 linear constraints, and 3,004 nonlinear constraints. Computing an optimal mesh took approximately 33 minutes on a 2.4 GHz Intel Xeon workstation with 2 GB RAM; AMPL consumed 265 MB RAM to generate the problem, while KNITRO allocated 597 MB RAM to solve it.

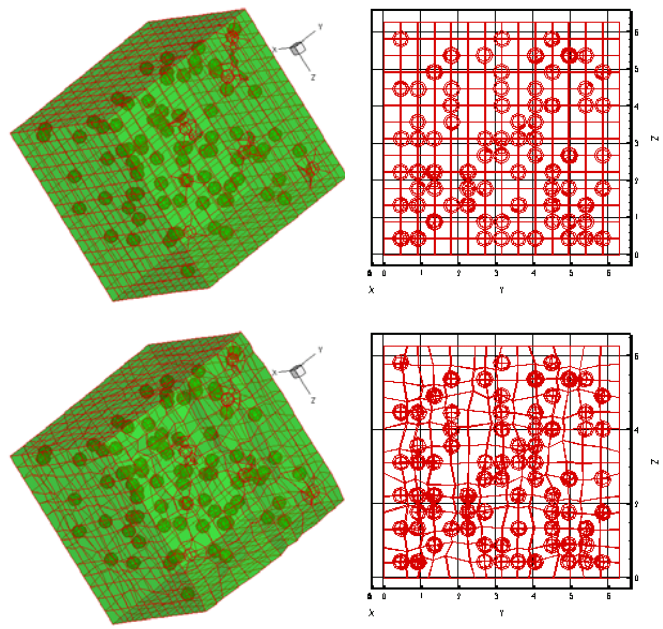

Figure 1: Original mesh (top left) and side view (top right) and optimized mesh (bottom left) and side view (bottom right) for fluid dynamics example.

The optimization problem we solve computes positions for the vertices in a given mesh to improve the average element quality according to a metric [13]; we do not change the mesh topology. Many metrics have been applied to such problems [3, 12, 14, 19, 20, 27]. We use the inverse mean-ratio metric [21, 20], a shape-quality metric measuring the distance between a trial element and an ideal element, a regular tetrahedron, for example. The objective function for the resulting optimization problem is nonconvex and consists of the sum of many fractional terms. The optimization problem and its properties are developed in Section 2. Proofs for the claims made in this section are found in [22, 23].

Applications with moving meshes or deforming geometries may require a mesh optimization step every time the domain is modified. For example, the particles in the fluid dynamics application could move as a function of time. In general, the time required to optimize the mesh must not dominate the computational savings achieved when solving the partial differential equation. Therefore, we want to compute an optimal mesh in a minimal amount of time and with minimal memory requirements. We dis-

cuss in Section 3 several techniques for improving the performance of the KNITRO libraries on an unconstrained version of the mesh optimization problem.

In Section 4 we discuss some of the lessons learned while working on this application. We also mention some of the complications associated with efficiently solving the fluid dynamics example where the vertices on the planar boundaries of the mesh are allowed to move within the plane.

2. Mesh Optimization Problem

The mesh optimization problem we solve minimizes the average inverse mean-ratio metric referenced to an ideal element. The description of this metric follows that of Knupp [19, 20] and Freitag and Knupp [12]. We discuss the metric only for tetrahedral elements. Each tetrahedron is defined by four vertices $[x_1, x_2, x_3, x_4]$, where each vertex belongs to $\Re^3$. Other element types can be modeled by decomposing them into tetrahedrons. Hexahedral elements, for example, are decomposed into eight overlapping tetrahedral elements.

The development of the inverse-mean ratio metric begins by constructing the incidence function $A : \Re^{3\times 4} \to \Re^{3\times 3}$:

$$A(x) := [x_2 - x_1, x_3 - x_1, x_4 - x_1] .$$

This function computes a matrix containing the edges emanating from the first vertex of the element. The volume of the tetrahedron is related to the determinant of $A(x)$, which can be positive or negative depending on the labeling of the vertices. We assume throughout that the vertices are labeled according to the right-hand rule so that the determinant is nonnegative.

