44th AIAA Aerospace Sciences Meeting and Exhibit
9 - 12 January 2006, Reno, Nevada

AIAA 2006-532

Grid Control of Viscous Unstructured Meshes Using Optimization

Mandar S. Sahasrabudhe *, Steve L. Karman Jr. †and W. Kyle Anderson‡
University of Tennessee, Chattanooga, Tennessee , 37403

A mesh smoothing approach is described that uses an optimization technique to improve the quality of unstructured viscous meshes. The cost function for the optimization is based on element condition number. The cost function prohibits element inversion and attempts to imposed a desired spacing and growth rate of elements in viscous layers. The method can be run using global or local optimization. If local optimization is used a cutoff threshold can be employed to restrict the optimization to the nodes with the highest cost function, reducing the run times considerably. Several examples of optimization of three-dimensional viscous meshes are included, with improvements in several quality metrics noted.

Nomenclature

α	Step Size Parameter
Δd	Viscous Spacing for first layer
C_{dev}	Standard deviation multiplier
$cn^{\star}$	Augmented Condition Number
f	Geometric Progression Factor
g	Geometric Growth Rate
h_n	Desired Normal Spacing
h_t	Existing Tangential Spacing
$Mean$	Mean
Std	Standard deviation
$Thresh$	Threshold Value
x	Physical Coordinate
y	Physical Coordinate
z	Physical Coordinate

I. Introduction

MEsh smoothing is a widely used technique to improve the quality of unstructured meshes . The mesh improvement techniques that have been utilized have varied from Linear Spring based Smoothing , Torsional Spring based Smoothing[1,2] , to using smoothing based on solving partial differential equations (Winslow smoothing)[3] and finally traditional optimization based smoothing.[4,5]

The optimization based smoothing procedures used include modified steepest descent algorithm[4] , conjugate gradient based optimization [5] and the use of linear programming. Within mesh optimization a wide variety of grid metrics have been used constructing the cost functions for optimizing the meshes. In 2D Angle , Aspect Ratio, Jacobian, Condition Number, Mean Ratio, Radius Ratio[1,6] have been utilized while Solid Angle, Aspect Ratio, Jacobian, Condition Number, Mean Ratio have been used successfully in 3D[7] . All these grid metrics have been used with some success for improving the quality of unstructured meshes, but the Jacobian, Aspect Ratio and Condition Number have been shown to be more robust for improving

*Research Associate and Graduate Student, Graduate School of Computational Engineering, and AIAA Student Member
†Research Professor, Graduate School of Computational Engineering, and AIAA Associate Fellow.
‡Professor, Graduate School of Computational Engineering, and AIAA Associate Fellow.

the quality of the meshes.[7] Unstructured smoothing methods until now have largely focused on smoothing for inviscid meshes and do not respect viscous clustering. In this paper we outline a process for generating high quality, three-dimensional unstructured viscous meshes using optimization techniques. The optimization technique can be applied to viscous and inviscid meshes that contain tetrahedral elements, as well as pyramid, prisms and hexahedral elements. The initial meshes can originate from many sources, including commercial packages like Gridgen[8] or proprietary grid generation packages such as described in Ref. 9, 10 and 11. The Gridgen mesh generator uses a modified Delaunay method to generate unstructured grids consisting of triangles and tetrahedra but can also generate hybrid meshes that include prism, pyramid and hexahedral elements. The unstructured mesh generation method described in Ref. 9 used a hierarchial tree structure and recursive cell subdivision to discretize regions around complicated geometries and used a volume-to-surface approach to generate the final body conforming mesh.

In Ref. 10 and 11 viscous meshes were created by inserting layers into an existing inviscid mesh using linear elastic smoothing. The insertion process attempts to insert layers with a specified initial spacing and growth rate and to have a smooth blending of the spacing at the top layer with the inviscid portion of the mesh. This paper describes an optimization technique that can be used to improve the quality of viscous meshes from any of these sources and ensure the desired spacing and smoothness is achieved.

II. Cost Function

The cost function is the most critical aspect to this mesh optimization technique. It must be defined to ensure that high quality elements are obtained and it must also enable enforcement of desired grid spacing in viscous meshes. Several baseline functions were considered during the development of the cost function, such as element volume, corner Jacobians, element aspect ratio and condition number. Mathematical definitions for these are provided in the Appendix. Element volume and corner Jacobians have some influence over whether positive element volumes are produced. Aspect ratio and condition number ensure good element shapes are produced, but cannot enforce positive volumes. The cost function that was developed combines the good shape enforcing nature of aspect ratio or condition number with the positivity enforcing nature of the corner Jacobians. Either aspect ratio or condition number can be used as the baseline function. Then the baseline function is combined with the corner Jacobian to produce a cost function that favors good element shape and positive volume.

The derived cost function is computed at a corner of an element using the four nodes and three edges common to the corner. Assuming condition number has been chosen for the baseline function, the formula for the cost function at a corner is given in equation 1.

$$cost = (\frac{-1}{cn^\star} + 1)^2 \tag{1}$$

where $cn^\star$ is the augmented condition number of the tetrahedral element formed at the corner. If the value of the Jacobian at the corner is negative then the augmented condition number is negative. The possible range of values for this cost function, computed at a corner, is 0 to 4. A value less than one indicates the tetrahedron has a positive volume. A value greater than one has a negative Jacobian, which means the corner is inverted. A value of zero is an ideal element with a condition number of 1. This form of cost function incorporates the inversion of the corner into the definition and driving this value to a minimum will correct an inverted element.

This cost function is computed at a node level or at a global level using a linear combination of the average (cavg) and maximum (cmax) of the corner cost functions. When a local cost function is computed for a node the corners that contribute are shown as arc segments in Figure 1. The opposite corners of the quadrilateral elements do not contribute to the cost function. A similar situation occurs in three dimensions. When a global cost function is computed all corners of the mesh contribute.

A linear combination is obtained by using a blending function that is based on the maximum value of the corner cost function. The blending function is shown in Figure 2 and the formula is provided below in equation 2.

$$blend = \frac{\tanh((cmax - 1)10) + 1}{2} \tag{2}$$

This blending function is used in equation 3 to produce the blended cost function.

$$cost = blend \cdot cmax + (1.0 - blend) \cdot cavg \tag{3}$$

The purpose for the blending is to provide a single cost function that works well in all regions of the mesh. A cost function based purely on the average value of the corner cost function tends to work well in most regions but does not prohibit a local value to exceed 1, which indicates an inverted tetrahedron. A cost function based purely on the maximum value of the corner cost functions ensures all positive volume elements but does not produce a smooth mesh. The blended cost function favors the average of the corner cost functions when the maximum cost function is less than 0.9 and favors the maximum cost function for values above 1.1. The transition is deliberately steep at a value of 1.0 to influence the worst corner to improve, but still has some smoothness due to the contribution of the average value. The combination of the derived cost function and the blending function results in a single cost function that incorporates a barrier to inverted elements and requires no additional constraints. The optimizer using this blended cost function should be able to untangle meshes that contain some inverted or skewed elements.

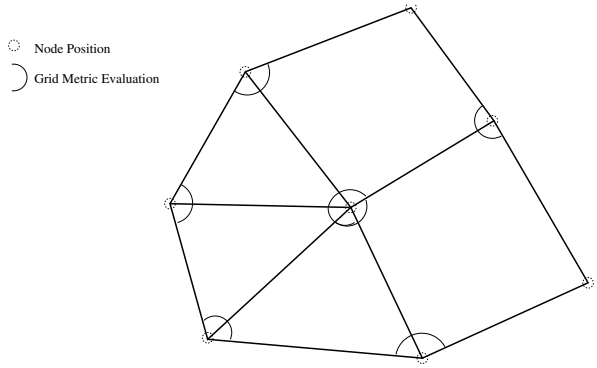


Figure 1. Grid Metric Evaluations for a hybrid mesh

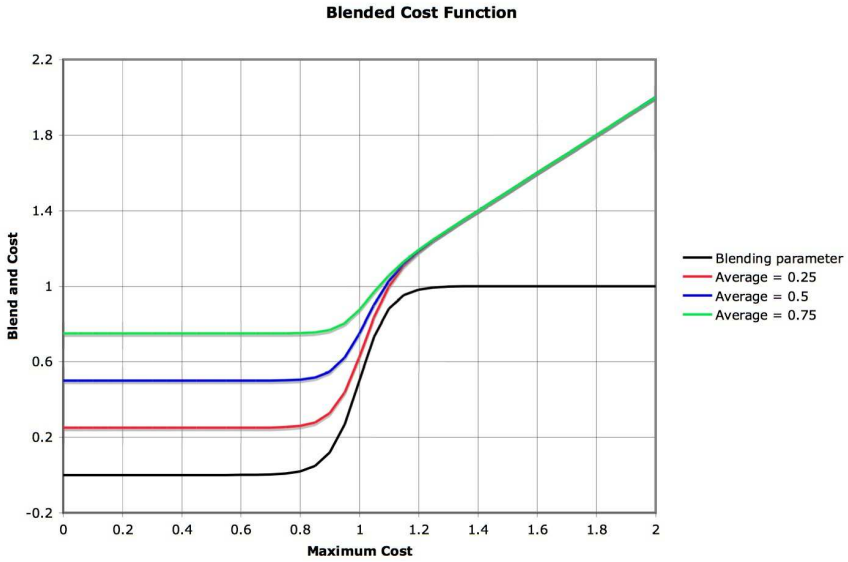


Figure 2. Blending Function for the cost function

III. Optimization

The optimization algorithm is similar to the approach outlined in Ref. 4 and 5. Either global or local optimization is possible. Local optimization attempts to improve the individual cost function at each node. Global optimization attempts to improve a global cost function and works on all nodes simultaneously. Details of each approach are presented below.

A. Local optimization

The local optimization algorithm operates on nodes one at a time and attempts to reduce the cost function at the node level. This requires the current value of the cost function, a search direction, and a local step size marching parameter, α. The definition of the cost function has been described earlier. The local search direction is determined by computing the gradient vector of the cost function using central differences. The perturbation distance used to compute the central differences is computed based on the minimum edge length emanating from the node. Central differences are used to compute the X, Y and Z components of the derivative. A steepest decent search direction is then determined by reversing the direction of the gradient vector. The local step size parameter, α, is initialized by dividing a fraction (10%) of the local minimum edge length by the magnitude of the gradient vector. This parameter is continually adjusted during the optimization sub-iterations, so the initial value is important, but not critical.

