AIAA JOURNAL
Vol. 44, No. 6, June 2006

# Mesh Generation Using Unstructured Computational Meshes and Elliptic Partial Differential Equation Smoothing

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

**A novel approach for generating unstructured meshes using elliptic smoothing is presented. Like structured mesh-generation methods, the approach begins with the construction of a computational mesh. The computational mesh is used to solve elliptic partial differential equations that control grid point distributions and improve mesh quality. Two types of elliptic partial differential equations are employed: modified linear-elastic theory and Winslow equations, with or without forcing functions. Results are included to illustrate the use of these methods for unstructured mesh generation, mesh untangling, and mesh movement and viscous mesh creation.**

## Nomenclature

| | | |
|---|---|---|
| $E$ | = | Young's modulus of elasticity |
| $S$, d$S$ | = | element surface area |
| $u$ | = | physical coordinate or physical perturbation component |
| $V$, d$V$ | = | element volume |
| $v$ | = | physical coordinate or physical perturbation component |
| $w$ | = | physical coordinate or physical perturbation component |
| $x$ | = | physical coordinate or computational coordinate |
| $y$ | = | physical coordinate or computational coordinate |
| $z$ | = | physical coordinate or computational coordinate |
| $\alpha$ | = | Winslow or linear-elasticity coefficient |
| $\beta$ | = | Winslow coefficient |
| $\Gamma$ | = | element boundary edge |
| $\gamma$ | = | Winslow or linear-elasticity coefficient |
| $\zeta$ | = | computational coordinate |
| $\eta$ | = | computational coordinate |
| $\theta$ | = | linear-elasticity coefficient |
| $\nu$ | = | Poisson's ratio |
| $\xi$ | = | computational coordinate |
| $\Phi$ | = | physical $u$ coordinate forcing function |
| $\psi$ | = | physical $v$ coordinate forcing function |
| $\Omega$ | = | element area or physical $w$ coordinate forcing function |

## I. Introduction

THE mesh-generation process followed by most unstructured mesh-generation methods is very similar to the process followed by structured mesh-generation methods. The process for generating structured meshes in two dimensions involves distributing grid points along edges of the domain, followed by distributing points in the interior of the domain. The process in three dimensions adds a step for distributing grid points on faces of the domain before distributing points in the interior of the domain. The latter steps in these processes might require mesh smoothing to improve the quality of the final mesh. This might be necessary because the methods for initializing the interior grid point locations can result in grid skewness or inverted elements. Unstructured mesh-generation methods typically follow a similar process, and the final mesh might also require smoothing or optimizing to improve the quality. Smoothing is also required when boundary motion is involved. How the smoothing is accomplished is usually quite different between structured and unstructured methods.

Structured grid-generation methods often employ elliptic partial differential equation smoothing techniques to improve mesh quality. This type of smoothing offers users a high degree of control over grid point spacing and grid line angularity. On the other hand, unstructured grid-generation methods tend to use techniques that involve some type of node averaging or node perturbation/optimization to improve mesh quality.[1,2] Unfortunately, averaging and optimizing methods are not always robust and can actually degrade the quality of the mesh. This is especially true for meshes that contain sharp edges and viscous clustering.

Elliptic partial differential equation smoothing methods solving the Laplace or Poisson equations have not been applied to the generation of unstructured style meshes for one basic reason; the nonexistence of a mapping from the physical domain to computational domain where the smoothing equations are solved. As such, the benefits of using elliptic smoothing methods have not been realized in the unstructured grid environment. This paper describes a novel approach for smoothing unstructured meshes by using meshes in a computational domain and smoothing based on elliptic partial differential equations to produce smooth, high-quality meshes in the physical domain.

A brief review of structured elliptic smoothing will be provided to establish a foundation for the unstructured elliptic smoothing and to show the parallels between the structured and unstructured smoothing processes. Then two types of elliptic partial differential equation techniques for use with unstructured meshes will be described. The first technique uses a modified linear-elastic theory to control grid point distribution for moving boundary problems. A variation of the method can be used to generate viscous meshes for nonmoving boundaries. The second technique solves the same elliptic partial differential equations, Laplace and Poisson, often used by structured mesh smoothing methods. Several examples will be used to illustrate various uses of these elliptic smoothing techniques on unstructured meshes.

## II. Structured Mesh Smoothing

Elliptic smoothing has been used for a long time to improve the quality of structured meshes. The origin of the method can be traced back to Winslow[3] and made popular by Thompson et al.[4] Structured elliptic smoothing is incorporated in many commercial

Presented as Paper 2005-0923 at the 43rd Aerospace Sciences Meeting, Reno, NV, 10–13 January 2005; received 3 February 2005; revision received 29 November 2005; accepted for publication 12 December 2005. 

*Research Professor, Graduate School of Computational Engineering. Associate Fellow AIAA.

†Professor, Graduate School of Computational Engineering. Associate Fellow AIAA.

‡Research Associate and Graduate Student, Graduate School of Computational Engineering.

mesh-generation packages, including Gridgen.[5] The success of the method is caused by the mathematical basis of the derivation and the relationships inherent in the transformation between computational and physical domains. Examining the details of this transformation is instructive and will be used to show parallels between structured mesh smoothing methods and the unstructured mesh smoothing methods, which is the topic of this paper.

The two-dimensional computational domain is represented by $(\xi, \eta)$ space, and the two-dimensional physical domain is represented by $(x, y)$ space. The mapping between these two domains is defined in the following transformations:

$$\xi = \xi(x, y), \qquad x = x(\xi, \eta)$$
$$\eta = \eta(x, y), \qquad y = y(\xi, \eta) \tag{1}$$

The elliptic smoothing equations are expressed as a Laplacian operator on the computational coordinates set equal to zero or set equal to forcing functions in the form of $P$ and $Q$ or $\Phi$ and $\Psi$, as follows.

$P$–$Q$ form:

$$\nabla^2\xi = \frac{\partial^2\xi}{\partial x^2} + \frac{\partial^2\xi}{\partial y^2} = P, \qquad \nabla^2\eta = \frac{\partial^2\eta}{\partial x^2} + \frac{\partial^2\eta}{\partial y^2} = Q$$

$\Phi$–$\Psi$ form:

$$\nabla^2\xi = \frac{\partial^2\xi}{\partial x^2} + \frac{\partial^2\xi}{\partial y^2} = (\nabla\xi \cdot \nabla\xi)\Phi$$
$$\nabla^2\eta = \frac{\partial^2\eta}{\partial x^2} + \frac{\partial^2\eta}{\partial y^2} = (\nabla\eta \cdot \nabla\eta)\Psi \tag{2}$$

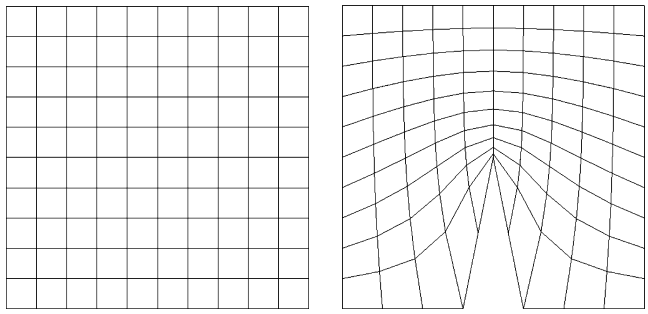
These Poisson equations are currently cast in physical space and represent the smooth variation of computational coordinates in physical space. In other words, the gradients of $\xi$ and $\eta$ will be constant, and the second derivatives of $\xi$ and $\eta$ will be zero. When the forcing functions are set equal to zero, the equations convert to the simple Laplace equations. Solutions to Laplace equations satisfy the maximum/minimum principle, which means the dependent variables on the interior of the domain are bounded by the values on the boundary of the domain. This helps ensure that grid lines do not cross if the boundaries of the domain are chosen to be constant $\xi$ and constant $\eta$ grid lines.

These equations are difficult to solve in the current reference frame because the computational coordinates are known and the physical coordinates are desired. So the equations are transformed to computational space in order to solve for the desired physical coordinates. The final transformed equations are sometimes referred to as the Winslow equations.[3] The final form of the equations using the $\Phi$–$\Psi$ formulation is

$$\alpha(x_{\xi\xi} + \Phi x_\xi) - 2\beta x_{\xi\eta} + \gamma(x_{\eta\eta} + \Psi x_\eta) = 0$$
$$\alpha(y_{\xi\xi} + \Phi y_\xi) - 2\beta y_{\xi\eta} + \gamma(y_{\eta\eta} + \Psi y_\eta) = 0$$
$$\alpha = x_\eta^2 + y_\eta^2, \qquad \beta = x_\xi x_\eta + y_\xi y_\eta, \qquad \gamma = x_\xi^2 + y_\xi^2 \tag{3}$$

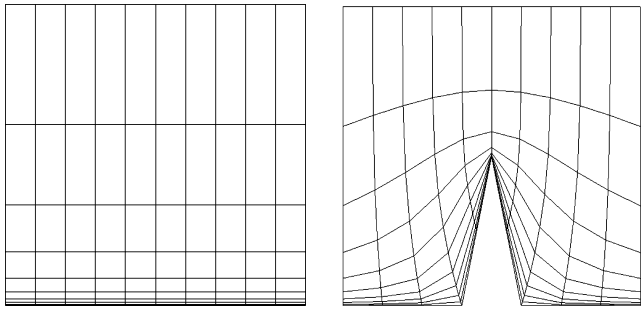
Smoothing methods that solve these equations are sometimes referred to as Winslow smoothing methods. Solutions to these equations will produce a smooth mesh in the physical domain. These solutions are typically obtained using an equally spaced mesh in computational space as shown in Fig. 1a. The physical mesh is shown in Fig. 1b. The interior grid lines do not protrude through the spike on the bottom wall. Even though the side boundaries in the physical domain are clustered toward the bottom boundary, the interior grid lines do not protrude through the spike on the bottom wall. The mesh is smooth and does not contain any grid line crossing.