Two elements x and y have the same shape if their edges are proportional. That is,

$$A(x) = \sigma A(y)$$

for some $\sigma > 0$. In particular, if two elements with nonzero volume have the same shape, then

$$\|A(x)A(y)^{-1}\|_F^2 = \|\sigma I\|_F^2 = 3\sigma^2$$

and

$$\det(A(x)A(y)^{-1}) = \det(\sigma I) = \sigma^3.$$

The inverse mean-ratio referenced to the given element y is then defined by a function $Q_y : \Re^{3\times 4} \to \Re$ that takes the ratio of these two quantities. In particular,

$$Q_y(x) := \frac{\|A(x)A(y)^{-1}\|_F^2}{3\det(A(x)A(y)^{-1})^{2/3}}.$$

This metric has a value of one if x and y have the same shape and a value greater than one if their shapes differ. Moreover, it is translation, rotation, and scale invariant. The metric values are preserved when the same even permutation is applied to the columns of x and y. Since y is fixed, this metric is computed by using the QR factorization of $A(y)$ so that we multiply $A(x)$ only by the upper triangular matrix R^{-1}. This modification reduces the number of floating-point operations required to compute the function, gradient, and Hessian of Q_y.

A mesh consists of a set of vertices V and the elements E connecting these vertices, where each element is an ordered set of four indices. The optimization problem to minimize the average inverse mean-ratio metric is then

$$\begin{array}{lll} \min\limits_{x\in\Re^{3\times|V|}} & \theta(x) := \dfrac{1}{|E|}\displaystyle\sum_{e\in E} \dfrac{\|A(x_e)A(y)^{-1}\|_F^2}{3\det(A(x_e)A(y)^{-1})^{2/3}} & \\ \text{subject to} & \det(A(x_e)A(y)^{-1}) > 0 & \forall e \in E \\ & x_i \in X_i & \forall i \in V, \end{array}$$

where x_e denotes the matrix of coordinates for element e and X_i is a set restricting the feasible locations for vertex i. In particular, the vertices on the boundary of the mesh are usually either fixed in space or constrained to lie on a particular piece of the boundary of the domain, while the other vertices are unrestricted. The volume constraints, the strict inequalities in the optimization problem, ensure a consistent orientation in the resulting mesh. A consistent orientation for all the elements is required for standard discretization methods to work correctly [7]. The objective function is twice continuously differentiable at all feasible points but is not necessarily convex on this region. Furthermore, the feasible region may be neither convex nor connected. While the Hessian of the objective function may not be positive definite, one can prove that $\nabla^2_{x_i,x_i}\theta(x)$ is positive definite for all i [22, 23].

The volume constraints are problematic because they involve a strict inequality. If at least two ver-

tices are fixed in position and the mesh is edge connected (between any two elements there is a sequence of elements whose neighbors share a common edge), then the objective function approaches infinity for any sequence of feasible points in which the volume of at least one element approaches zero [22]. Therefore, the volume orientation constraints can be dropped to produce the optimization problem

$$\begin{array}{ll} \min\limits_{x \in \Re^{3\times|V|}} & \dfrac{1}{|E|}\displaystyle\sum_{e\in E} \dfrac{\|A(x_e)A(y)^{-1}\|_F^2}{3\max\left\{\det(A(x_e)A(y)^{-1}),0\right\}^{2/3}} \\ \text{subject to} & x_i \in X_i \quad \forall i \in V. \end{array}$$

In this case, the objective function is defined to have a value of plus infinity whenever the volume of at least one element is nonpositive. With this reformulation, we must provide a starting point where the orientation constraints are satisfied. However, most meshing packages used to construct the original mesh for the given domain, such as [26, 28], provide a set of vertices satisfying the orientation constraints.

3. Unconstrained Results

In this section, we assume that all vertices on the boundary of the domain are fixed in position and the remaining vertices are unrestricted. Once the boundary vertices are removed from the problem, we are left with an unconstrained optimization problem with an objective function that is twice continuously differentiable on an open set containing the level set for the given feasible mesh.