The local optimization proceeds by cycling through each of the nodes and perturbing the node in the search direction according to the current step size parameter α . A new value of the local cost function is computed. If the new local cost function is improved the new location is accepted. If the new local cost function is worse the new location is rejected and the node is reset to the original position. The local step size parameter is increased by 25% if the local cost function improves and decreased by 50% if the cost function worsens.

Local optimization offers an opportunity to accelerate the process by restricting the calculations to the nodes with the worst values of the cost function. This is implemented using the average and standard deviation of the local cost functions. A threshold value is set equal to the mean plus a multiple of the standard deviation.

$$Threshold = Mean + Std \cdot C_{dev} \tag{4}$$

If the standard deviation multiplier,C_{dev}, is set to zero then all nodes with a cost function above the mean will be processed. Positive values of C_{dev} will set the threshold above the mean and negative values of C_{dev} will set the threshold below the mean. Large positive values of C_{dev} set the threshold at a level where only the nodes with the highest cost function are processed. Large negative values of C_{dev} will set the threshold below zero and turn on all the nodes for processing. The statistical analysis is performed each iteration, so the threshold value is adjusted continually. Implicit in this approach is the assumption that working on the nodes with the worst values of the cost function will tend to reduce the average and standard deviation. As a result, the overall system will converge to the same final grid, given sufficient number of iterations. A significant portion of the cost savings is achieved by reducing the computation of the search direction to the nodes above the threshold.

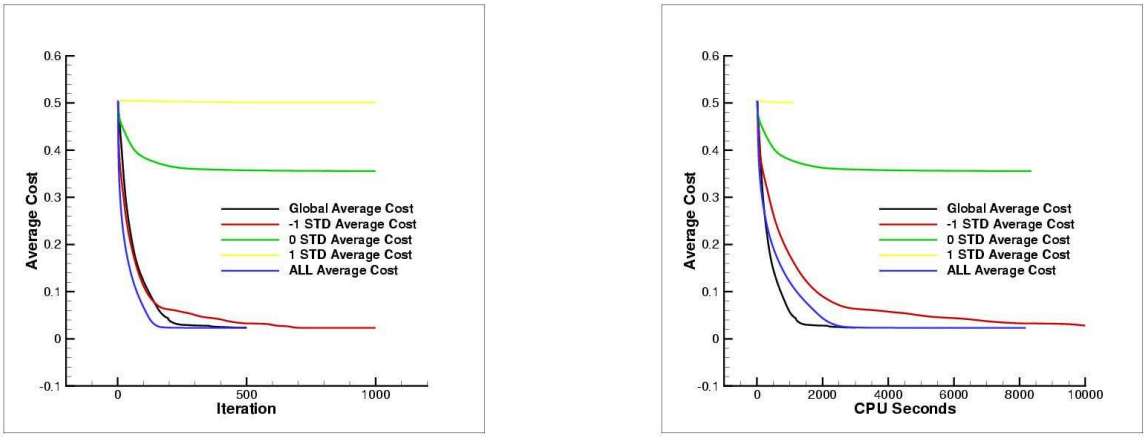
B. Global Optimization

The global optimization works on all nodes simultaneously in an attempt to reduce a global cost function. The local search directions and local step size parameters described in the previous section are used with the global optimization. The update, process however, is different. For global optimization all nodes are perturb in the local search direction using the local step size parameter. A new global cost function is computed and compared with the current value of the global cost function. If the new cost function is improved all of the perturbations are accepted. If the new cost function is worse all nodes are reset to the original positions. The local step size parameters are incrementally increased by 25% if the global cost function improves and decreased by 50% if the global cost function worsens.

C. Global versus Local Optimization

A numerical experiment was performed to determine the affect on convergence of different values of the standard deviation multiplier, C_{dev}. The configuration for this experiment is similar to the first case shown in the results section. The case was computed using global optimization and local optimization with 4 values of the standard deviation multiplier; 1, 0, -1, -10 (All nodes). All cases achieved convergence of the average cost function and contained all positive volumes and Jacobians. The cost-convergence plots versus iteration are shown in Figure 3(a). The global optimization case and local optimization cases with standard deviation multiplier of -1 and -10 achieved the same level of convergence. The other two local optimization cases did not converge to the same final mesh. The original mesh is shown in Figure 4(a). The final meshes are plotted in Figure 4(b)-(f). The case with the standard deviation multiplier set to 1.0 did not substantially alter the original mesh. The case with the standard deviation multiplier set to 0.0 converged to an unreasonable solution.

The local optimization case that worked on all nodes converged in the least iterations. The global optimization case converged in less iteration than all of the other local cases. Figure 3(b), indicates that the global optimization converges in less CPU time than all other cases for this experiment. One conclusion from this experiment is that global optimization tends to be more efficient than local optimization. Local optimization can converge to the same final mesh if the optimizer is allowed to work on enough nodes. It is believed the two cases that converged to different solutions did so because the threshold was too high and the optimizer could not alter enough nodes to impact the cost function statistical analysis. The nodes that were allowed to move could not be moved enough to significantly lower the mean and standard deviation. This does not invalidate the use of the local optimization, though. If a mesh has only a few nodes that have a high local cost function the threshold can be set to a value that restricts the optimizer to work on only those nodes and require significantly less computer time.



(a) Iteration Cost

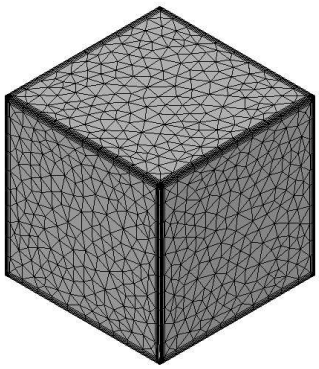
(b) Cpu Time Cost

Figure 3. Comparison of Cost for Local and Global update schemes on a Cube Geometry

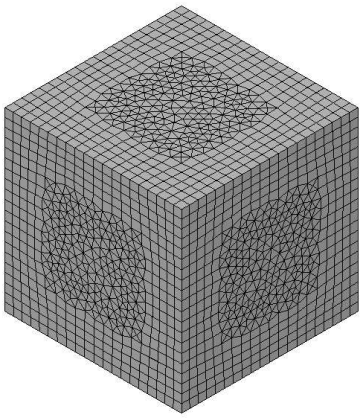
IV. Viscous Grid Optimization

Optimization of viscous meshes adds a complexity to the mesh-smoothing problem. The cost function, as currently defined, will not respect the tightly packed elements in the boundary layer region. If the optimizer with the current cost function is applied to a viscous mesh the result will be similar to the fully converged meshes shown in the numerical experiment in the previous section. In order to control the spacing of the viscous layers the cost function must be modified to enforce the desired normal spacing at each layer of the mesh. This is achieved by scaling the edges of the tetrahedron used to compute the cost function at a corner. This requires several pieces of information; identification of boundaries where viscous layers exist, the number of viscous layer present, and the desired normal spacing distribution through the layers.

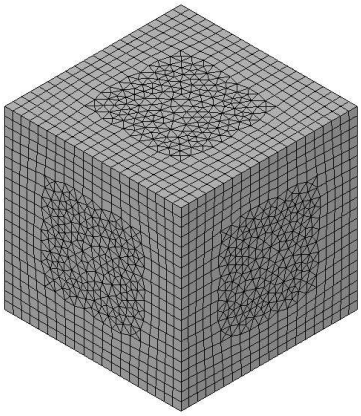
The identification of viscous boundaries and normal distribution is specified in the same fashion as



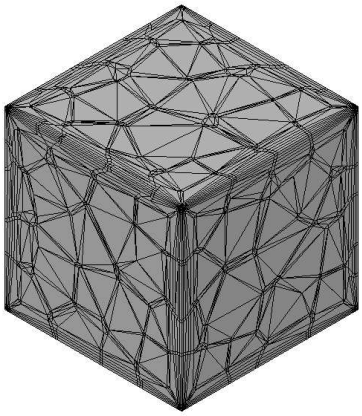
(a) Original Viscous Mesh



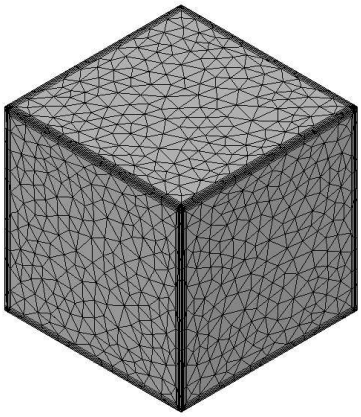
(b) Locally Optimized with all nodes



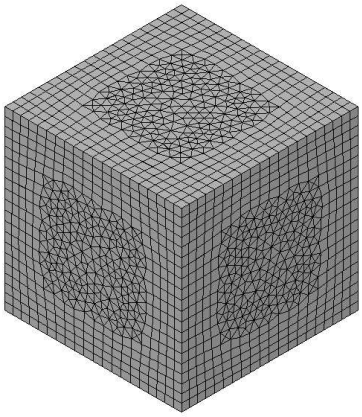
(c) Locally Optimized with standard deviation set to -1



(d) Locally Optimized with standard deviation set to 0



(e) Locally Optimized with standard deviation set to 1



(f) Globally Optimized

Figure 4. Comparison of Local and Global update schemes on a Cube Geometry

described in Ref. 11. The initial spacing Δd, geometric progression factor, f, and geometric growth rate, g, are specified by the user. The geometric progression factor is used to compute the spacing of one layer based on the spacing of the previous layer using the formulae below. The geometric progression factor can be constant or can be changed from layer to layer using the geometric growth rate, g.

$$\begin{aligned} \Delta d^i &= \Delta d^{i-1} f^i \\ f^i &= f^{i-1} g \\ for\ i &= 2, no.of layers \end{aligned} \tag{5}$$

In viscous layers of the mesh a scaling of the edges of the corner tetrahedron in the normal direction is performed to enforce the desired normal spacing. The scaling is based on the desired normal spacing for the layers. In two dimensions, for a quadrilateral viscous element, the scaling is computed as the tangential spacing divided by the desired normal spacing. The current tangential spacing is defined as h_t in equation 6 and the desired normal spacing is defined as h_n. S_n is used to scale the edges of the element in the normal direction before the cost function is computed. Since the tangential edge lengths is not altered S_t is set to 1.

$$\begin{aligned} S_n &= \frac{h_t}{h_n} \\ S_t &= 1.0 \end{aligned} \tag{6}$$

The effect of this scaling is to force the optimizer to produce an ideal element using the scaled edge lengths while the un-scaled element will tend to have the desired normal spacing. A similar scaling is performed in three dimensions for prismatic elements and hexahedral elements. The viscous scaling is only incorporated in quadrilateral elements in two dimensions and prismatic and hexahedral elements in three dimensions. For quadrilateral and hexahedral elements the scaling can be imposed in any combination of principle directions of the element. For prismatic elements, only the edges in the direction from one triangular face to the other triangular face can be scaled.