It is possible to use a computational domain that is not equally spaced. The difference formulas for first and second derivatives of a function $f$ in the $x$ and $y$ directions, derived from Taylor-series expansion, are as follows:



a) b)

**Fig. 1 Computational and physical domains for structured elliptic smoothing.**



a) b)

**Fig. 2 Clustered computational mesh and physical mesh with elliptic Laplacian smoothing.**

$$\Delta_i = x_{i+1,j} - x_{i,j}, \qquad \nabla_i = x_{i,j} - x_{i-1,j}$$
$$\Delta_j = y_{i,j+1} - y_{i,j}, \qquad \nabla_j = y_{i,j} - y_{i,j-1}$$
$$f_x = \frac{\nabla_i^2 f_{i+1,j} + \left(\Delta_i^2 - \nabla_i^2\right) f_{i,j} - \Delta_i^2 f_{i-1,j}}{\nabla_i \Delta_i^2 + \Delta_i \nabla_i^2}$$
$$f_y = \frac{\nabla_j^2 f_{i,j+1} + \left(\Delta_j^2 - \nabla_j^2\right) f_{i,j} - \Delta_j^2 f_{i,j-1}}{\nabla_j \Delta_j^2 + \Delta_j \nabla_j^2}$$
$$f_{xx} = \frac{2(\Delta_i f_{i-1,j} + \nabla_i f_{i+1,j} - (\nabla_i + \Delta_i) f_{i,j})}{\nabla_i \Delta_i^2 + \Delta_i \nabla_i^2}$$
$$f_{yy} = \frac{2(\Delta_j f_{i,j-1} + \nabla_j f_{i,j+1} - (\nabla_j + \Delta_j) f_{i,j})}{\nabla_j \Delta_j^2 + \Delta_j \nabla_j^2}$$
$$f_{xy} = \frac{\left\{\begin{array}{l} \nabla_i^2\left[\nabla_j^2 f_{i+1,j+1} + \left(\Delta_j^2 - \nabla_j^2\right) f_{i+1,j} - \Delta_j^2 f_{i+1,j-1}\right] \\ +\left(\Delta_i^2 - \nabla_i^2\right)\left[\nabla_j^2 f_{i,j+1} + \left(\Delta_j^2 - \nabla_j^2\right) f_{i,j} - \Delta_j^2 f_{i,j-1}\right] \\ -\Delta_i^2\left[\nabla_j^2 f_{i-1,j+1} + \left(\Delta_j^2 - \nabla_j^2\right) f_{i-1,j} - \Delta_j^2 f_{i-1,j-1}\right] \end{array}\right\}}{\left(\nabla_i \Delta_i^2 + \Delta_i \nabla_i^2\right)\left(\nabla_j \Delta_j^2 + \Delta_j \nabla_j^2\right)} \tag{4}$$

If these formulas are used with an unequally spaced, but still Cartesian computational mesh that mimics the desired clustering of the boundaries of the physical mesh, the result is the mesh shown in Fig. 2. The side boundaries of the computational domain are now clustered to match the spacing of the side boundaries in the physical domain. The interior of the physical mesh is also clustered to the bottom boundary, and the grid lines do not cross.

This result is similar to traditional Winslow methods that solve Poisson's equations with control functions. Poisson control functions must be defined to produce the desired grid clustering and/or the desired grid line angularity near the boundaries. Many different techniques for constructing these control functions have been developed over the years.[6,7] The results in Fig. 2 indicate that it

is possible to manipulate the computational mesh and control the spacing in the physical mesh without using forcing functions.

## III. Unstructured Mesh Smoothing

Until recently, smoothing methods for unstructured meshes have been limited to simple averaging techniques, sometimes called spring analogy methods, or optimization techniques using node perturbations that attempt to maximize or minimize a function that is a measure of smoothness or quality.[1,2] These methods have been used with some success to untangle meshes and improve element shape quality, but they do not offer the robustness afforded by elliptic methods, especially when sharp geometry shapes are involved. A linear-elastic theory method of smoothing was described in Ref. 8, which showed increased robustness and improved results when applied to unstructured dynamic mesh problems. This paper revisits the linear-elastic approach and introduces an approach for solving the Winslow equations on unstructured meshes. This section will describe the details of the existing linear-elastic-theory approach first, followed by a detailed description of the Winslow approach.

### A. Linear-Elastic-Theory Smoothing

Linear-elastic-theory smoothing is generally used for mesh movement where boundaries of the mesh are deforming and interior mesh points need to be adjusted to produce a valid mesh.[8] It can also be used to generate the initial mesh under certain circumstances, as will be shown later.

In this approach it is assumed the mesh obeys the isotropic linear elastic relations, with a constant modulus of elasticity (Young's modulus) as shown next:

$$\nabla^2 u + \frac{1}{1-2\nu}\frac{\partial}{\partial x}\nabla\cdot\boldsymbol{V} = 0, \qquad \nabla^2 v + \frac{1}{1-2\nu}\frac{\partial}{\partial y}\nabla\cdot\boldsymbol{V} = 0 \tag{5}$$

where $\nu$ in the denominator is Poisson's ratio and the nodal displacement vector is given by $\boldsymbol{V} = u\hat{i} + v\hat{j}$. The parameter $\nu$ is typically manipulated so that the coefficient $1/(1-2\nu)$ is equal to the aspect ratio of the local cell. This produces stiffness in regions with high-aspect-ratio cells and ensures that boundary-layer elements track closely with the local boundary as it moves. The solution to these equations is a vector field defining the displacement of each node.

The terms in the equations can be expanded and cast in a form similar to the Winslow equations:

$$\left(1+\frac{1}{1-2\nu}\right)\frac{\partial^2 u}{\partial x^2} + \left(\frac{1}{1-2\nu}\right)\frac{\partial^2 v}{\partial x\partial y} + \frac{\partial^2 u}{\partial y^2} = 0$$

$$\frac{\partial^2 v}{\partial x^2} + \left(\frac{1}{1-2\nu}\right)\frac{\partial^2 u}{\partial x\partial y} + \left(1+\frac{1}{1-2\nu}\right)\frac{\partial^2 v}{\partial y^2} = 0 \tag{6}$$

Yang and Mavriplis showed how linear-elastic smoothing can be used to perform very large deformations of inviscid and viscous meshes.[9] They chose to use a different form of the linear-elastic relations, where the modulus of elasticity $E$ was allowed to vary and Poisson's ratio was constant. Under these assumptions their formulation can be transformed to a form similar to Eq. (6):

$$\frac{\partial}{\partial x}\left[\frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\frac{\partial u}{\partial x}\right] + \frac{\partial}{\partial y}\left[\frac{E}{2(1+\nu)}\frac{\partial u}{\partial y}\right] + \frac{\partial}{\partial x}\left[\frac{E\nu}{(1+\nu)(1-2\nu)}\frac{\partial v}{\partial y}\right] + \frac{\partial}{\partial y}\left[\frac{E}{2(1+\nu)}\frac{\partial v}{\partial x}\right] = 0$$

$$\frac{\partial}{\partial x}\left[\frac{E}{2(1+\nu)}\frac{\partial v}{\partial x}\right] + \frac{\partial}{\partial y}\left[\frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\frac{\partial v}{\partial y}\right] + \frac{\partial}{\partial x}\left[\frac{E}{2(1+\nu)}\frac{\partial u}{\partial y}\right] + \frac{\partial}{\partial y}\left[\frac{E\nu}{(1+\nu)(1-2\nu)}\frac{\partial u}{\partial x}\right] = 0 \tag{7}$$