The resulting unconstrained optimization problem was modeled in AMPL [11] and solved by applying KNITRO 4.0 [8, 33] and LOQO 6.06 [31, 32]. We always used the Interior/CG version of KNITRO and the default version of LOQO for these tests. The results on a representative set of test meshes are given in Table 1, where the number of variables and nonzeros in the Hessian matrix are provided for each example. The times are reported in seconds on a 2.0 GHz Intel Xeon workstation with 4 GB RAM. No other users were on the workstation when the results were generated, and all data and executables were on local disk drives. Each problem was run three times; the lowest time is reported. The largest models could not be solved because of memory requirements. In particular, AMPL consumed 567 MB to generate the duct8 instance, while KNITRO allocated 1,055 MB to solve it and LOQO used 1,008 MB.

Table 1: Unconstrained results using AMPL.

Mesh	Variables	Nonzeros	KNITRO	LOQO
gear	780	8,256	1.87	2.05
foam5	867	10,518	2.34	3.20
hook	1,200	16,872	3.15	5.84
duct20	1,146	17,601	3.12	14.06
duct15	2,895	50,106	7.96	30.80
duct12	6,906	129,102	21.59	108.44
duct10	13,440	262,329	49.27	160.96
duct8	26,214	529,212	124.81	929.33
duct4	425,952	9,209,799	-	-
duct2	3,323,229	45,882,111	-	-

Using AMPL provides many advantages: models can be quickly constructed, derivatives are automatically generated for the functions, and many numerical methods are readily available to solve the resulting instances. In particular, one can obtain results for the mesh optimization problems in a few days. However, a price is paid for this convenience: it can take considerable time to compute the optimal mesh, and the amount of memory consumed can be large.

Since our ultimate goal is to embed mesh optimization within a larger code for solving partial differential equations, we must compute a solution in a short amount of time with a small memory requirement. Therefore, we implemented a simple framework that reads the description of a mesh from a file, constructs the unconstrained optimization problem, calls an optimization routine, and writes the solution back to a file. When reading the mesh, we check it for duplicated vertices, topological errors, and inverted elements, and we compute the vertices on the boundary of the mesh.

The average inverse mean-ratio objective function requires that a value of plus infinity be returned whenever the volume constraints are not satisfied. Therefore, if the volume of at least one element is smaller than 1.0×10^{-14}, we consider the volume constraints to be violated, and the objective function is set to plus infinity. This strategy is reasonable when the mesh is well scaled because the objective function becomes very large.

The gradient and Hessian of the objective function

are calculated analytically by assembling the gradients and Hessians for each element function into a vector and symmetric sparse matrix. In order to facilitate the assembly of the Hessian matrix, once the sparsity pattern is obtained, an additional vector is calculated that tells the offset into the Hessian matrix data vector where the element Hessians are to be accumulated. The gradient and Hessian of the elements with respect to vertices fixed on the boundary of the mesh are ignored.

The code for calculating the gradient of the element function uses the reverse mode of automatic differentiation [6, 16]. The code was written and refined by hand to eliminate unnecessary operations, resulting in a more efficient gradient evaluation. The Hessian calculation uses the forward mode of differentiation on the gradient evaluation. The resulting code was written by using matrix-matrix products for efficiency. AMPL was used to verify the correctness of the analytic gradient and Hessian evaluations.

Table 2 presents the results obtained when using the KNITRO libraries for the optimization solver. Two versions of the code were run: one where the Hessian matrix was directly provided to the code in the (i, j, k) format, the "Hessian" column, and the other where a routine for computing Hessian-vector products was supplied, the "Product" column. The routine stores the upper triangular part of the Hessian matrix in a block compressed sparse row format, where each block corresponds to a vertex-vertex interaction in the original mesh, and performs a Hessian-vector product using this structure. The version of KNITRO using Hessian-vector products is faster than when the Hessian matrix is supplied and significantly faster on the larger examples. Moreover, the block upper triangular structure stores only one index per block, instead of two indices per variable, resulting in some memory savings.