V. Results

The optimization algorithm has been used to generate viscous grids for a cube geometry, an ONERA M6 wing geometry and multi-element wing. All of the examples in this paper are three-dimensional geometries and illustrate the degree of control provided by the optimization algorithm for smoothing viscous grids.

A. Cube Geometry

The first case is a simple cube-shaped geometry, but it demonstrates the versatility of the mesh optimization technique presented in this paper. The inviscid mesh is shown in Figure 5(a). Viscous layers are inserted on all walls using the method presented in Ref. 11. The initial viscous mesh has 5 layers inserted with an initial spacing of 0.01 and a geometric progression factor of 1.2. The initial viscous mesh is shown in Figure 5(b). If pure optimization is performed, without maintaining the viscous clustering, the resulting mesh is shown in Figure 5(c).

If the viscous scaling of the cost function is imposed using the desired initial spacing and geometric progression factor the optimizer produces the mesh shown in Figure 5(d). This is different from the initial viscous mesh for several reasons. One reason is the initial inviscid mesh, produced by Gridgen, is generated using a Delaunay method that does not result in an optimized mesh according to the cost function used by the optimizer. The Delaunay method attempts to produce a tetrahedral mesh, which maximizes the minimum angles in the mesh. The interior mesh points are most likely generated at the circumcenters of high aspect ratio tetrahedra, so their placement does not guarantee an optimal mesh according to the optimizer cost function based on element condition number. And the insertion of the viscous layers did not significantly alter the inviscid portion of the mesh. So the optimizer had some work to do. A second reason is the surface points are not optimally positioned, either. In fact, the edges of the domains are not even equally spaced. The optimizer corrected this situation; the tangential spacing away from the corners is now more equally spaced. And thirdly, the optimizer attempts to improve the cost function at all mesh points. Therefore, the nodes are moved in an attempt to minimize the cost function at each point. As a result, some points, such as the points on the edge of the viscous mesh are pulled away from the surface, sacrificing the desired viscous spacing of the top layer.

Several quality metrics were computed before and after each optimization. Figure 6 shows the minimum, average and maximum values of the element condition number for each element type in the mesh. This case contained tetrahedra, prisms and hexahedra, so the pyramid portion of the figure can be ignored. The original inviscid, tetrahedral mesh, prior to inserting the viscous layers, is colored blue. The original viscous mesh is colored red. There are considerably high element condition numbers for the prisms and hexahedra in the original viscous mesh. The mesh that was optimized without the viscous scaling (optimized) is colored green. All the statistics for the prisms and hexahedra show dramatic improvements over the original viscous mesh, but the viscous clustering was sacrificed. The mesh that was optimized with the viscous scaling (optimized-viscous) is colored purple. There is significant improvement over the original viscous mesh, especially for the maximum values for the prisms and hexahedra. However, the values for the optimized-viscous mesh did not reduce as far as the optimized mesh. These metrics do not account for the viscous scaling used in the optimizer, so the values for a clustered mesh should show values greater than one for the condition number.

A similar plot for element aspect ratio is shown in Figure 7. Again, these values do not reflect the viscous scaling used by the optimizer, but the improvement of the optimized and optimized-viscous is similar to the condition number plot. The definition of aspect ratio for the various element types is provided in the appendix.

The next quality metric plot, shown in Figure 8(e), is for the maximum solid angle for each element. Solid angle, defined in the appendix, is a measure of the surface area on a unit sphere resulting from the projection of the edges emanating from the corner of the element. The total surface area of a unit sphere is 4 pi, so any angle approaching 2 pi is considered extremely high and could lead to inaccuracies in the flowfied solution. This case does not have any extremely large solid angles, but the optimizer still managed to improve the values over the original viscous mesh.

One more optimization case was computed for this geometry. The desired initial spacing values were modified for three of the boundaries. The normal spacing value for the solid surface was reduced to 0.0025. The normal spacing value for the symmetry walls (top and bottom in the figure) was reduced to 0.005. And the spacing for the farfield boundary was doubled to 0.002. The optimizer produced the mesh shown in Figure 5. The statistical minimum, average and maximum normal spacing for all boundaries and all cases

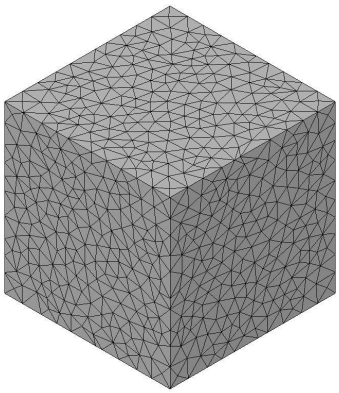
is shown in Figures 9, 10, 11. The desired normal spacing of the original viscous mesh is achieved, as shown in red in the figures, but the mesh has not been "optimized". The optimized mesh without viscous spacing control loses the desired values, as shown in green. The optimized-viscous mesh mostly recovers the desired initial normal spacing, shown in purple. The reasons for the differences were explained earlier. The yellow bars show that the optimizer was able to come very close to achieving the new normal spacing values. This demonstrates the use of the technique to optimize a mesh with different normal spacing values than those used to generate the mesh.

B. ONERA M6 Wing

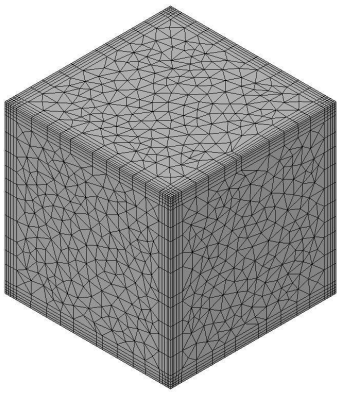
The second case was a commonly used validation case, Onera M6 wing[12] . Viscous layers were inserted on all wing surfaces and allowed to march up the symmetry plane using the method presented in Ref. 11. The viscous mesh shown in Figure 12 had 19 layers inserted with an initial spacing of 0.0001 and a geometric progression factor of 1.15 and a growth rate of 1.01. The total number of points was 701,461. There were 1,011,739 tetrahedra and 1,028,166 prisms. The optimizer used the same specified initial spacing, geometric progression factor and geometric growth rate. The optimizer ran on a Linux workstation and took approximately 263 seconds per iteration. The convergence of the minimum, average and maximum values of the cost function are shown in Figure 13.

Before and after images of the symmetry plane mesh are shown in Figure 14. The meshes appear to be very similar from this perspective. However, magnified views of the leading edge region in Figure 15 show the changes resulting from the optimization. The initial viscous mesh does not extend as far from the surface as the optimized mesh. The optimizer was able to extend the viscous layers consistently at all section of the airfoil. The method used to generate the original viscous mesh imposed restrictions on the penetration of the layers based on the local element size. So smaller elements near the leading edge resulted in more tightly packed viscous layers. An examination of the before and after images of the wing tip region in Figure 16 shows a similar result. The original mesh was prevented from extending further away from the surface because of the linear-elastic equations used to generate the layers. With the viscous scaling, the optimizer attempts to impose the specified spacing and growth of the layers. It also produces a nearly orthogonal mesh at the surface. In fact, the cost function attempts to produce orthogonal corners at all elements in the mesh. The cost function is based on the condition number and the ideal condition number is achieved when the edges of the corner are the same length and orthogonal.

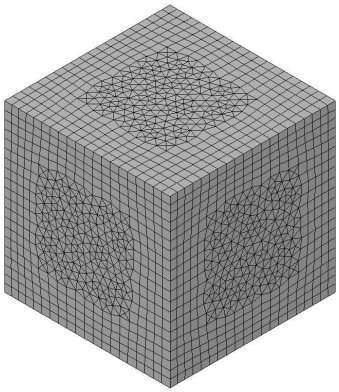
Quality metrics for aspect ratio, condition number and solid angle are shown in Figures 17, 18, 19, The mesh only contained tetrahedra and prisms, so the sections for pyramid and hexahedra can be ignored. Both the aspect ratio and condition number showed dramatic improvement for the optimized mesh over the original mesh. It should be noted that these two metrics do not take into account the viscous scaling that was used in the optimizer cost function. The solid angle values showed improvement for the tetrahedra and a slight degradation for the maximum hexahedron value. Figure 20 shows that the optimizer increased the average values of the normal spacing on all wing surfaces and came close to achieving the desired initial normal spacing of 0.0001.