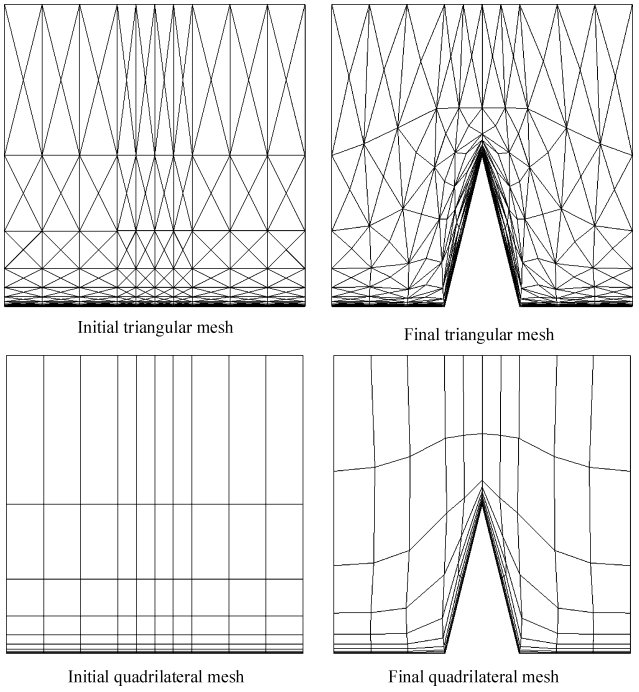



**Fig. 3 Linear-elastic smoothing applied to spike geometry.**

Yang and Mavriplis chose to set Young's modulus $E$ to the inverse of the element volume or to the inverse of the distance from the deforming boundaries. This formulation has been tested by the present authors with $E$ prescribed as the inverse of the cell volume and is indeed more robust for large deformations than the formulation in Eqs. (5) and (6). Poisson's ratio for Eq. (7) must be specified within the valid range from 0 to $\frac{1}{2}$. In structural mechanics, for metals in the elastic range the value of Poisson's ratio lies between $\frac{1}{4}$ and $\frac{1}{3}$ (Ref. 10). Numerical experiments were performed to determine a reasonable value for Poisson's ratio in the context of mesh smoothing. A global value of 0.2 for Poisson's ratio was chosen. Further experimentation should be conducted to fully understand the influence of Poisson's ratio on the physical mesh.

Linear-elastic-theory smoothing can be used to generate the spike mesh shown earlier by deforming the bottom boundary from the initial horizontal line in the computational mesh to the final geometry through a series of smaller incremental steps. The resulting triangular and quadrilateral meshes are shown in Fig. 3. The clustering of the original meshes is maintained as the bottom boundary is moved from the original location to the final position.

### B. Elliptic (Winslow) Smoothing

The use of Winslow smoothing with unstructured meshes is difficult because a mapping from an unstructured physical domain to a computational domain is not available. A local $(\xi, \eta)$ coordinate system can be constructed at the cell or node level, but a global mapping of the entire domain is generally not defined.[11] Knupp used trigonometric functions to approximate the computational coordinates and Taylor-series expansions of these coordinates to compute first and second derivatives for the Winslow equations.[11] The results indicated slightly improved mesh quality with comparable computational performance. The possibility of extending the method to three dimensions was questioned in the conclusions.

However, if a "valid" mesh with the same element topology is available, then the Winslow equations can be solved on that mesh. This was the basic technique used in Ref. 8 to solve the linear-elastic equations. The starting mesh was a valid mesh for the current boundary position. The deformation of the boundary nodes was defined using the displacement vector $\boldsymbol{V}$. The solution to the linear elastic relations was obtained on the original mesh. That solution is the displacement vector for each node of the entire mesh. The new grid point positions were computed by adding this displacement vector to the old positions.

Winslow equations can also be solved using an existing computational mesh that matches the element topology of the physical mesh. The equations must first be discretized for an unstructured mesh. For convenience, an exchange of variables is performed from $(x, y)$ to $(u, v)$ and from $(\xi, \eta)$ to $(x, y)$. Then the Winslow equations are reformulated, consistent with the nomenclature used in the linear-elastic equations, as follows:

$$\alpha(u_{xx} + \Phi u_x) - 2\beta u_{xy} + \gamma(u_{yy} + \Psi u_y) = 0$$
$$\alpha(v_{xx} + \Phi v_x) - 2\beta v_{xy} + \gamma(v_{yy} + \Psi v_y) = 0$$
$$\alpha = u_y^2 + v_y^2, \qquad \beta = u_x u_y + v_x v_y, \qquad \gamma = u_x^2 + v_x^2 \quad (8)$$

In this context the $x$ and $y$ coordinates are the computational coordinates, and the $u$ and $v$ coordinates are the physical coordinates.

One approach to solving these equations is to consider the coefficients $\alpha$, $\beta$, $\gamma$, $\Phi$, and $\Psi$ as constant and integrate the first equation over $x$–$y$ space:

$$\alpha \int\int (u_{xx} + \Phi u_x)\,\mathrm{d}\Omega - 2\beta \int\int u_{xy}\,\mathrm{d}\Omega + \gamma \int\int (u_{yy} + \Psi u_y)\,\mathrm{d}\Omega = 0 \quad (9)$$

$\Omega$ represents the area in two dimensions. The divergence theorem can be used to transform the double integrals into line integrals:

$$\alpha\left(\oint u_x \hat{n}_x\,\mathrm{d}\Gamma + \Phi \oint u \hat{n}_x\,\mathrm{d}\Gamma\right) - 2\beta \oint u_y \hat{n}_x\,\mathrm{d}\Gamma + \gamma\left(\oint u_y \hat{n}_y\,\mathrm{d}\Gamma + \Psi \oint u \hat{n}_y\,\mathrm{d}\Gamma\right) = 0$$
$$\alpha\left(\oint v_x \hat{n}_x\,\mathrm{d}\Gamma + \Phi \oint v \hat{n}_x\,\mathrm{d}\Gamma\right) - 2\beta \oint v_y \hat{n}_x\,\mathrm{d}\Gamma + \gamma\left(\oint v_y \hat{n}_y\,\mathrm{d}\Gamma + \Psi \oint v \hat{n}_y\,\mathrm{d}\Gamma\right) = 0 \quad (10)$$

These integral equations are discretized at the nodes using the control volume (area) for the node. Several choices for the control volume are shown in Fig. 4. For the first option, the faces of the control volume extend from one cell center to the midedge to the neighboring cell center. This is a dual of the mesh and results in no overlapping regions. The second option for the control volume would include the complete volumes (areas) of the surrounding elements. This would result in overlapping control volumes, but is still a valid option. The final option is a modification of the second whereby only a portion of a quadrilateral element is included in the node control volume. For quadrilateral elements only the triangle consisting of the node in question and its adjacent neighbors is included in the control volume. The advantage of this choice is the ability to formulate the system of equations using only triangular-shaped elements. All of these control volume options extend to three dimensions, where the third option would require only tetrahedral-shaped elements to formulate the system of equations.

Most of the integrals in Eq. (10) contain a first derivative of $u$ or $v$ with respect to the $x$ or $y$ computational directions. These first derivative terms are simply components of the gradient of a scalar function. In this case the scalar functions are the physical coordinates $u$ and $v$. The resulting discrete equation, after substituting for the first derivative terms, is as follows:

$$\frac{u_1}{2A}\left[\alpha\left(n_{x_2} + n_{x_3}\right)t_x - 2\beta\left(n_{y_2} + n_{y_3}\right)t_x + \gamma\left(n_{y_2} + n_{y_3}\right)t_y\right] + \frac{u_2}{2A}\left[\alpha\left(n_{x_1} + n_{x_3}\right)t_x - 2\beta\left(n_{y_1} + n_{y_3}\right)t_x + \gamma\left(n_{y_1} + n_{y_3}\right)t_y\right] + \frac{u_3}{2A}\left[\alpha\left(n_{x_1} + n_{x_2}\right)t_x - 2\beta\left(n_{y_1} + n_{y_2}\right)t_x + \gamma\left(n_{y_1} + n_{y_2}\right)t_y\right] + u_f[\alpha\Phi t_x + \gamma\Psi t_y] = 0$$
$$\frac{v_1}{2A}\left[\alpha\left(n_{x_2} + n_{x_3}\right)t_x - 2\beta\left(n_{y_2} + n_{y_3}\right)t_x + \gamma\left(n_{y_2} + n_{y_3}\right)t_y\right] + \frac{v_2}{2A}\left[\alpha\left(n_{x_1} + n_{x_3}\right)t_x - 2\beta\left(n_{y_1} + n_{y_3}\right)t_x + \gamma\left(n_{y_1} + n_{y_3}\right)t_y\right] + \frac{v_3}{2A}\left[\alpha\left(n_{x_1} + n_{x_2}\right)t_x - 2\beta\left(n_{y_1} + n_{y_2}\right)t_x + \gamma\left(n_{y_1} + n_{y_2}\right)t_y\right] + v_f[\alpha\Phi t_x + \gamma\Psi t_y] = 0 \quad (11)$$

Equations (11) represent a linear system of equations for a triangle with unknown variables $u_1$, $u_2$, $u_3$, $v_1$, $v_2$, and $v_3$. The area vector $(n_{xi}, n_{yi})$ corresponds to the outward-pointing normal vector opposite of node $i$. The vector $(t_x, t_y)$ corresponds to the area vector for the node under construction in the global system. Similar equations can be constructed for all triangles in the mesh, as well as subtriangles for the corners of the quadrilateral elements. Combining the equations from all elements in the mesh will result in a sparse matrix linear system of equations that can be solved for new physical coordinates using techniques, such as point-implicit with underrelaxation. Because the coefficients $\alpha$, $\beta$, $\gamma$, $\Phi$, and $\Psi$ were assumed frozen but are actually functions of the physical locations, an outer iteration loop is required to update these coefficient values. When a point-implicit method is used, an inner iteration loop is required to converge the linear system.