In order to reduce the computational time further, the vertices and elements in the initial mesh were reordered by using a breadth-first search ordering [25] to improve the locality of reference prior to applying KNITRO. Figure 2 shows the sparsity pattern for the Hessian matrix of the original and reordered mesh for the duct8 problem. In particular, the ordering starts by selecting the (boundary) vertex farthest from the origin as a starting point. A breadth-first search of the vertices in the mesh is then performed. The order in which the vertices were visited is reversed, as in the reverse Cuthill-McKee ordering [9], to obtain a symmetric permutation of the vertices for the optimization problem. The elements in the mesh are then reordered according to when they are referenced by the vertices. Other orderings can be applied that may give rise to further improvements in performance [17].

Table 2: Results using KNITRO libraries with original vertex order.

Mesh	Variables	AMPL	Hessian	Product
gear	780	1.87	0.07	0.07
foam5	867	2.34	0.11	0.11
hook	1,200	3.15	0.12	0.11
duct20	1,146	3.12	0.10	0.10
duct15	2,895	7.96	0.26	0.25
duct12	6,906	21.59	0.64	0.63
duct10	13,440	49.27	1.39	1.33
duct8	26,214	124.81	3.65	3.07
duct4	425,952	-	205.11	148.46
duct2	3,323,229	-	-	-

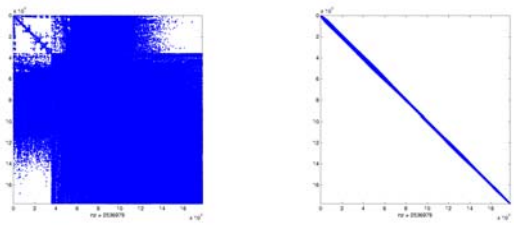

Figure 2: Sparsity pattern of the Hessian matrix for the duct8 mesh with original ordering (left) and the breadth-first search ordering (right).

Table 3 presents the results on the original and reordered meshes. The time for the reordered test problems includes the cost of computing the breadth-first search and reordering the problem data. The savings attributed to reducing the bandwidth of the matrices is considerable, especially on the larger optimization problems.

To further improve performance, we would like to apply an appropriate preconditioner in the conjugate gradient method used within the optimization solver. For the mesh optimization problems we know that the diagonal blocks of the Hessian matrix are positive definite [22, 23], so a block Jacobi precondi-

Table 3: Results using KNITRO libraries with breadth-first search vertex order.

	Original		Reordered	
Mesh	Hessian	Product	Hessian	Product
gear	0.07	0.07	0.07	0.07
foam5	0.11	0.11	0.11	0.11
hook	0.12	0.11	0.12	0.11
duct20	0.10	0.10	0.10	0.10
duct15	0.26	0.25	0.25	0.25
duct12	0.64	0.63	0.59	0.58
duct10	1.39	1.33	1.18	1.16
duct8	3.65	3.07	2.95	2.49
duct4	205.11	145.98	134.06	81.74
duct2	-	-	-	-

Table 4: Results using block Jacobi preconditioner.

Mesh	Variables	AMPL	KNITRO	Newton
gear	780	1.87	0.07	0.06
foam5	867	2.34	0.11	0.09
hook	1,200	3.15	0.11	0.09
duct20	1,146	3.12	0.10	0.07
duct15	2,895	7.96	0.25	0.19
duct12	6,906	21.59	0.58	0.45
duct10	13,440	49.27	1.16	0.88
duct8	26,214	124.81	2.49	1.78
duct4	425,952	-	81.74	46.41
duct2	3,323,229	-	-	324.92

tioner can be applied. Unfortunately, the available version of the KNITRO libraries does not currently allow the user to provide a preconditioner. Therefore, a simple inexact Newton method [18, 24] with an Armijo linesearch [1] was built on the same framework to solve the mesh optimization problems.

In particular, given x^k, the algorithm computes a direction d^k by solving the symmetric system of linear equations

$$\nabla^2\theta(x^k)d^k = -\nabla\theta(x^k)$$

by applying a conjugate gradient method with a block Jacobi preconditioner [25]. Since the Hessian can be indefinite, the conjugate gradient method may terminate with a direction of negative curvature. In such a case, the base of the direction is used as the starting point for the linesearch. The Armijo linesearch finds the smallest nonnegative integer m such that

$$\theta(x^k + \beta^m d^k) \leq \theta(x^k) + \sigma\beta^m \nabla\theta(x^k)^T d^k,$$

where $0 < \sigma < \frac{1}{2}$ and $0 < \beta < 1$ are constants. The iterate is then updated with the rule $x^{k+1} = x^k + \beta^m d^k$, and a new direction is computed. The algorithm terminates when $\|\nabla\theta(x^k)\|_2$ is less than 1.0×10^{-6}.