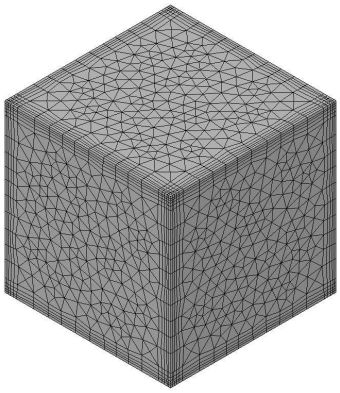
(a) Original Inviscid Mesh



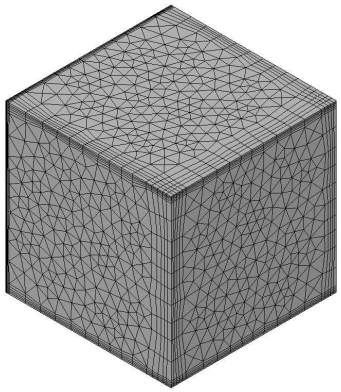
(b) Original Viscous Mesh



(c) Optimized Mesh without Clustering



(d) Optimized Viscous Mesh



(e) Optimized Viscous Mesh with irregular spacing

Figure 5. Original and Optimized Meshes for a Cube Geometry

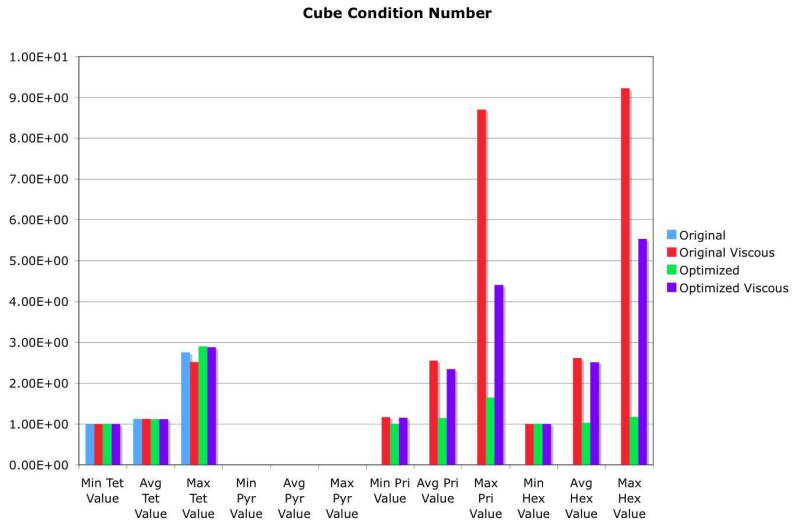


Figure 6. Condition Number Statistics for for Cube Geometry

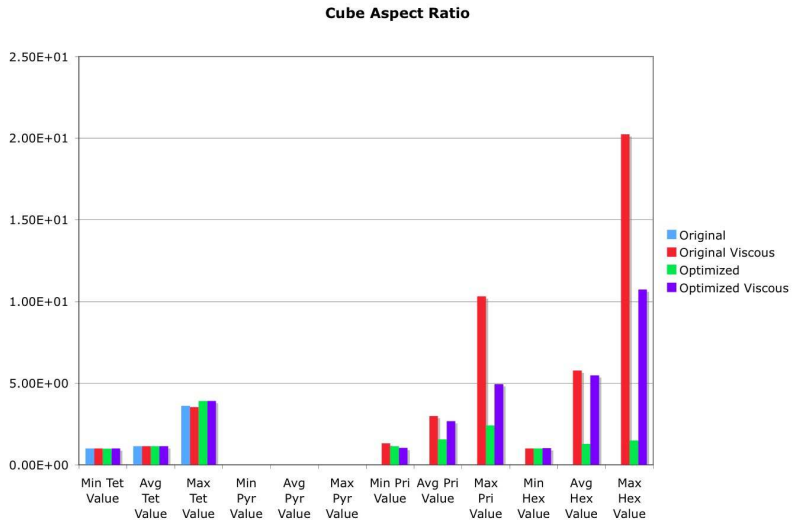
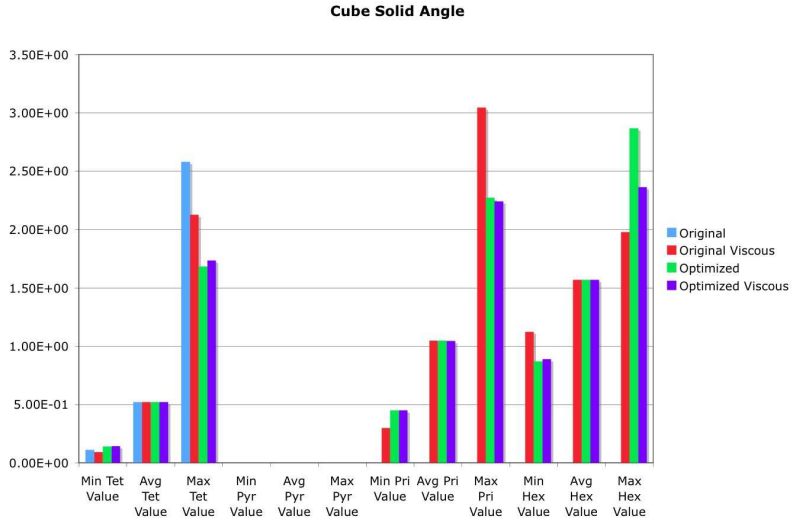