The Winslow smoothed spike mesh using an unstructured solver is shown in Fig. 5. The top two images correspond to the case in which the computational mesh is equally spaced. The bottom two images correspond to the case in which the sidewalls of the computational mesh match the spacing of the sidewalls of the physical mesh. As can be seen, the characteristics of the physical mesh using an unstructured solver are similar to the behavior of the physical meshes using the structured solver in Figs. 1 and 2, in that equally spaced computational meshes and Winslow without forcing functions will tend to drive the mesh toward an equal area mesh and clustering of the

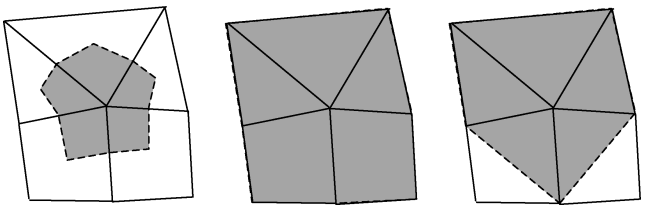


**Fig. 4 Several options for control volume surrounding a node.**

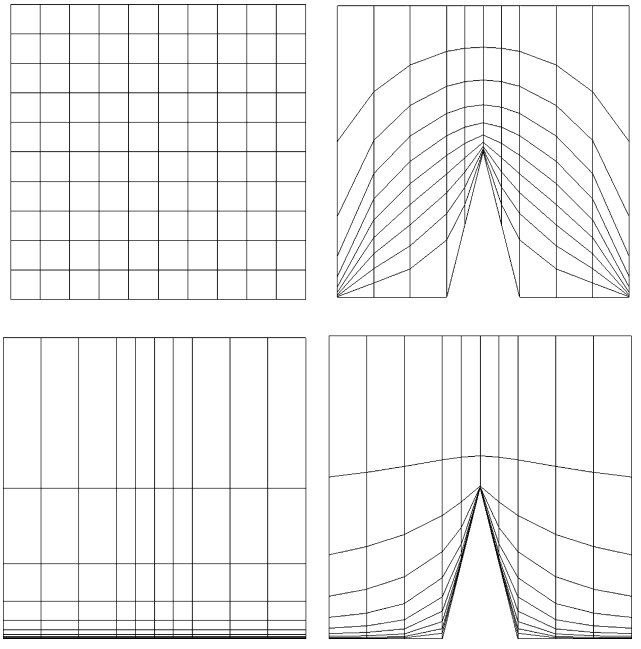


**Fig. 5 Solution to spike mesh using unstructured Winslow–Laplace smoothing.**

computational mesh without forcing functions will force the physical mesh to try and mimic the spacing in the computational mesh.

## IV. Computational Meshes

As stated earlier, Winslow smoothing requires a computational mesh in order to solve the governing equations. Structured mesh smoothing typically employs a rectangular Cartesian mesh for the computational mesh. This computational mesh is implied in the meshing process, in that the domains are generated assuming a rectangular topology with dimensions specified by the user. The structured computational mesh is assumed to be uniform and equally spaced. A computational mesh is not implied for unstructured meshes; therefore, one must be provided with the same element topology as the resulting physical mesh. This section will discuss options for obtaining a computational mesh. The options generally involve the generation of an initial, valid mesh by some means. The mesh might or might not conform to the actual geometry. The mesh might or might not contain the spacing desired in the physical mesh, such as clustering to viscous, no-slip walls. As in the case with structured elliptic smoothing, the placement of the grid points on the actual boundary and control of the mesh spacing will be handled in the solution of the Winslow equations.

The most common forms of unstructured meshes are triangular meshes in two dimensions and tetrahedral meshes in three dimensions. Hybrid meshes are also possible, where quadrilateral elements are introduced in two dimensions and triangular prisms, pyramid and hexahedral elements are introduced in three dimensions. The process of generating these meshes can involve several steps, such as extruding meshes from surfaces and tessellating volume regions with an advancing front technique or a Delaunay-based insertion method. If the process is successful, the resulting mesh can be used as a computational mesh. If it can be assumed that this initial mesh conforms to the true geometry, then the initial mesh can also serve as a physical mesh and will be an exact copy of the computational mesh. Winslow smoothing can then be used to exert further control over the distribution of the points and quality of the physical mesh. The computational mesh will remain unchanged in the process. The physical mesh will be modified to satisfy the Winslow equations, with or without forcing functions.

One possible approach for generating computational meshes was presented in Ref. 12. The approach is relatively automatic and produces quadrilateral-dominant meshes in two dimensions and hexahedral-dominant meshes in three dimensions.[12] This mesh-generation process is reversed from traditional approaches. The volume mesh is constructed first, followed by the creation of body-conforming elements at the boundaries. Details of the approach can be found in the reference. An example of the type of mesh produced with this method is shown in Fig. 6. This type of mesh is, by no means, required to perform Winslow smoothing on unstructured meshes. It is merely an example of an approach for generating volume meshes in an automatic manner. Computational meshes generated by other means are also used throughout this paper. The most important aspect about the initial computational mesh, whether it is the particular approach mentioned in Ref. 12 or any other approach, is that the computational mesh should be easily generated and be a valid mesh with all positive volumes (areas).

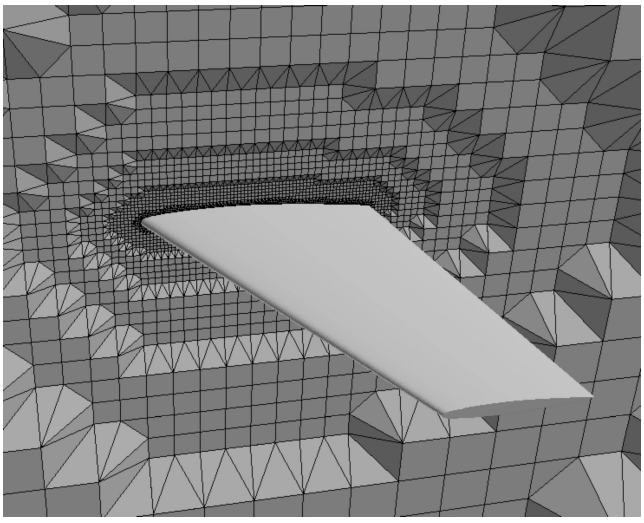


**Fig. 6 Example of a three-dimensional volume mesh for an ONERA M6 wing geometry.**

## V. Robust Generation of High-Quality Viscous Meshes

Winslow smoothing and linear-elastic smoothing can be combined to produce high-quality viscous meshes. Winslow smoothing can be used to generate the inviscid mesh, and linear-elastic smoothing can then be used to insert viscous layers at solid boundaries. An example is shown in Fig. 7 for a viscous mesh about an airfoil configuration. The inviscid mesh is shown on the left-hand side of the figure, and the final viscous mesh, after inserting 10 layers, is shown on the right side of the figure.

The inviscid mesh was generated using the method described in Ref. 12, although any valid inviscid mesh could have been used. During the generation of this inviscid mesh, Winslow smoothing without forcing functions was used to improve the quality of the elements near the airfoil surface. The initial computational and physical meshes prior to the Winslow smoothing were identical. Performing a general cutting of the Cartesian elements with the geometry created the boundary elements. The resulting surface distribution was uneven. The smoothing operation attempts to redistribute the boundary points more evenly. After performing the Winslow smoothing pass, slight differences between the computational and physical mesh occur near the airfoil surface. The similarity of the computational and physical meshes away from the airfoil demonstrates that the characteristics of the computational mesh are imposed on the physical mesh. A smoothing approach using spring analogy would likely distort the mesh far from the airfoil surface.

Inserting multiple layers at the airfoil surface created the viscous mesh. As each layer is inserted, the previous layer of boundary points is pushed a short distance into the interior of the domain in the "normal direction" using linear-elastic smoothing. The normal direction can be defined in a number of ways. Sharp corners, such as airfoil trailing edge can have multiple normal directions. Or the normal direction can be defined as the direction along the edge emanating from the boundary point in the direction of the interior point. Currently a single normal vector is computed for each boundary node by averaging the normal vectors of the adjacent edge faces.

The short distance that each node is pushed depends on the current local normal spacing and the desired normal spacing for the current layer. No point is pushed more than a percentage of the current edge length consistent with a specified geometric progression factor. If possible, the node is pushed a distance corresponding to the current layer spacing, assuming a desired initial spacing and the same geometric progression factor. As each layer is pushed into the interior, the interior mesh element shapes are preserved by the linear-elastic smoothing equations. Following this scenario, the normal direction at each boundary node is always the same for each layer insertion pass.

This approach to viscous mesh generation is different from marching methods that are sometimes used to generate viscous layers. Marching methods add the next layer on top of the previous

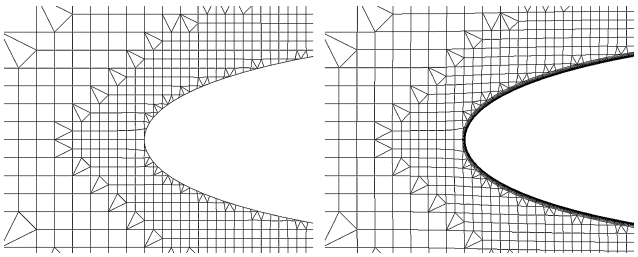


**Fig. 7 Combined approach for generating viscous meshes.**

layers. As new layers are added, the normal direction must be recomputed, which can eventually result in poorly defined normal vectors if proper care if not taken to control the normal vectors. Marching methods can also suffer from possible collisions in concave corners or with other fronts marching away from nearby surfaces. The current approach always inserts new layers at the boundary where the normal direction is well defined and does not change. The linear-elastic equations ensure that interior mesh element quality is maintained. In addition, the largest layer displacement occurs during the first layer insertion. The distance specified is limited to never be larger than the local element size. Each subsequent layer is pushed away from the boundary a smaller distance, following a reverse geometric progression, until the final layer is inserted at the desired viscous spacing at the wall. The process of inserting layers in this reversed mode using the linear-elastic smoothing is extremely robust.