The conjugate gradient method terminates if the system of equations is solved to within a specified tolerance, if a direction of negative curvature is encountered, or if 100 conjugate gradient iterations have been performed. In particular, the conjugate gradient implementation terminates when

$$\|\nabla^2\theta(x^k)d^k + \nabla\theta(x^k)\|_2 \leq 10^{-2} \times \|\nabla\theta(x^k)\|_2.$$

That is, the relative tolerance is used for the termination test.

Table 4 presents the final results obtained by using the reordered meshes with AMPL, the Hessian-vector product version of KNITRO, and the inexact Newton method with a block Jacobi preconditioner. As expected, the preconditioner helps to further reduce the computational time. The results are less dramatic on the smaller problems because of the fixed time required to set up the problem; the improvement from preconditioning is more dramatic on the larger examples. In particular, the preconditioned code was able to compute a solution to the duct2 problem; the KNITRO libraries were terminated after an hour of computational time without finding a solution.

4. Conclusions

This article discussed an optimization problem for finding the optimal vertex positions in a mesh according to the average inverse mean-ratio metric. Also presented was a computational study of three techniques to improve solver performance on the unconstrained version of the problem. Modifying the data structures, reordering the problem data, and preconditioning the iterative method can significantly reduce the computational time, especially for large instances.

However, developing a framework for a specific problem and validating the result is a time-consuming task recommended only if performance is crucial. Tweaking the implementation to make the code more efficient is another critical step requiring a

significant time investment. After going through this process for the mesh optimization application, we observed that solving the problem through AMPL was 25–70 times slower than using the preconditioned inexact Newton code and could consume over 120 times the memory.

We prefer using solver libraries rather than implementing our own numerical optimization algorithms. However, the design of the library is important to achieve high performance. Matrix-free methods are desirable because they allow the application developer to determine the data structures used to store the matrices and to identify ways to efficiently perform the required matrix-vector products, which can reduce both computational time and memory requirements. Moreover, allowing the user to specify a preconditioner for the iterative methods employed can be beneficial; however, an appropriate strategy to precondition constrained optimization problems is an open question. Reordering the matrices to reduce the bandwidth can also result in significant improvements in time for unconstrained optimization problems. The correct reordering to use for constrained problems is likely algorithm dependent and also an open question.

We remark that the fluid dynamics application in the introduction solved a constrained version of the mesh optimization problem, where the constraints restrict the vertices on the boundary to planes or spheres, rather than fixing them in position. We were able to construct a model for this application because we knew the geometry of the problem. A general-purpose tool for constrained mesh optimization problems would require either knowledge of the geometry or adding a strategy to the framework to uncover simple geometric objects such as planes and ellipses. The latter strategy has not yet been implemented in our framework but is a planned extension. Once constraints are added to the framework, we can investigate the effect of the reordering and matrix-free strategies on the solution time when using the KNITRO libraries. Preconditioning the constrained case and making effective use of a good starting point in an interior-point method are open issues. However, this work is essential for using mesh optimization with the original fluid dynamics application where the particles move as a function of time.

Acknowledgments

This work 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 Contract W-31-109-Eng-38.

Lori Freitag-Diachin, Patrick Knupp, and Suzanne Shontz introduced me to the mesh shape-quality optimization problem and the inverse mean-ratio metric; David Gay provided the elegant version of the volume calculation used in the AMPL models; Paul Hovland wrote the initial version of the analytic gradient of the element function; Richard Waltz kindly provided the version of the KNITRO libraries used; and Lin Zhang, S. Balachandar, and Paul Fischer provided the meshes and performance data for the computational fluid dynamics example.

REFERENCES

[1] L. Armijo. Minimization of functions having Lipschitz-continuous first partial derivatives. *Pacific Journal of Mathematics*, 16:1–3, 1966.

[2] I. Babuška and M. Suri. The p and h-p versions of the finite element method, basic principles and properties. *SIAM Review*, 36:578–632, 1994.