Figure 7. Aspect Ratio Statistics for Cube Geometry



(a) Solid Angle

Figure 8. Solid Angle for Cube Geometry

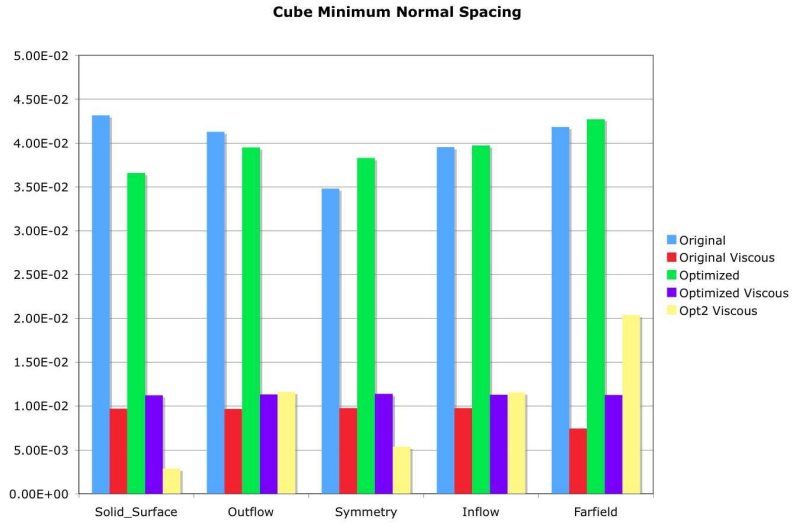


Figure 9. Minimum Normal Spacing for Cube Geometry

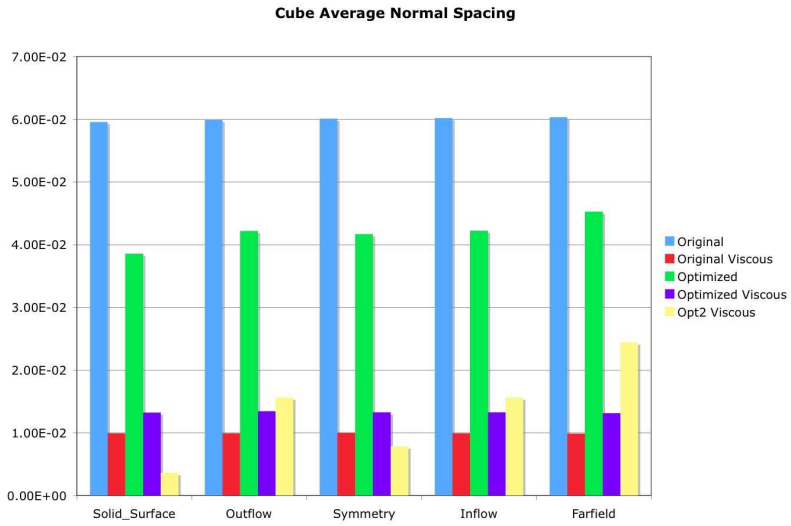


Figure 10. Average Normal Spacing Statistics for Cube Geometry

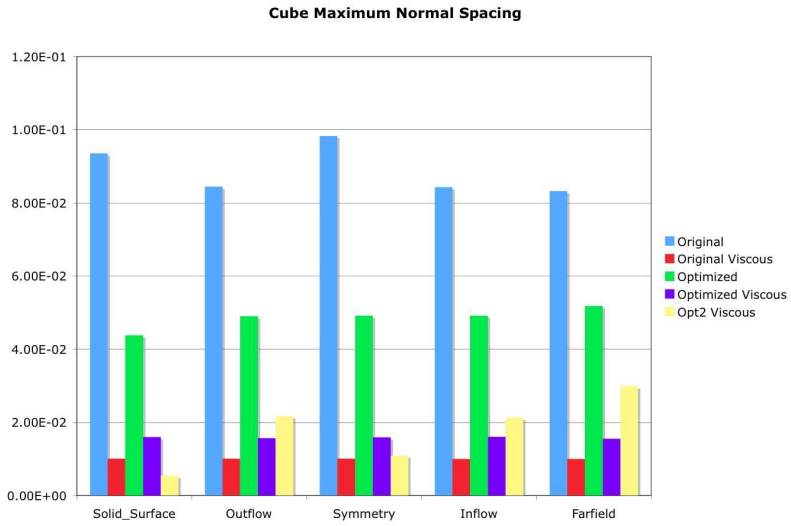


Figure 11. Maximum Normal Spacing for Cube Geometry

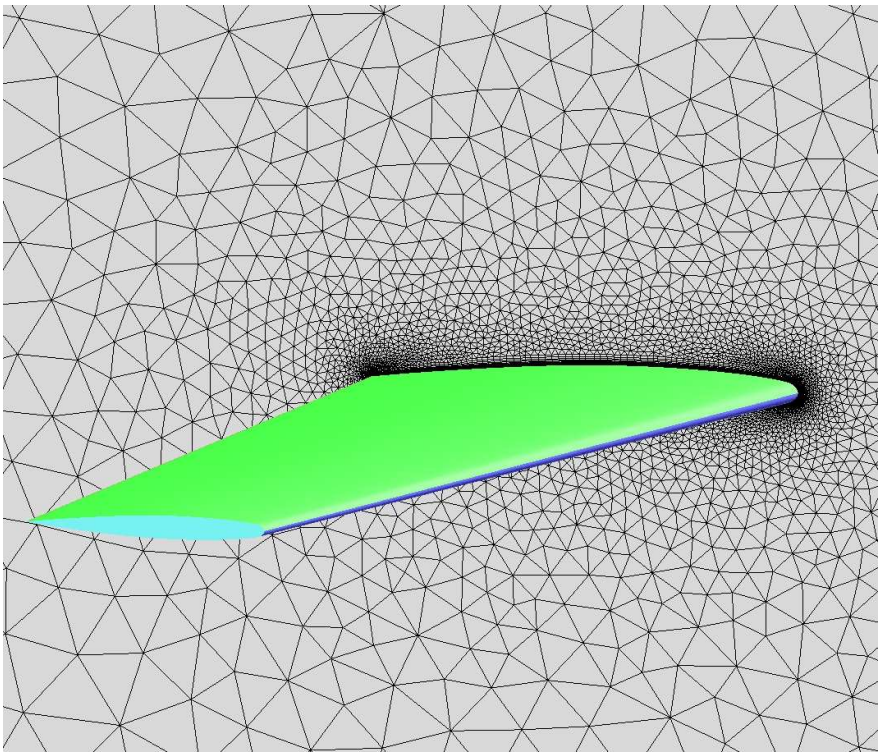


Figure 12. Original Viscous Mesh for an ONERA M6 Wing Geometry

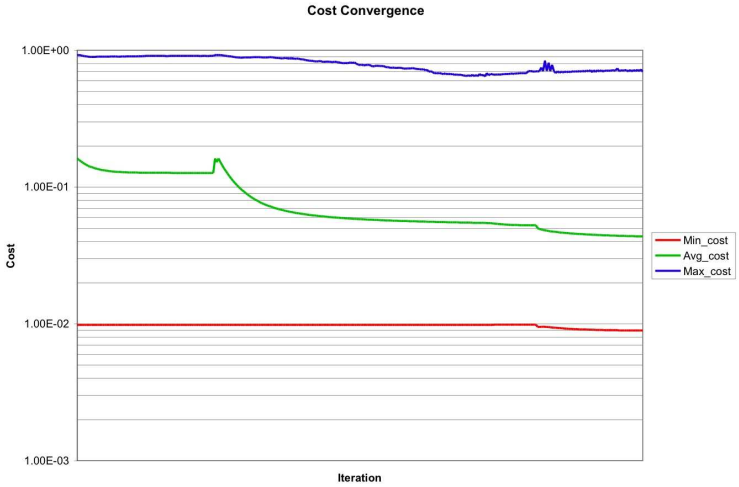
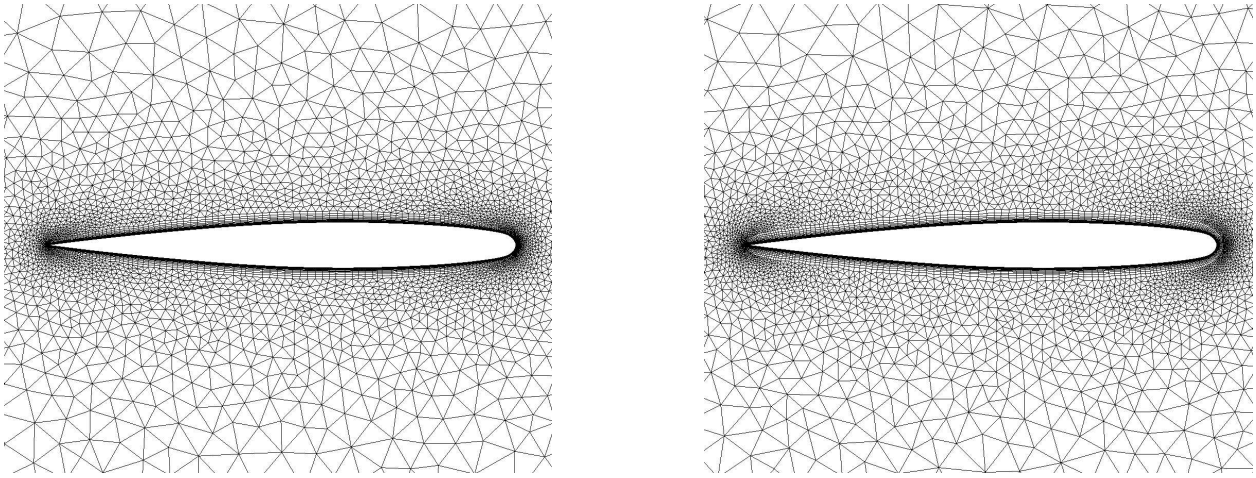


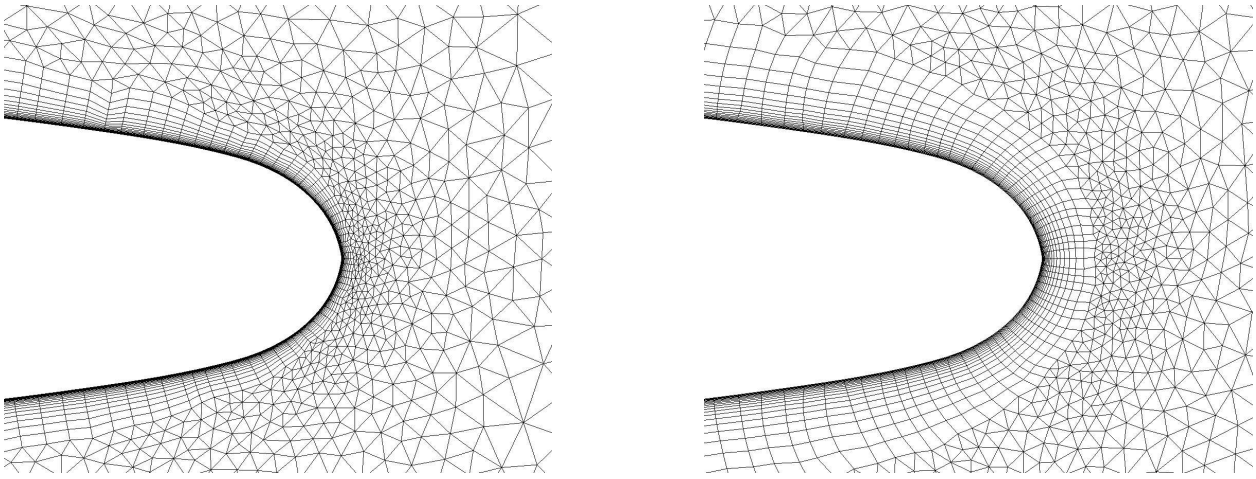
Figure 13. Convergence of cost function and average value of the gradient magnitude



(a) Original Viscous Mesh

(b) Optimized Viscous Mesh

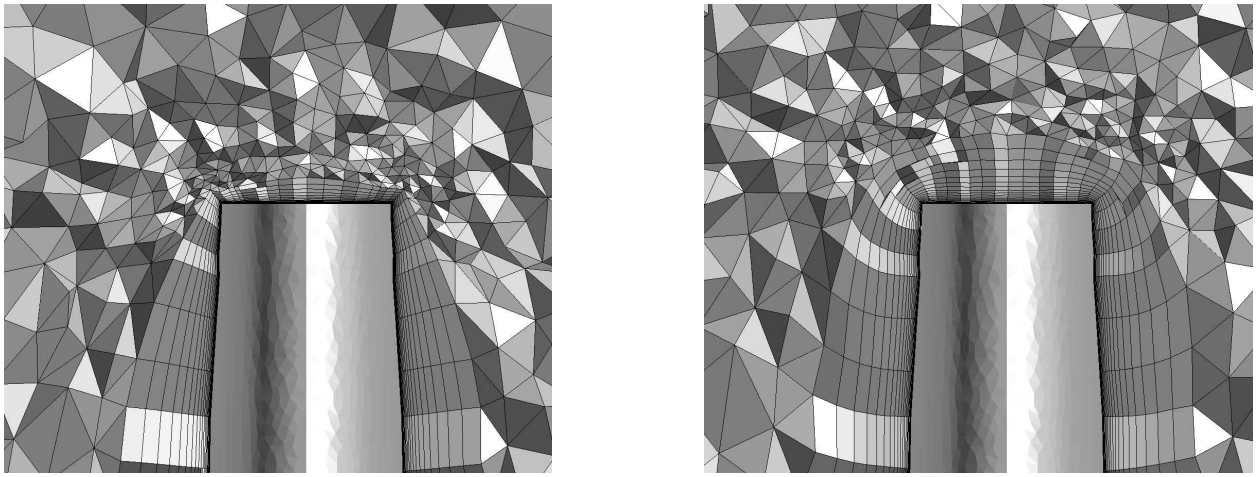
Figure 14. Original and Optimized Meshes for an ONERA M6 Wing Geometry