## VI. Discussion of Linear-Elastic Smoothing vs Winslow Smoothing

The equations for Winslow smoothing and linear-elasticity smoothing can be combined into a generalized set of equations. When solving the linear-elastic equations, $u$ and $v$ represent grid point velocities. When solving Winslow equations, $u$ and $v$ represent the physical coordinates. The two-dimensional, combined equations are given next. The three-dimensional combined equations are included in the Appendix:

$$\frac{\partial}{\partial x}\left[\alpha_{11}\frac{\partial u}{\partial x}\right]+\frac{\partial}{\partial y}\left[\alpha_{12}\frac{\partial u}{\partial y}\right]+\alpha_{11}\Phi\frac{\partial u}{\partial x}+\alpha_{12}\Psi\frac{\partial u}{\partial y}-2\beta\frac{\partial^2 u}{\partial x\,\partial y}+\frac{\partial}{\partial x}\left[\theta_{11}\left(\frac{\partial v}{\partial y}\right)\right]+\frac{\partial}{\partial y}\left[\theta_{12}\frac{\partial v}{\partial x}\right]=0$$

$$\frac{\partial}{\partial x}\left[\alpha_{21}\frac{\partial v}{\partial x}\right]+\frac{\partial}{\partial y}\left[\alpha_{22}\frac{\partial v}{\partial y}\right]+\alpha_{21}\Phi\frac{\partial v}{\partial x}+\alpha_{22}\Psi\frac{\partial v}{\partial y}-2\beta\frac{\partial^2 v}{\partial x\,\partial y}+\frac{\partial}{\partial x}\left[\theta_{21}\frac{\partial u}{\partial y}\right]+\frac{\partial}{\partial y}\left[\theta_{22}\left(\frac{\partial u}{\partial x}\right)\right]=0$$

For linear elasticity:

$$\alpha_{11}=\frac{E(1-\nu)}{(1+\nu)(1-2\nu)},\qquad \alpha_{12}=\frac{E}{2(1+\nu)}$$

$$\theta_{11}=\frac{E\nu}{(1+\nu)(1-2\nu)},\qquad \theta_{12}=\frac{E}{2(1+\nu)}$$

$$\alpha_{21}=\frac{E}{2(1+\nu)},\qquad \alpha_{22}=\frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$$

$$\theta_{21}=\frac{E}{2(1+\nu)},\qquad \theta_{22}=\frac{E\nu}{(1+\nu)(1-2\nu)}$$

$$\beta=\Phi=\Psi=0$$

For Winslow:

$$\alpha_{11}=\alpha_{21}=u_y^2+v_y^2,\qquad \alpha_{12}=\alpha_{22}=u_x^2+v_x^2$$

$$\beta=u_xu_y+v_xv_y,\qquad \theta_{11}=\theta_{12}=\theta_{21}=\theta_{22}=0 \tag{12}$$

Winslow smoothing and linear-elasticity smoothing can be used in a couple of different ways. When the boundaries are stationary, Winslow smoothing can be used to improve the quality of the mesh. Forcing functions can provide additional control over grid point distribution. The linear-elasticity smoothing method requires boundary motion. If there is no boundary motion, linear-elasticity smoothing cannot be used to smooth the mesh. When boundary motion is involved, either smoothing can be used to control the mesh movement and quality.

For the stationary case, if the boundary distribution of the computational mesh matches the boundary distribution of the physical mesh and the forcing functions are equal to zero, Winslow smoothing will result in the physical mesh matching the computational mesh, exactly. This means that the interior grid points can be scrambled and Winslow smoothing can be used to recover the original mesh. This is obviously an academic exercise, as described. If the boundary distributions are identical, one could always obtain the original mesh by simply substituting the computational coordinates for the physical coordinates. The significance of this result occurs when boundary motion is involved. In a simulation of a moving boundary, Winslow smoothing will return the original mesh if the boundary points return to their original positions, whereas the linear-elasticity smoothing method is not guaranteed to produce the original mesh under similar circumstances.

Both smoothing methods can be used to reposition interior nodes when changes are made to the boundary nodes, such as in design optimization applications. In the case of Winslow smoothing, the character of the computational mesh is maintained in the physical mesh. Recall that the clustering of the computational mesh in the structured grid was reflected in the physical mesh for the spike problem. As the deformation from the computational mesh grows, the clustering of the mesh near the boundary will change to satisfy the governing grid equations. As the deviation from the computational mesh increases in Winslow smoothing, the clustering of the interior points will change in the usual manner associated with Winslow smoothing, namely pulling out of concave corners and clustered more toward convex corners. This differs from the linear-elasticity method because the clustering of interior grid points is controlled by the Young's modulus and Poisson's ratio terms. High-aspect-ratio cells are maintained relative to the local boundary motion. If more control over the point distribution is desired, then Winslow smoothing with forcing functions can be employed.

## VII. Forcing Functions

The spacing and topology of the nonuniform computational meshes have a direct influence on the characteristics of the physical mesh. This was demonstrated earlier on the spike example problem and the airfoil case described in Sec. V. However, it is sometimes useful to add forcing functions to provide increased control over grid point placement and grid line angularity. Research is underway to develop methods for utilizing forcing function for the general case. Currently only a fixed-grid method has been implemented and tested.

One of the simplest uses of forcing functions is to recover an existing mesh or slightly modify an existing mesh to improve element quality. Under these circumstances the required forcing functions can be computed from the existing mesh. The Winslow equations must first be solved for the forcing functions, as shown next using the same notation as Eq. (3):

$$\alpha(u_{xx}+\Phi u_x)-2\beta u_{xy}+\gamma(u_{yy}+\Psi u_y)=0$$

$$\alpha(v_{xx}+\Phi v_x)-2\beta v_{xy}+\gamma(v_{yy}+\Psi v_y)=0$$

solve for $\Phi$ and $\Psi$:

$$\begin{bmatrix}\alpha u_x & \gamma u_y\\ \alpha v_x & \gamma v_y\end{bmatrix}\begin{Bmatrix}\Phi\\ \Psi\end{Bmatrix}=\begin{Bmatrix}2\beta u_{xy}-\alpha u_{xx}-\gamma u_{yy}\\ 2\beta v_{xy}-\alpha v_{xx}-\gamma v_{yy}\end{Bmatrix}$$

$$\begin{Bmatrix}\Phi\\ \Psi\end{Bmatrix}=\begin{bmatrix}\alpha u_x & \gamma u_y\\ \alpha v_x & \gamma v_y\end{bmatrix}^{-1}\begin{Bmatrix}2\beta u_{xy}-\alpha u_{xx}-\gamma u_{yy}\\ 2\beta v_{xy}-\alpha v_{xx}-\gamma v_{yy}\end{Bmatrix} \tag{13}$$

The right-hand side of the equation set is driven by the residual of the Winslow equations with no forcing functions, the terms contained in the { } brackets. So an existing mesh that already satisfies the Winslow equations with no forcing functions will result in zero control functions. A special case is where the physical mesh and computational mesh match exactly.

If one desires to slightly modify the existing physical mesh to improve the smoothness and eliminate kinks in the mesh, the resulting

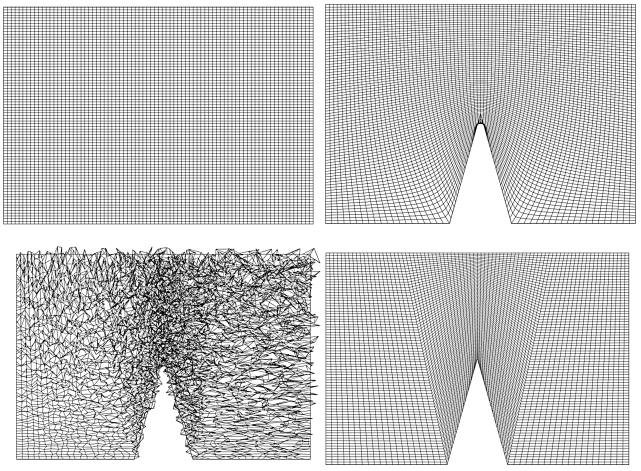


**Fig. 8 Scrambled spike mesh recovered using Winslow smoothing with fixed-grid forcing functions.**

forcing functions can be averaged with neighboring values. These averaged forcing functions can then be used with Winslow smoothing to improve an existing mesh. If no averaging of the computed forcing functions is performed, the original mesh can be fully recovered. To demonstrate, the original spike problem is initialized with a transfinite interpolation procedure. The computational mesh is an equally spaced Cartesian mesh shown in the upper-left-hand corner of Fig. 8. The forcing functions are computed using Eq. (13). The physical mesh is scrambled to produce the mesh in the lower-left-hand corner of Fig. 8. The original forcing functions are applied to the Winslow equations to produce the physical mesh on the lower-right-hand corner of the figure. In this case no averaging of the forcing functions was performed, and the resulting physical mesh is the original mesh created with the transfinite interpolation procedure. If the forcing functions are not used, the resulting physical mesh is shown in the upper-right-hand corner of the figure.