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

[4] M. Berzins. Solution-based mesh quality for triangular and tetrahedral meshes. In *Proceedings of the Sixth International Meshing Roundtable*, pages 427–436. Sandia National Laboratories, 1997.

[5] M. Berzins. Mesh quality – geometry, error estimates or both? In *Proceedings of the Seventh International Meshing Roundtable*, pages 229–237. Sandia National Laboratories, 1998.

[6] C. H. Bischof, P. D. Hovland, and B. Norris. Implementation of automatic differentiation tools. *Higher-Order and Symbolic Computation*, to appear, 2004.

[7] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods*. Springer-Verlag, New York, 2002.

[8] R. Byrd, M. E. Hribar, and J. Nocedal. An interior point method for large scale nonlinear programming. *SIAM Journal on Optimization*, 9:877–900, 1999.

[9] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 24th National Conference ACM*, pages 157–172. ACM Press, 1969.

[10] M. O. Deville, P. F. Fischer, and E. H. Mund. *High-Order Methods for Incompressible Fluid Flows*. Cambridge University Press, Cambridge, 2002.

[11] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Brooks/Cole–Thomson Learning, Pacific Grove, California, second edition, 2003.

[12] L. Freitag and P. Knupp. Tetrahedral mesh improvement via optimization of the element condition number. *International Journal for Numerical Methods in Engineering*, 53:1377–1391, 2002.

[13] L. Freitag, P. Knupp, T. Munson, and S. Shontz. A comparison of optimization software for mesh shape-quality improvement problems. In *Proceedings of the Eleventh International Meshing Roundtable*, pages 29–40. Sandia National Laboratories, 2002.

[14] 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.

[15] L. Freitag and C. Ollivier-Gooch. A cost/benefit analysis for simplicial mesh improvement techniques as measured by solution efficiency. *International Journal of Computational Geometry and Applications*, 10:361–382, 2000.

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

[17] H. Han and C. Tseng. A comparison of locality transformations for irregular codes. In *Proceedings of the Fifth International Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers*, pages 70–84, Rochester, New York, 2000. Springer-Verlag.

[18] C. T. Kelley. *Solving Nonlinear Equations with Newton's Method*. SIAM, Philadelphia, Pennsylvania, 2003.

[19] 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. *International Journal for Numerical Methods in Engineering*, 48:401–420, 2000.

[20] P. Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities, Part II – A framework for volume mesh optimization and the condition number of the Jacobian matrix. *International Journal for Numerical Methods in Engineering*, 48:1165–1185, 2000.

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

[22] T. S. Munson. Mesh shape-quality optimization using the inverse mean-ratio metric. Preprint ANL/MCS-P1136-0304, Argonne National Laboratory, Argonne, Illinois, 2004.

[23] T. S. Munson. Mesh shape-quality optimization using the inverse mean-ratio metric: Tetrahedral proofs. Technical Memorandum ANL/MCS-TM-275, Argonne National Laboratory, Argonne, Illinois, 2004.

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

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

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

[27] M. Shephard and M. Georges. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering*, 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, Pennsylvania, May 1996. ACM.

[29] J. Shewchuk. What is a good linear element? Interpolation, conditioning, and quality measures. In *Proceedings of the Eleventh International Meshing Roundtable*, pages 115–126. Sandia National Laboratories, 2002.

[30] L. N. Trefethan. *Spectral Element Methods in MATLAB*. SIAM, Philadelphia, Pennsylvania, 2000.

[31] R. J. Vanderbei. LOQO user's manual – Version 4.05. Technical report, Princeton University, Princeton, New Jersey, 2000.

[32] R. J. Vanderbei and D. F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.

[33] R. Waltz and J. Nocedal. KNITRO user's manual – Version 3.1. Technical Report 5, Northwestern University, Evanston, Illinois, 2003.

[34] L. Zhang, S. Balachandar, P. Fischer, and T. Munson. Private communication, December 2004.

The submitted manuscript has been created by the University of Chicago as Operator of Argonne National Laboratory ("Argonne") under Contract No. W-31-109-ENG-38 with the U.S. Department of Energy. The U.S. Government retains for itself, and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.