(a) Original Viscous Mesh

(b) Optimized Viscous Mesh

Figure 15. Original and Optimized Meshes for an ONERA M6 Wing Geometry



(a) Original Viscous Mesh

(b) Optimized Viscous Mesh

Figure 16. Original and Optimized Meshes for an ONERA M6 Wing Geometry

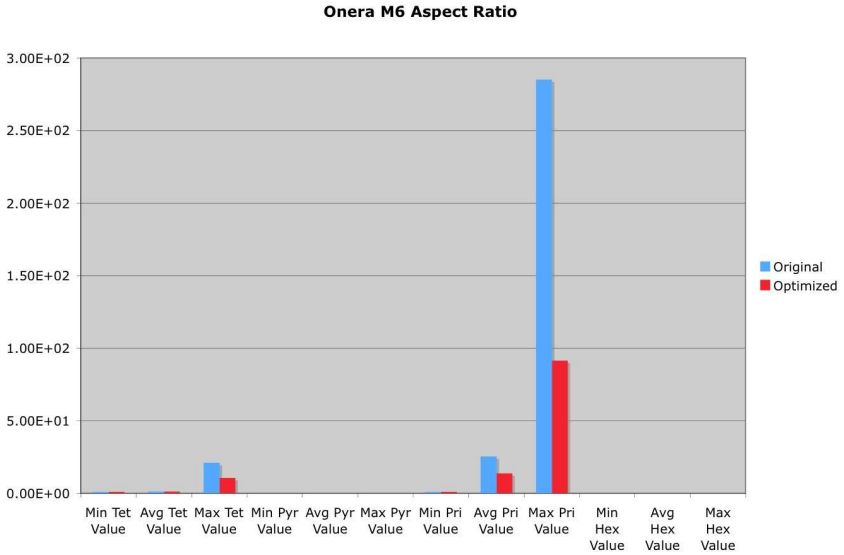


Figure 17. Aspect Ratio for ONERA M6 Wing Geometry

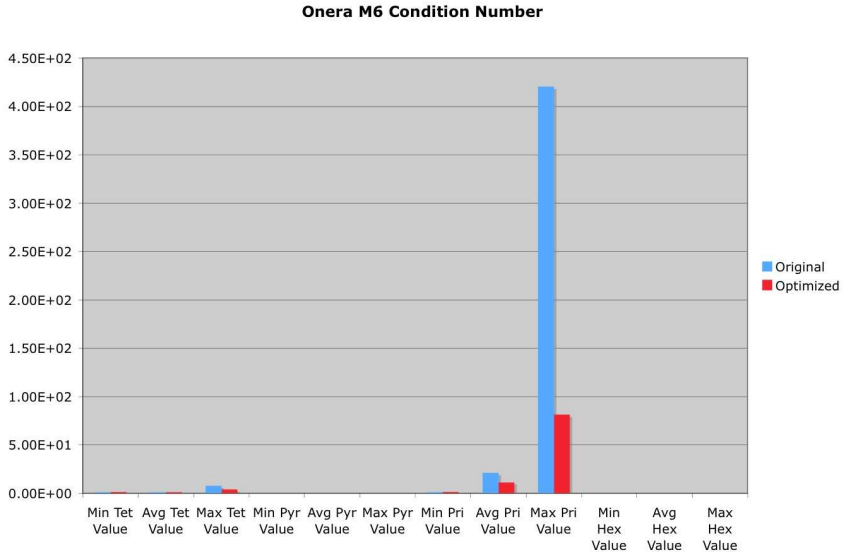


Figure 18. Condition Number for ONERA M6 Wing Geometry

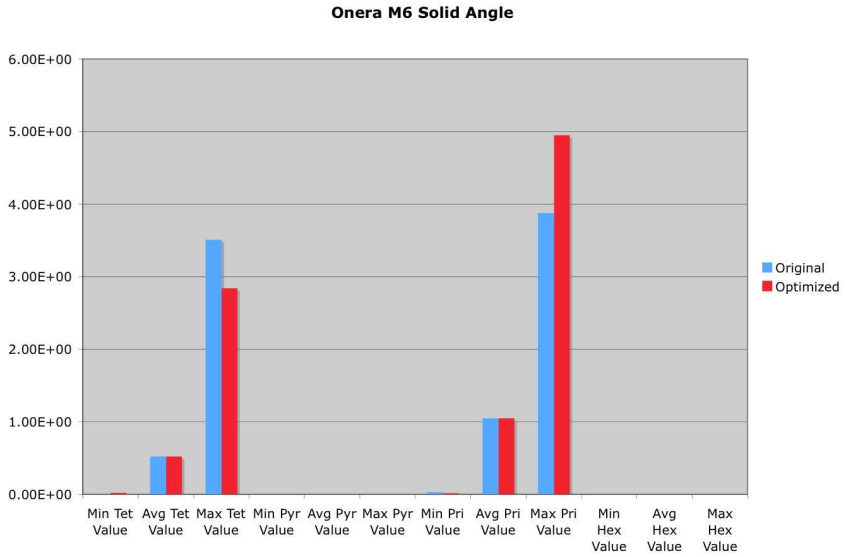


Figure 19. Solid Angle for ONERA M6 Wing Geometry

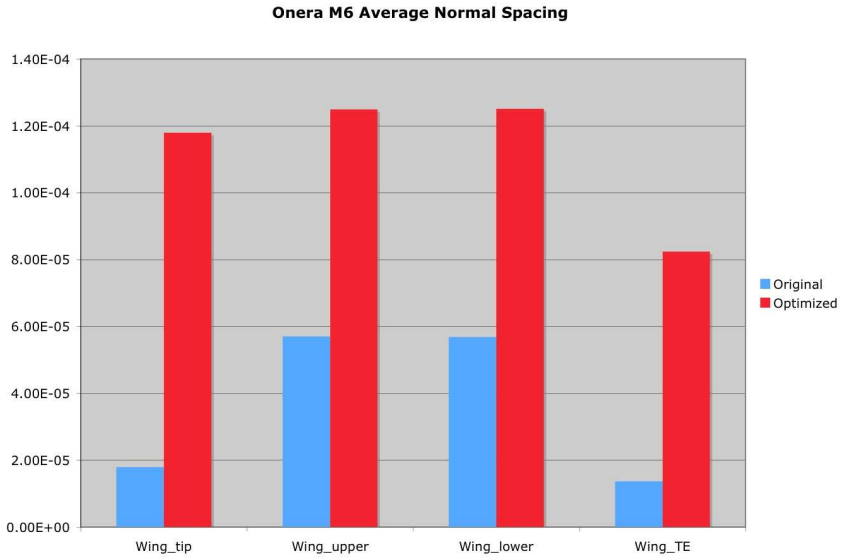


Figure 20. Average Normal Spacing for ONERA M6 Wing Geometry

C. Multi-Element Wing

A more complex geometry was used for the third case, a multi-element, high-lift wing configuration. A two-dimensional, multi-element airfoil with cord length of 1 was extruded and tapered to create the generic wing geometry. Viscous layers were inserted on all wing surfaces and allowed to march up the symmetry plane using the method presented in Ref. 11. The viscous mesh had 20 layers inserted with an initial spacing of 0.0001 and a constant geometric progression factor of 1.15. The total number of points was 1,493,136. There were 1,992,846 tetrahedra and 2,245,160 prisms. The optimizer used the same geometric progression factor, but reduced the initial normal spacing to 0.00005. The optimizer ran on a Linux workstation. The convergence of the average value of the cost function and the average value of the gradient magnitude are shown in Figure 21. The reason for the erratic plot is the optimizer was run in global mode for some iterations and local mode for other iterations. When global optimization was used the optimizer took approximately 474 seconds per iteration. When local optimization was used, the threshold waas set to -1 and the optimizer took approximately 1064 seconds per iteration. The steepest and smoothest convergence was with the local optimization.

Before and after images of wing section meshes on the symmetry plane are shown in Figure 22 through Figure 25.[13] The viscous layers are pulled in closer to the surface in the optimized mesh due to the smaller specified initial spacing. The differences in the average normal spacing for the two meshes are shown in Figure 26. Except for a few trailing edge sections and the slat tip, all average normal spacing values have been reduced. The tetrahedral elements in the gap regions between the main element and the slat and flap are less compressed or distorted in the optimized mesh. A comparison of the meshes at an axial cut through the wing tip of the main element is shown in Figure 27. Again, the viscous layers are pulled in closer to the surface and the mesh is more orthogonal at the surface. The comparison of the Aspect Ratio, Condition Number and Solid Angle quality metrics are shown in Figures 28, 29, 30.

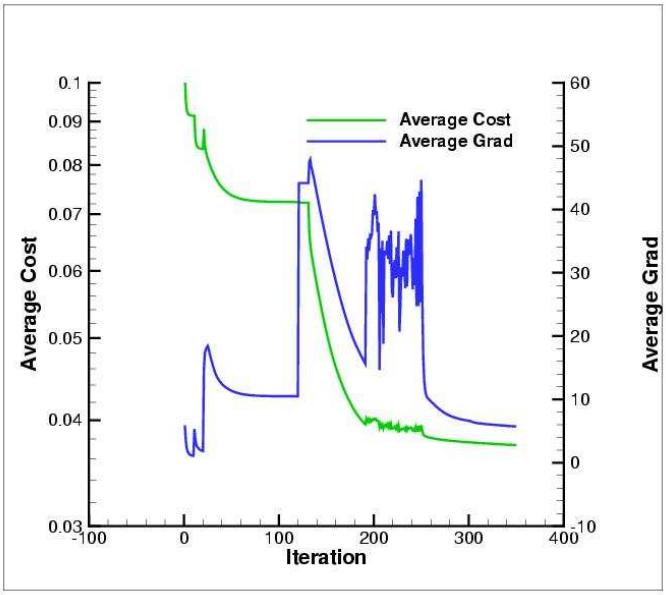


Figure 21. Convergence of cost function and average value of the gradient magnitude

VI. Conclusions

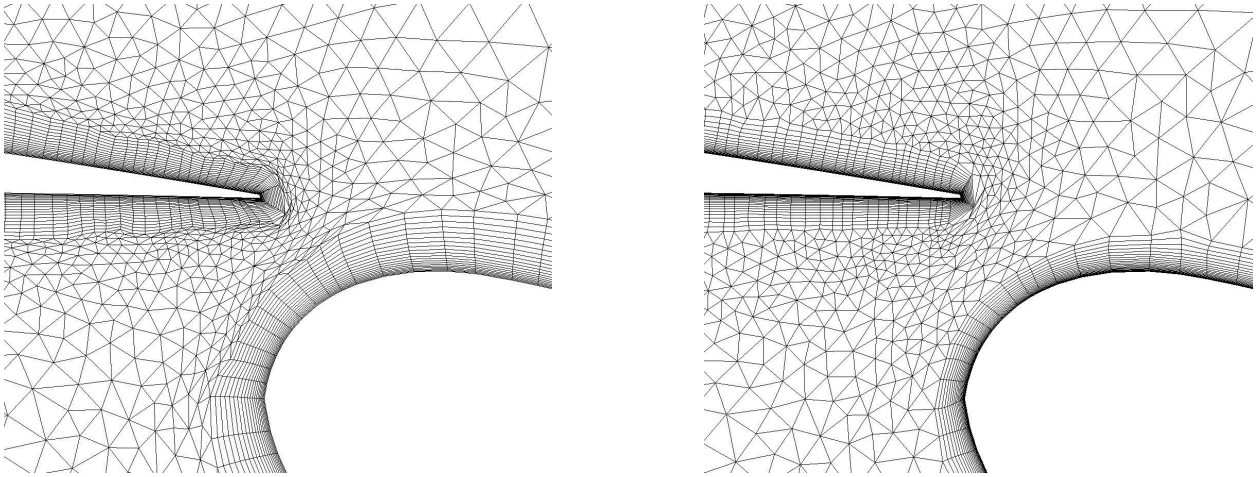
An optimization method has been developed for smoothing unstructured viscous meshes. The method can be applied as a local optimization technique that works on one node at a time or it can be applied as a global optimization, applied to all nodes simultaneously. The cost function of the mesh optimizer employs a combination of element condition number and Jacobian to produce smooth, high quality elements and prevent element inversion. A modification of the cost function allows for edge spacing in the normal direction to be enforced for viscous meshes.

Several three-dimensional example cases were included. The first case demonstrated the control exerted on viscous layers achieved by using the modified cost function. Quality metrics confirmed the optimized mesh was a significant improvement compared to the original viscous mesh. The second case applied this technique to a classic three-dimensional wing configuration. The optimized mesh showed improvements in the quality metrics while attempting to enforce the desired normal spacing. The last case was a generic multi-element wing configuration. The optimizer was used to reduce the normal viscous spacing on all wing surfaces and improve the quality of the elements in the gaps between the main element and the slat and flap sections.

All cases were computed on a LINUX workstation. Approximate run times per iteration were reported for the last two cases. The run times are considered high when compared to previous unstructured mesh smoothing strategies, such as node averaging or spring analogy. However, the increased cost provides a great amount of control over the mesh, including improved quality as measured by several quality metrics and enforcement of viscous grid spacing. Reduced run times per iteration can be achieved by utilizing local optimization and setting a threshold to identify the nodes with the highest cost function. Local optimization with threshold limiting was shown to significantly reduce the time per iteration but must be used wisely to ensure that a sufficient number of nodes are optimized to enable the overall cost function to be reduced. Additional speed-up can be achieved by porting this serial code to a parallel cluster.