## VIII. Results

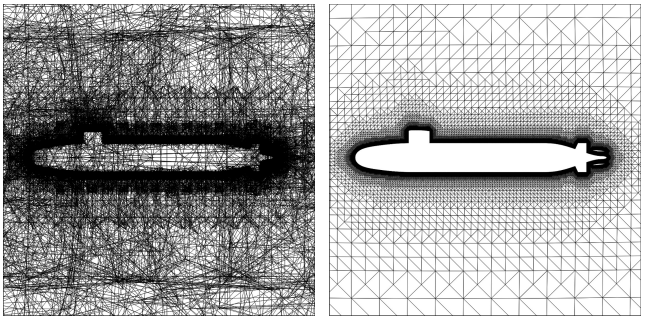
Several examples are included to demonstrate the geometric complexity that can be modeled using these mesh smoothing techniques and to illustrate the various uses of the two forms of elliptic smoothing described in this paper. Winslow smoothing is demonstrated on a mesh untangling case and a mesh movement case. No forcing functions were used because research is underway to construct forcing functions for optimizing mesh quality. Linear-elastic smoothing is demonstrated on a mesh moving case and three-dimensional viscous layer insertion cases. Winslow solutions and linear-elastic solutions are computed with the same program. The timings reported for the three-dimensional insertion cases using linear elasticity would be indicative of the computational cost when computing Winslow solutions.

### A. Submarine

To further demonstrate the ability of Winslow smoothing to recover the computational mesh, a physical mesh for a two-dimensional submarine shape was intentionally tangled and then smoothed using Winslow equations with zero forcing functions. Initially the computational mesh was copied into the physical mesh, and so the two meshes were exactly identical. Then the physical mesh was tangled, shown in Fig. 9a. The recovered mesh after smoothing is shown in Fig. 9b. The Winslow equations were fully converged to machine zero and the resulting mesh coordinates compared exactly with the original physical coordinates.

### B. Oscillating NACA0012 Airfoil

An oscillating airfoil is used to demonstrate the use of the two smoothing techniques for moving boundary problems. The first case uses Winslow smoothing with airfoil rotations of $\pm 10$ deg about the quarter-chord location. The mesh for the 10-deg rotation is shown in Fig. 10a. The mesh for the original location, or 0-deg rotation, is shown in Fig. 10b. And the mesh for the negative 10-deg rotation is shown in Fig. 10c.



**a)** **b)**

**Fig. 9 Tangled mesh for submarine shape untangled using Winslow smoothing without forcing functions.**

The second case uses linear-elasticity smoothing to move the interior points to follow the prescribed rotation of the airfoil boundary points. The prescribe motion is the same $\pm 10$-deg rotation about the quarter-chord location. The final mesh for the 10-deg rotation is shown in Fig. 11a. The mesh for the original location is shown in Fig. 11b. And the mesh for the negative 10-deg rotation is shown in Fig. 11c. This particular mesh is the same mesh from Fig. 10 converted to all triangles and smoothed using a separate computer program that did not handle quadrilateral elements.

This illustrates that both smoothing techniques are plausible methods to perform this type of mesh movement. Rotations of $\pm 10$ deg are considered moderate rotations. Larger rotations are possible using these methods. But as the rotation angle increases, the skewing of the elements near the leading edge and trailing edge increases, and eventually both methods will fail. For instance, it would be inconceivable to expect either method to perform a 180-deg rotation. The element connectivity does not change, and leading-edge points and trailing-edge points would switch position. If the outer boundary points were held fixed, the mesh near the airfoil would twist around the quarter-chord point. There is a limit to how far the rotation can be handled with any fixed-connectivity movement method, whether it is elliptic smoothing or spring analogy. Elliptic smoothing offers the hope that larger deformation angles can be achieved. Beyond that limit another approach must be considered, such as complete remeshing or overset.

### C. ONERA M6 Viscous Layers

Viscous layers were inserted into an existing mesh for an ONERA M6 configuration, shown in Fig. 12. The initial inviscid mesh, provided by personnel at NASA Langley Research Center, contained 183,342 tetrahedral elements and 33,718 mesh points. Linear-elastic smoothing was used to insert 15 viscous layers above the surface of the wing. The initial spacing was specified as $1.0e-4$, and the growth of the spacing followed a geometric progress factor of 1.15. The final mesh contained 93,808 mesh points and added 118,755 prismatic elements to the viscous mesh. The case was computed on a Macintosh PowerBook 1.5-GHz G4 laptop in a total time of 474 s and required fewer than 100 MB of memory, requiring roughly 1 KB of memory per point. Each insertion of a layer required the solution of the linear-elastic equations to compute the perturbation of the existing nodes away from the wing surface. The solver used a combination of an implicit line solver and an implicit point solver. Implicit lines were constructed from boundary points marching outward. Mesh points that were not included in an implicit line solution were computed using a point-implicit scheme. As an example, the last inserted layer contained 6703 implicit lines, with maximum line length of 23. The number of free points computed with the point implicit scheme was 22,816. A total of 100 iterations were used to converge the solution at least two orders of magnitude during each layer insertion step. The cost per mesh point per iteration was approximately $5.23e-06$ s. The same case was computed on a 2.4-GHz

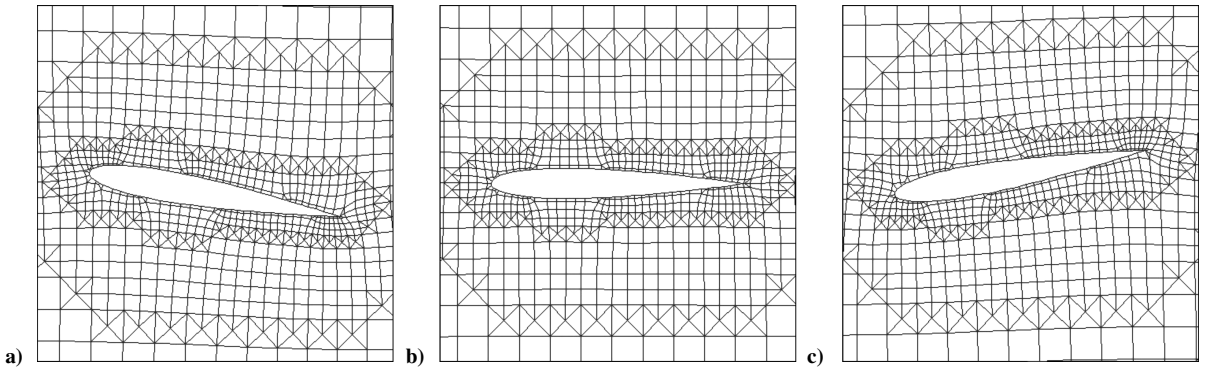



**Fig. 10 NACA0012 mesh for 10-deg airfoil oscillation using Winslow smoothing.**

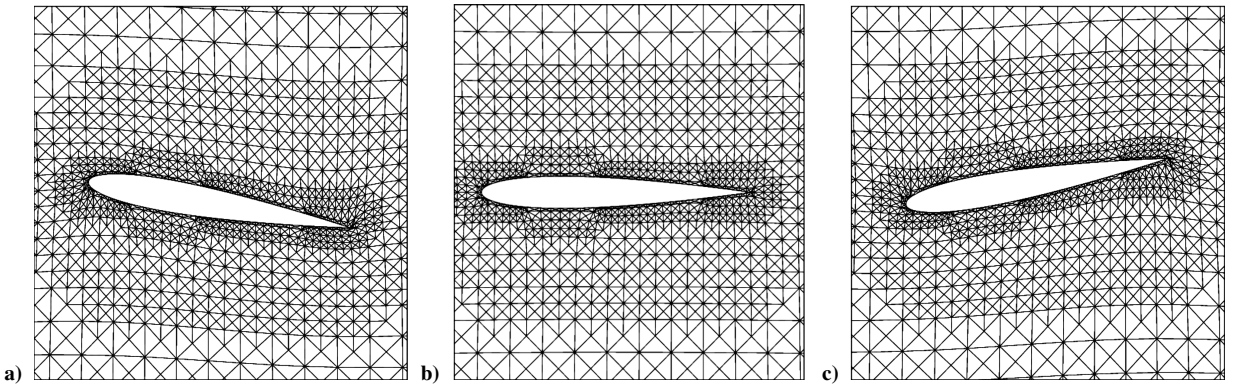



**Fig. 11 NACA0012 mesh for 10-deg airfoil oscillation using linear elasticity.**

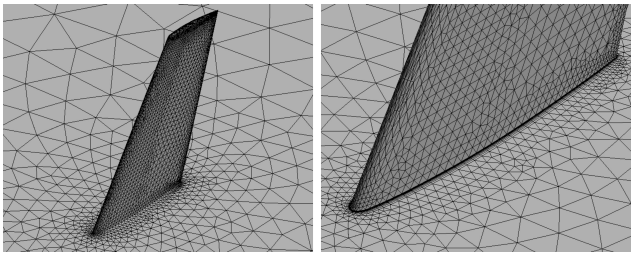


**Fig. 12 Viscous layers inserted into an existing ONERA M6 mesh using linear elasticity.**

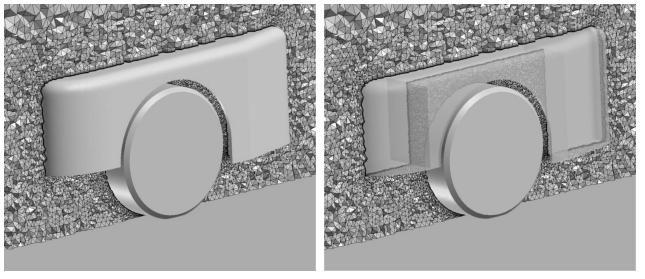


**Fig. 13 Viscous mesh for tire in a wheel well.**

Intel Linux workstation in a total time of 251 s. This translates into $2.77e{-}06$ s per mesh point per iteration.