VII. Acknowledgments

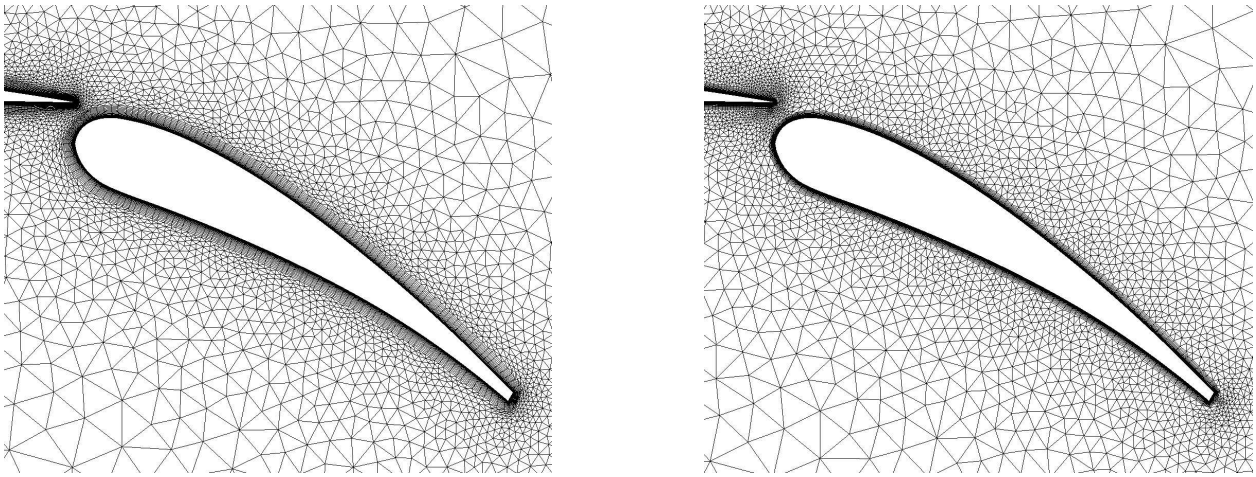
The University of Tennessee at Chattanooga through the Lupton Renaissance Fund sponsored this work. This support is greatly appreciated.



(a) Original Viscous Mesh

(b) Optimized Viscous Mesh

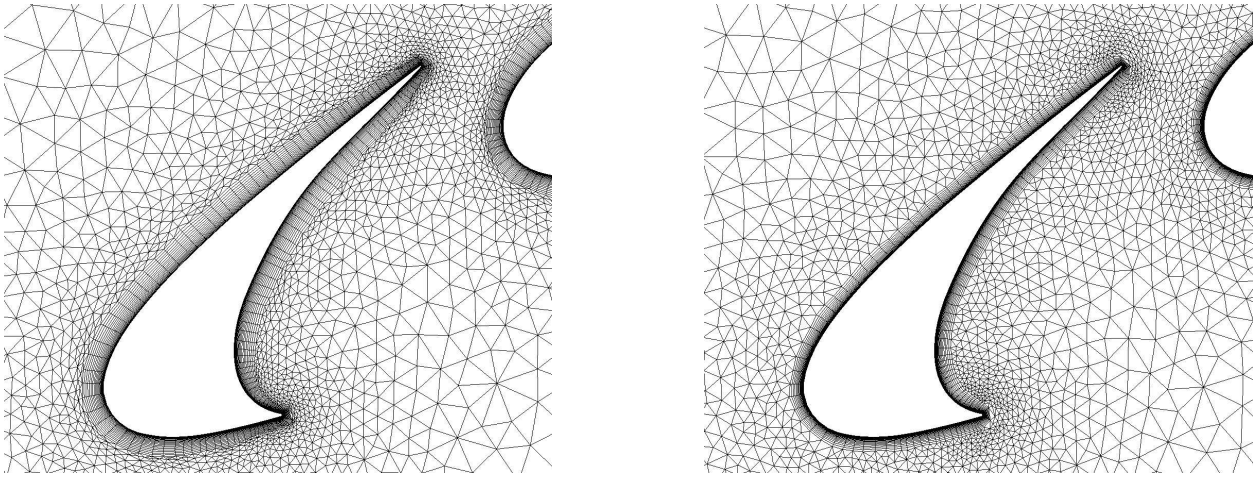
Figure 22. Original and Optimized Meshes for a Multi-Element Wing Geometry



(a) Original Viscous Mesh

(b) Optimized Viscous Mesh

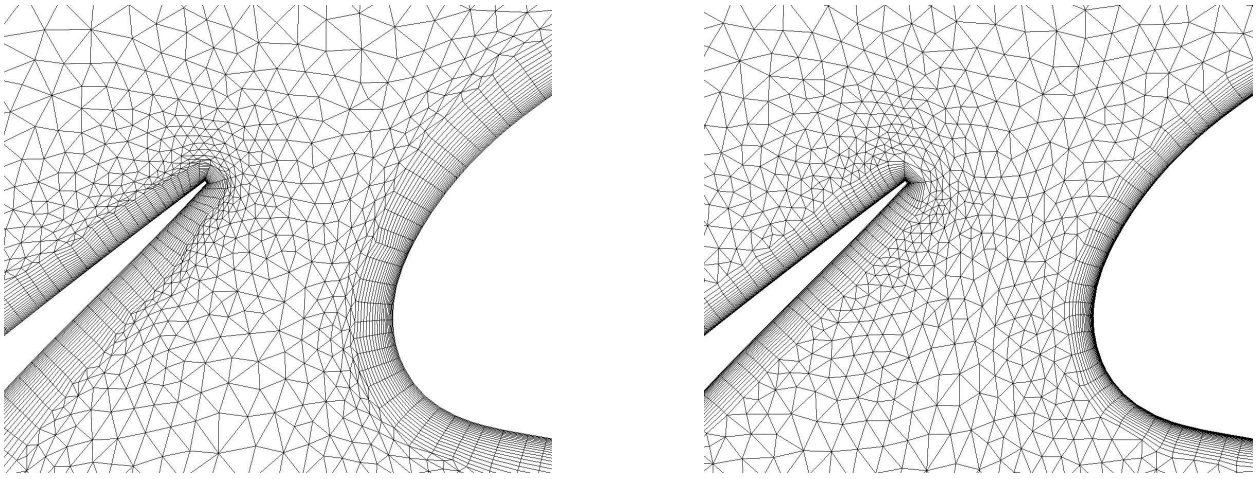
Figure 23. Original and Optimized Meshes for a Multi-Element Wing Geometry



(a) Original Viscous Mesh

(b) Optimized Viscous Mesh

Figure 24. Original and Optimized Meshes for a Multi-Element Wing Geometry



(a) Original Viscous Mesh

(b) Optimized Viscous Mesh

Figure 25. Original and Optimized Meshes for a Multi-Element Wing Geometry

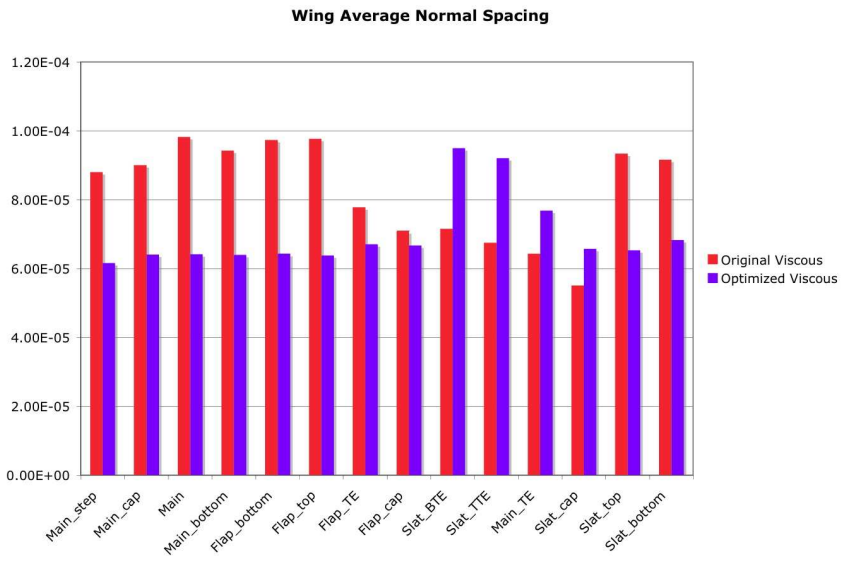
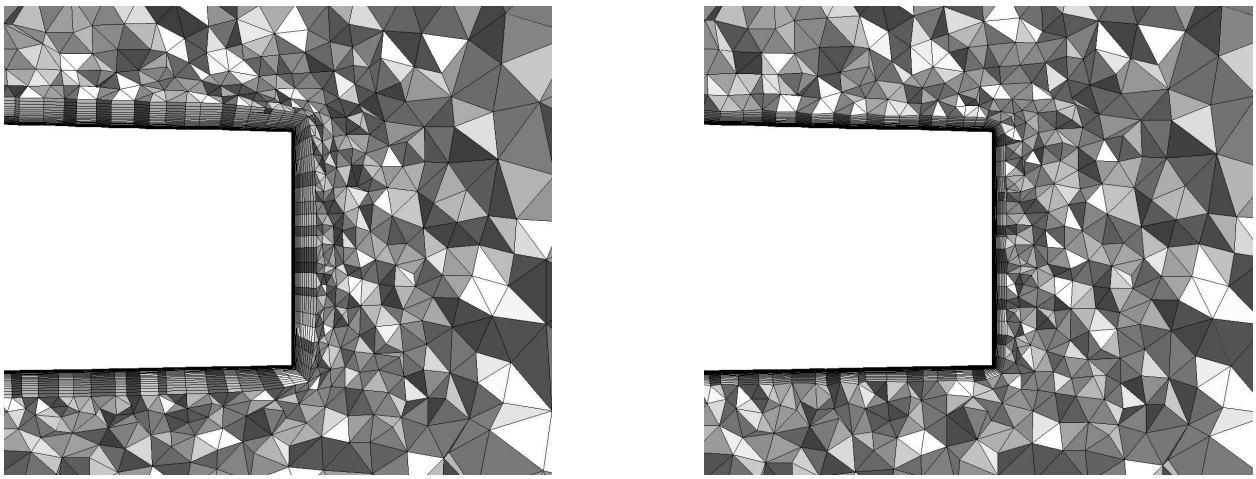