### D. Tire in Wheel Well

Linear-elastic smoothing was used to insert viscous layers into an existing mesh for a configuration that included a tire in a wheel well on the ground. The configuration is shown in Fig. 13 with a crinkled surface displayed for a cutting plane located in the center of the wheel well. The transparent view on the right shows the existence of the mesh in the interior of the wheel well. Viscous layers were

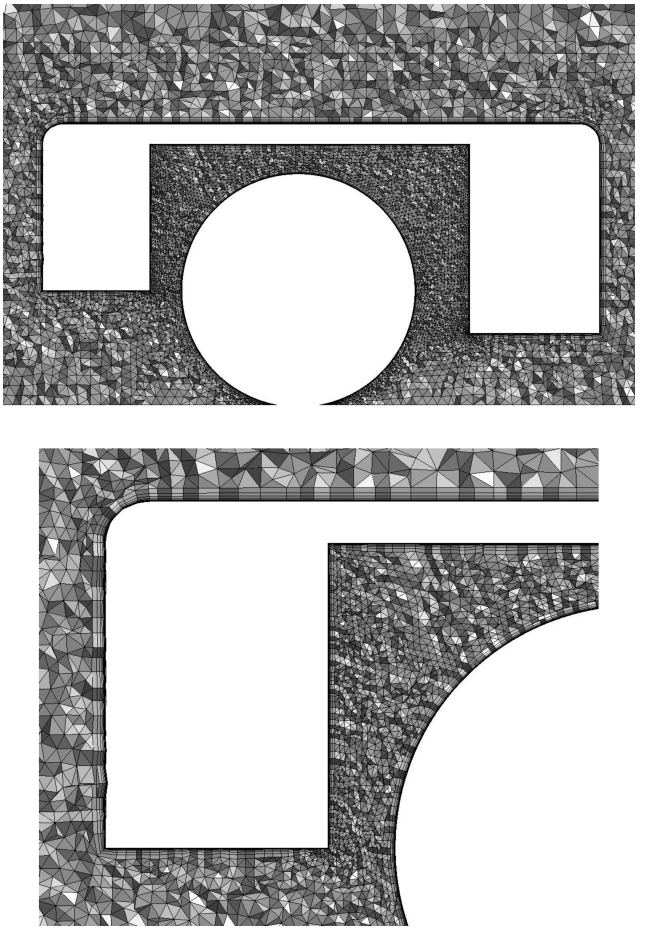


**Fig. 14 Side views of mesh for centerline cutting plane.**

inserted on all surfaces shown except for the ground plane, where tangent slip boundary conditions were to be applied.

A total of 19 layers was inserted with an initial spacing of $5.0e-06$ and a geometric progression factor of 1.2. Fifty iterations were used to solve the linear-elastic equations during each insertion step, with convergence reaching greater than two orders of magnitude. The total time to compute the mesh on a 2.4-GHz Linux workstation was 71.3 mins. The initial mesh consisted of 493,308 mesh points and 2,802,007 tetrahedral elements. The final mesh contained 1,378,765 mesh points and added 1,766,582 prismatic elements. The memory requirements were roughly 1 KB per point, which were similar to the previous case. An approximate computational cost per mesh point per iteration was $3.2e-06$. Side views of the mesh for the same cutting plane are shown in Fig. 14. The viscous mesh contains the desired initial spacing at the wall and matches the tetrahedral mesh at the top layer.

## IX. Conclusions

A method for performing elliptic partial differential equation smoothing on unstructured meshes that can use two different partial differential equations has been described. The first equation set is modified linear elasticity theory. These equations can be used to deform interior nodes based on boundary node deformation. Moving boundary problems solved with linear-elastic smoothing resulted in a reasonable deformation of the interior nodes while maintaining valid element shapes. Clustering of node near the deforming boundary is maintained as the boundary moves. A robust method for generating viscous meshes using layer insertion was described. This method used the linear-elastic form of smoothing to insert new layers at the boundary and push the interior points away from the surface in a very robust manner. The resulting viscous meshes are generally of a high quality, and the approach is generally more robust than an extrusion method.

The second equation set is Winslow's equations. This equation set requires a computational mesh and can be used to smooth initial meshes and smooth meshes with boundary motion. With no forcing functions, the interior grid spacing of the physical mesh mimics the spacing of the computational mesh. As a result, any clustering of the computational mesh will be reflected in the physical mesh. Forcing functions are possible and offer increased control over grid point placement and clustering. Additional research is underway to take full advantage of the forcing functions in the Winslow equations to control grid points spacing and grid line angularity at the boundaries in viscous meshes and to optimize the element quality of inviscid meshes. The full potential of the ability to solve Winslow equations on unstructured meshes will begin to be realized with the development of proper forcing functions to manipulate the mesh and control the element quality.

Improvements in mesh quality can be measured in a number of different ways. Some of the more common quality measures include statistical analysis of cell area/volume, cell aspect ratio, cell condition number, cell skewness, corner angle/solid angle measures, and corner Jacobians. Statistical analyses of any number of these quality measures can be performed before and after performing smoothing. The purpose for applying the Winslow smoothing to some of the static boundary cases included in this paper was to improve the quality of the elements near the boundary, resulting from the particular method used to generate the computational mesh, a hierarchical Cartesian method with general cutting of the elements at the boundary. This process produced an uneven distribution of the boundary points. The Winslow smoothing more evenly distributed the boundary points and made only minor changes to interior elements. As such, a statistical analysis was not included. Other static cases in this paper involved demonstrating the recovery of an initial mesh that was intentionally tangled. The quality of the final mesh would match exactly the quality of the original mesh. Future research into more general forcing functions to control viscous spacing or flowfield solution adaptation will require quantifiable evidence of improved mesh quality. This could include a measure of improved solution accuracy as a quality measure.

The computational cost of the smoothing scheme is reasonable for the cases computed thus far. Timings were provided for the three-dimensional cases only. The smaller case was computed in a matter of minutes on a Macintosh laptop and a Linux workstation. The larger case was computed on a Linux workstation in just over an hour. Much larger viscous insertion meshes have been generated, but not shown. These are generally computed on a 64-bit Linux workstation with 32 GB of RAM. The time to insert viscous layers is problem dependent, but is generally of the same order of magnitude required to generate the initial inviscid tetrahedral mesh. No timings for the Winslow smoothing were presented. However, the linear-elastic smoothing timings are indicative of the computational time required for the Winslow smoothing because the same matrix construction and matrix solution algorithms are used. The number of iterations required to achieve a desired level of convergence will vary from problem to problem. Winslow smoothing with forcing functions will also require more time because the matrix construction algorithm does not handle the creation of the forcing functions.

Future development will parallelize the solver to run across distributed memory clusters. This will dramatically improve the run times and is essential to make the dynamic mesh problems practical. These elliptic smoothing techniques are more complicated than algebraic or spring analogy methods, and so it is understandable that more computing time is required. The improved robustness and quality is achieved at an increased computational cost, and parallel computing will be necessary to solve the very large problems.

### Appendix: Three-Dimensional Combined Equations

The three-dimensional equations for Winslow smoothing and linear-elasticity smoothing can be combined into a generalized set of equations and are included for reference.[5] The numerical method used to discretize these equations is an extension of the method described earlier for the two-dimensional form of the equations

$$
\begin{aligned}
&\frac{\partial}{\partial x}\left[\alpha_{11}\frac{\partial u}{\partial x}\right]+\frac{\partial}{\partial y}\left[\alpha_{12}\frac{\partial u}{\partial y}\right]+\frac{\partial}{\partial z}\left[\alpha_{13}\frac{\partial u}{\partial z}\right]+\alpha_{11}\Phi\frac{\partial u}{\partial x}\\
&\quad+\alpha_{12}\Psi\frac{\partial u}{\partial y}+\alpha_{13}\Omega\frac{\partial u}{\partial z}+2\left[\beta_1\frac{\partial^2 u}{\partial x\partial y}+\beta_2\frac{\partial^2 u}{\partial y\partial z}+\beta_3\frac{\partial^2 u}{\partial x\partial z}\right]\\
&\quad+\frac{\partial}{\partial x}\left[\theta_{11}\left(\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)\right]+\frac{\partial}{\partial y}\left[\theta_{12}\frac{\partial v}{\partial x}\right]+\frac{\partial}{\partial z}\left[\theta_{13}\frac{\partial w}{\partial x}\right]=0\\
&\frac{\partial}{\partial x}\left[\alpha_{21}\frac{\partial v}{\partial x}\right]+\frac{\partial}{\partial y}\left[\alpha_{22}\frac{\partial v}{\partial y}\right]+\frac{\partial}{\partial z}\left[\alpha_{23}\frac{\partial v}{\partial z}\right]\\
&\quad+\alpha_{21}\Phi\frac{\partial v}{\partial x}+\alpha_{22}\Psi\frac{\partial v}{\partial y}+\alpha_{23}\Omega\frac{\partial v}{\partial z}\\
&\quad+2\left[\beta_1\frac{\partial^2 v}{\partial x\partial y}+\beta_2\frac{\partial^2 v}{\partial y\partial z}+\beta_3\frac{\partial^2 v}{\partial x\partial z}\right]+\frac{\partial}{\partial x}\left[\theta_{21}\frac{\partial u}{\partial y}\right]\\
&\quad+\frac{\partial}{\partial y}\left[\theta_{22}\left(\frac{\partial u}{\partial x}+\frac{\partial w}{\partial z}\right)\right]+\frac{\partial}{\partial z}\left[\theta_{23}\frac{\partial w}{\partial y}\right]=0\\
&\frac{\partial}{\partial x}\left[\alpha_{31}\frac{\partial w}{\partial x}\right]+\frac{\partial}{\partial y}\left[\alpha_{32}\frac{\partial w}{\partial y}\right]+\frac{\partial}{\partial z}\left[\alpha_{33}\frac{\partial w}{\partial z}\right]\\
&\quad+\alpha_{31}\Phi\frac{\partial w}{\partial x}+\alpha_{32}\Psi\frac{\partial w}{\partial y}+\alpha_{33}\Omega\frac{\partial w}{\partial z}\\
&\quad+2\left[\beta_1\frac{\partial^2 w}{\partial x\partial y}+\beta_2\frac{\partial^2 w}{\partial y\partial z}+\beta_3\frac{\partial^2 w}{\partial x\partial z}\right]+\frac{\partial}{\partial x}\left[\theta_{31}\frac{\partial u}{\partial z}\right]\\
&\quad+\frac{\partial}{\partial y}\left[\theta_{32}\frac{\partial v}{\partial z}\right]+\frac{\partial}{\partial z}\left[\theta_{33}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\right]=0
\end{aligned}
\tag{14}
$$

For linear elasticity:

$$\beta_1 = \beta_2 = \beta_3 = 0, \qquad \Phi = \Psi = \Omega = 0$$

$$\alpha_{11} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}, \qquad \alpha_{12} = \frac{E}{2(1+\nu)}, \qquad \alpha_{13} = \frac{E}{2(1+\nu)}$$

$$\theta_{11} = \frac{E\nu}{(1+\nu)(1-2\nu)}, \qquad \theta_{12} = \frac{E}{2(1+\nu)}, \qquad \theta_{13} = \frac{E}{2(1+\nu)}$$

$$\alpha_{21} = \frac{E}{2(1+\nu)}, \qquad \alpha_{22} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}, \qquad \alpha_{23} = \frac{E}{2(1+\nu)}$$

$$\theta_{21} = \frac{E}{2(1+\nu)}, \qquad \theta_{22} = \frac{E\nu}{(1+\nu)(1-2\nu)}, \qquad \theta_{23} = \frac{E}{2(1+\nu)}$$

$$\alpha_{31} = \frac{E}{2(1+\nu)}, \qquad \alpha_{32} = \frac{E}{2(1+\nu)}, \qquad \alpha_{33} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$$

$$\theta_{31} = \frac{E}{2(1+\nu)}, \qquad \theta_{32} = \frac{E}{2(1+\nu)}, \qquad \theta_{33} = \frac{E\nu}{(1+\nu)(1-2\nu)}$$

For Winslow:

$$\theta_{ij} = 0, \qquad \boldsymbol{r} = (u, v, w)$$

$$\alpha_{11} = \alpha_{21} = \alpha_{31} = (\boldsymbol{r}_y \cdot \boldsymbol{r}_y)(\boldsymbol{r}_z \cdot \boldsymbol{r}_z) - (\boldsymbol{r}_y \cdot \boldsymbol{r}_z)(\boldsymbol{r}_y \cdot \boldsymbol{r}_z)$$

$$\beta_1 = (\boldsymbol{r}_y \cdot \boldsymbol{r}_z)(\boldsymbol{r}_z \cdot \boldsymbol{r}_x) - (\boldsymbol{r}_x \cdot \boldsymbol{r}_y)(\boldsymbol{r}_z \cdot \boldsymbol{r}_z)$$

$$\alpha_{12} = \alpha_{22} = \alpha_{32} = (\boldsymbol{r}_z \cdot \boldsymbol{r}_z)(\boldsymbol{r}_x \cdot \boldsymbol{r}_x) - (\boldsymbol{r}_z \cdot \boldsymbol{r}_x)(\boldsymbol{r}_z \cdot \boldsymbol{r}_x)$$

$$\beta_2 = (\boldsymbol{r}_z \cdot \boldsymbol{r}_x)(\boldsymbol{r}_x \cdot \boldsymbol{r}_y) - (\boldsymbol{r}_y \cdot \boldsymbol{r}_z)(\boldsymbol{r}_x \cdot \boldsymbol{r}_x)$$

$$\alpha_{13} = \alpha_{23} = \alpha_{33} = (\boldsymbol{r}_x \cdot \boldsymbol{r}_x)(\boldsymbol{r}_y \cdot \boldsymbol{r}_y) - (\boldsymbol{r}_x \cdot \boldsymbol{r}_y)(\boldsymbol{r}_x \cdot \boldsymbol{r}_y)$$

$$\beta_3 = (\boldsymbol{r}_x \cdot \boldsymbol{r}_y)(\boldsymbol{r}_y \cdot \boldsymbol{r}_z) - (\boldsymbol{r}_z \cdot \boldsymbol{r}_x)(\boldsymbol{r}_y \cdot \boldsymbol{r}_y)$$

## Acknowledgment

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

## References

[1]Freitag, L. A., and Knupp, P. M., "Tetrahedral Element Shape Optimization via the Jacobian Determinant and Condition Number," *Proceedings of the 8th International Meshing Roundtable*, Sandia National Lab., Albuquerque, NM, 1999, pp. 247–258.

[2]Brewer, M., Diachin, L. F., Knupp, P., Leurent, T., and Melander, D., "The Mesquite Mesh Quality Improvement Toolkit," *Proceedings of the 12th International Meshing Roundtable*, Sandia National Lab., Albuquerque, NM, 2003, pp. 239–250.

[3]Winslow, A., "Numerical Solution of the Quasilinear Poisson Equation in a Nonuniform Triangle Mesh," *Journal of Computational Physics*, Vol. 1, No. 2, 1966, pp. 149–172.

[4]Thompson, J. F., Thames, F. C., and Mastin, C. W., "Boundary-Fitted Curvilinear Coordinate Systems for Solution of Partial Differential Equations on Fields Containing Any Number of Arbitrary Two-Dimensional Bodies," NASA CR-2729, July 1977.

[5]Steinbrenner, J. P., Chawner, J. R., and Fouts, C. L., "The GRIDGEN 3D Multiple Block Grid Generation System," Flight Dynamics Lab., Wright Research and Development Center, Final Rept. WRDC-TR-90-3022, Vol. 1, Wright–Patterson AFB, OH, July 1990.

[6]Thomas, P. D., and Middlecoff, J. F., "Direct Control of the Grid Point Distribution in Meshes Generated by Elliptic Equations," *AIAA Journal*, Vol. 18, No. 6, 1979, pp. 652–656.

[7]Sorenson, R. L., "A Computer Program to Generate Two-Dimensional Grids About Airfoils and Other Shapes by Use of Poisson's Equations," NASA TM-81198, May 1980.

[8]Nielsen, E. J., and Anderson, W. K., "Recent Improvements in Aerodynamic Design Optimization on Unstructured Meshes," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 1155–1163.

[9]Yang, Z., and Mavriplis, D. J., "Unstructured Dynamic Meshes with Higher-Order Time Integration Schemes for the Unsteady Navier–Stokes Equations," AIAA Paper 2005-1222, Jan. 2005.

[10]Hornsey, E., McFarland, D., Muhlbauer, K., and Smith, B., *Mechanics of Materials, An Individualized Approach*, Houghton Mifflin, Boston, 1977, p. 6.

[11]Knupp, P., "Winslow Smoothing on Two-Dimensional Unstructured Meshes," *Engr. with Computers*, Vol. 15, No. 3, 1999, pp. 263–268.

[12]Karman, S. L., Jr., "Hierarchical Unstructured Mesh Generation," AIAA Paper 2004-0613, Jan. 2004.

D. Gaitonde
*Associate Editor*