Figure 26. Average Normal Spacing for a Multi-Element Wing Geometry



(a) Original Viscous Mesh

(b) Optimized Viscous Mesh

Figure 27. Original and Optimized Meshes for a Multi-Element Wing Geometry

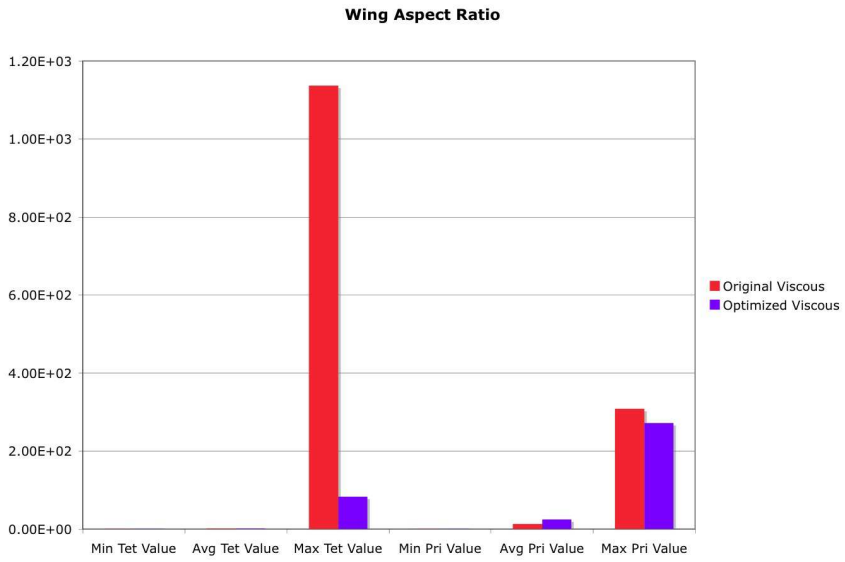


Figure 28. Aspect Ratio for a Multi-Element Wing Geometry

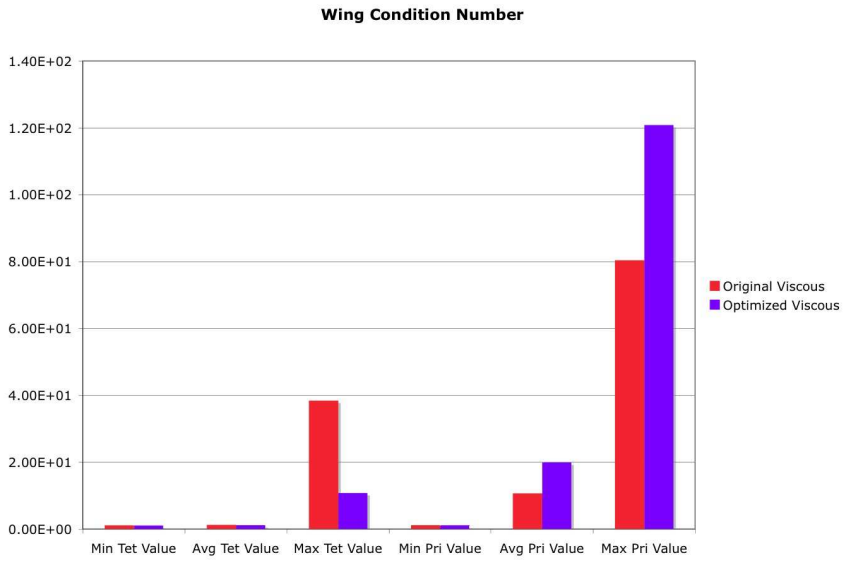


Figure 29. Condition Number for a Multi-Element Wing Geometry

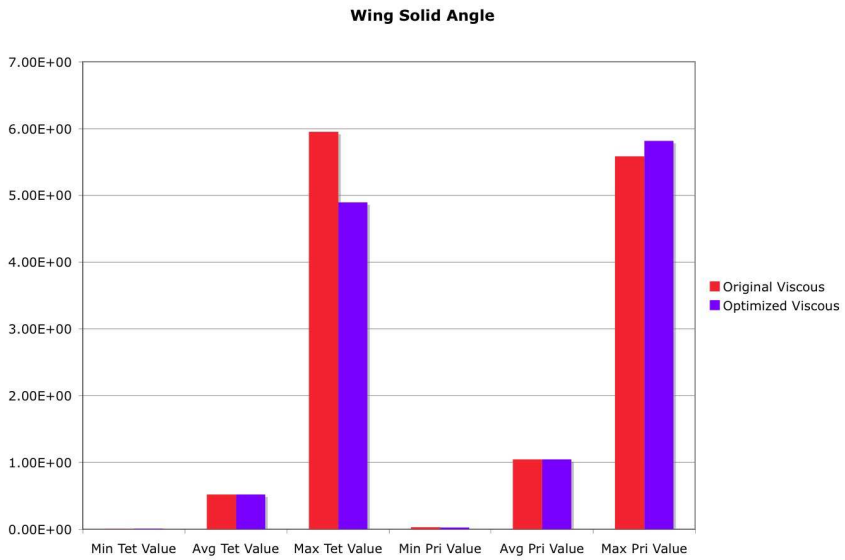


Figure 30. Solid Angle for a Multi-Element Wing Geometry

Appendix

1. Aspect Ratio

The definition of Aspect Ratio for tetrahedra[14] used in optimization is given in equation 7.

$$\begin{aligned} AR &= \frac{\rho_{out}}{3\rho_{in}} \\ \rho_{out} &= \frac{\sqrt{(a+b+c)(a+b-c)(a+c-b)(b+c-a)}}{24V} \\ \rho_{in} &= \frac{3V}{S_1+S_2+S_3+S_4} \end{aligned} \tag{7}$$

where V is the Volume, S_1, S_2, S_3 and S_4 are the surface areas of the four faces and $a = l_{12}l_{34}$, $b = l_{13}l_{24}$ and $c = l_{14}l_{23}$ are the products of opposite edge lengths. The value of aspect ratio varies from 1 for an ideal element to ∞ for badly shaped elements.

The definition of Aspect Ratio used in the measurement of grid quality metrics for other element types is the traditional definition given in equation 8.

$$AR = \frac{e_{max}}{e_{min}} \tag{8}$$

2. Jacobian

The Jacobian is constructed in 2D using the two edge vectors e_1 and e_2 as given in equation 9.

$$J = \begin{vmatrix} x_0 - x_1 & x_0 - x_2 \\ y_0 - y_1 & y_0 - y_2 \end{vmatrix} \tag{9}$$

The Jacobian is constructed in 3D using the three edge vectors e_1 , $e_2$and e_3 as given in equation 10.

$$J = \begin{vmatrix} x_0 - x_1 & x_0 - x_2 & x_0 - x_3 \\ y_0 - y_1 & y_0 - y_2 & y_0 - y_3 \\ z_0 - z_1 & z_0 - z_2 & z_0 - z_3 \end{vmatrix} \tag{10}$$

A positive value of the Jacobian indicates a valid element and a value below zero indicates an inverted element.

3. Condition Number

The condition number is defined[6,7] as the product of the Frobenius norm of the Jacobian Matrix with the Frobenius norm of its inverse and is it's formula is given in equation 11.

$$\kappa = \frac{|J||J^{-1}|}{3} \tag{11}$$

This condition number varies over the nodes of an element. The condition ranges from a value of 1 for an ideal element to ∞ for a degenerate element.

4. Normalized Jacobian

The Normalized Jacobian is the Jacobian with normalized edge vectors. The Jacobian is formulated in 2D as given in equation 12.

$$J = \begin{vmatrix} \frac{x_0 - x_1}{l_{01}} & \frac{x_0 - x_2}{l_{02}} \\ \frac{y_0 - y_1}{l_{01}} & \frac{y_0 - y_2}{l_{02}} \end{vmatrix} \tag{12}$$

where l_{01}, l_{02} are the edge vector lengths. It is defined in 3D as given in equation 13.

$$J = \begin{vmatrix} \frac{x_0 - x_1}{l_{01}} & \frac{x_0 - x_2}{l_{02}} & \frac{x_0 - x_3}{l_{03}} \\ \frac{y_0 - y_1}{l_{01}} & \frac{y_0 - y_2}{l_{02}} & \frac{y_0 - y_3}{l_{03}} \\ \frac{z_0 - z_1}{l_{01}} & \frac{z_0 - z_2}{l_{02}} & \frac{z_0 - z_3}{l_{03}} \end{vmatrix} \tag{13}$$

where l_{01}, l_{02}, l_{03} are the edge vector lengths. A positive value of the Normalized Jacobian indicates a valid element and ranges from 0 to 1 while a value below zero indicates an inverted element.

5. Solid Angle

Solid Angle[14] is defined in equation 14

$$sin(\frac{\theta_0}{2}) = \frac{12V}{\prod_{1\leq i\leq j\leq 3}(l_{0i}+l_{0j}+l_{ij})(l_{0i}+l_{0j}-l_{ij})} \tag{14}$$

where V is the volume and l_{01}, l_{02}, l_{03} are the edge vector lengths. An orthogonal corner would have a solid angle of $\pi/2$ while degenerate elements would have solid angles near 0 to 2π.

References

[1]Zhou, T. and Shimada, K., "An Angle-Based Approach to Two-dimensional Mesh Smoothing," *The 9th International Meshing Roundtable*, 2000, pp. 373–84.

[2]Farhat, C., Degand, C., Koobus, B., and Lesoinee, M., "Torsional springs for two-dimensional dynamic unstructured fluid meshes," *Comp. Meth. Appl. Mech. Engrg*, Vol. 163, 1998, pp. 231–245.

[3]Winslow, A., "Numerical Solution of the Quasilinear Poisson Equation in a Nonuniform Triangle Mesh," *Journal of Computational Physics*, Vol. 2, 1967, pp. 149–1724.

[4]Freitag, L. ., Jones, M., and Plassman, P., "An efficient parallel algorithm for mesh smoothing," *Proceedings of the Fifth International Meshing Roundtable*, 1995, pp. 47–58.

[5]Freitag, L. . and Gooch, C. O., "On combining Laplacian and optimization based mesh smoothing techniques," *AMD Trends in Unstructured Mesh Generation*, Vol. 220, 1997, pp. 37–43.

[6]Knupp, P., "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*, Vol. 48, 2000, pp. 401–420.

[7]Knupp, P., "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*, Vol. 48, 2000, pp. 1165–1185.

[8]Gridgen, "PointWise," *http://www.pointwise.com/gridgen*, Gridgen Users Manual.

[9]Karman, S. L. J., "Hierarchical Unstructured Mesh Generation," *42nd Aerospace Sciences Meeting and Exhibit, January 2004, Reno, NV*, 2004, AIAA-2004-0613.

[10]Karman, S. L. J., Anderson, W., and Sahasrabudhe, M., "Mesh Generation Using Unstructured Computational Meshes and Elliptic Partial Differential Equation Smoothing," *43rd Aerospace Sciences Meeting and Exhibit, January 2005, Reno, NV*, 2005, AIAA-2005-0923.

[11]Karman, S. L. J., "Unstructured Viscous Layer Insertion Using Linear-Elastic Smoothing," *44th Aerospace Sciences Meeting and Exhibit, January 2006, Reno, NV*, 2006, AIAA-2005-0531.

[12]"ONERA M6 validation case," *http://www.grc.nasa.gov/WWW/wind/valid/m6wing/m6wing.html*.

[13]Fieldview, "Intelligent Light Inc," *http://www.ilight.com*.

[14]Liu, A. and Joe, B., "Relationship between Tetrahedron Shape Measures," *BIT*, Vol. 34, 1994, pp. 268–287.