AIAA 91-0721

Numerical Aspects of Computing Viscous High Reynolds Number Flows on Unstructured Meshes

Timothy J. Barth
CFD Branch
NASA Ames Research Center
Moffett Field, CA

29th Aerospace Sciences Meeting
January 7–10, 1991/Reno, Nevada

Numerical Aspects of Computing High Reynolds Number Flows on Unstructured Meshes

Timothy J. Barth
CFD Branch
NASA Ames Research Center
Moffett Field, Ca

§ Abstract

Various numerical and practical aspects associated with the solution of the Navier-Stokes equations on triangular meshes are addressed. We depart from the standard "element" data structure used in finite elements which describes a mesh element-wise from knowledge of the vertices of each element. Instead, we use an "edge" data structure which describes a mesh edge-wise given the vertices of each edge and neigboring cell information. In developing algorithms for the Navier-Stokes equations, the use of the edge data structure necessitates the rederivation of many of the discrete operators appearing in the Navier-Stokes equations. In this paper, edge formulas for the Galerkin and finite-volume discretization of gradient, divergence, Hessian and Laplacian operators are derived. The Galerkin discretization (with linear elements) of the Laplacian operator reveals that a discrete maximum principle is guaranteed in two space dimensions if and only if the mesh is a Delaunay triangulation. Sufficient conditions for a discrete maximum principle in three space dimensions are developed and found to be in agreement with standard finite element theory. Other issues which are equally important in the solution of the Navier-Stokes equations are addressed. Some issues concerning the generation of highly stretched triangular meshes are reviewed. A triangulation method which minimizes the maximum angle in triangles is shown to have desirable properties for highly stretched meshes. Other practical aspects of computing Navier-Stokes flow including the modeling of turbulence on unstructured meshes are discussed and a new turbulence modeling strategy is presented. The turbulence modeling capability is then coupled with an implicit solution strategy. Numerical results for high Reynolds number flow about single and multi-element airfoils are presented.

§ Introduction

Several algorithms for solving the Euler and Navier-Stokes equations on unstructured simplex (triangular) meshes now exist based on finite element or finite volume methodologies [1,2,3,4,5,6,7,8,9,10,11]. It is not surprising that many of the finite volume schemes that assume solution variables at the vertices of the mesh and piecewise linear distributions can be reinterpreted within the framework of finite elements or vice-versa. Performing this alternate interpretation is often a fruitful exercise. In particular, we consider an "edge" data structure which describes a mesh on an edge-by-edge basis by storing for each edge the two vertices of the edge and the centroids of the two neighboring cells. Although the edge data structure describes any two-dimensional mesh with nonoverlapping cells, we will concentrate on meshes composed of triangles. The edge data structure has been used previously in finite volume schemes for the Euler equations [1]. The use of an edge data structure in finite elements represents a departure from the conventional finite element data structure which describes a mesh element-wise by storing the vertices of each element. The development of numerical algorithms for solving the Navier-Stokes using the edge data structure necessitates the rederivation of "edge formulas" (formulas that are implemented on an edge-by-edge basis) representing discretized forms of gradient, divergence, Hessian, and Laplacian operators. Edge-based formulations can be advantageous in theory and application. In a later section, a simple edge formula is derived for the discretization of the Laplacian operator. In this form, precise theoretical conditions for a discrete maximum principle can be ascertained. This is in contrast to the standard result in finite element theory concerning a discrete maximum principle (Ciarlet [12]) which is based on an element analysis and gives sufficient but not necessary conditions. An application which exploits the edge formulation of the discrete gradient and divergence operators can be found in the paper by Hammond and Barth [13]. In this work a recent theorem by Chrobak and Eppstein [14] concerning directed graphs with minimum out-degree is exploited in conjunction with the edge formulation in the development of a massively parallel algorithm for solving the Euler equations on the Thinking Machines CM-2 computer.

The remaining portion of the paper will address practical issues associated with solving the Navier-Stokes equations on unstructured meshes. The generation of highly stretched meshes poses a new challenge in mesh generation. Some popular mesh generation methods fail in fundamental ways for highly stretched meshes. We will offer one technique which avoids the major pitfalls. For high Reynolds number flows, modeling the effects of turbulence is a particularly difficult task. A new turbulence model for wall bounded flows developed in [24] is adapted for computation on un-

structured meshes. Finally, numerical calculation of turbulent Navier-Stokes flow about single and multiple component airfoils will be presented using a hybrid strategy of sparse matrix solves and explicit time stepping.

§ Edge-based Formulations

Consider a 2-D unstructured mesh with vertices, edges, and cells (faces) independently numbered. The edge data structure for a 2-D mesh requires exactly four pieces of information for each interior edge of the mesh:

(1) The two vertices which form the edge

(2) The two adjacent cell centroids which share the edge

In the remaining portion of this section we will show that the discretization of several common differential operators can be reformulated into a form compatible with the edge data structure. Recall that for a given set of vertices and bounding edges, many different triangulations exist. The edge data structure in 2-D has the property that the storage requirement is *independent* of the particular triangulation. This is true because the total number of edges $n(e)$ (interior and boundary) is uniquely determined once given the number of vertices $n(v)$, number of boundary edges $n(e)_{boundary}$ and the number of closed polygonal curves $n(c)$ which are formed from these bounding edges. The exact formula is well known.

$$n(e) = 3\,(n(v) + n(c) - 2) - n(e)_{boundary} \qquad (2-\text{D Triangulation})$$

For 3-D meshes of tetrahedra an exact formula relating edges and vertices does not exist. Even so, by ignoring boundary effects, an approximate formula which relates the number of edges to the number of vertices and tetrahedra $n(T)$ is given by

$$n(e) \approx n(v) + n(T) \qquad (3-\text{D Triangulation})$$

The number of edges appearing in a 2-D or 3-D mesh is an important quantity even if the edge data structure is not used for machine computation. For certain discretizations involving only adjacent neighbors we will show that the number of nonzero entries of the Jacobian matrix associated with the discretization can be precisely related to the number of vertices and edges of a mesh.

Edge-based Gradient and Divergence Formulas

As a first example, we will derive an edge formula for the integral averaged gradient of a function u, $\int \nabla u \; da$, for the the region Ω_0 described by the union of all triangles which share the vertex v_0, see figure 1.0.

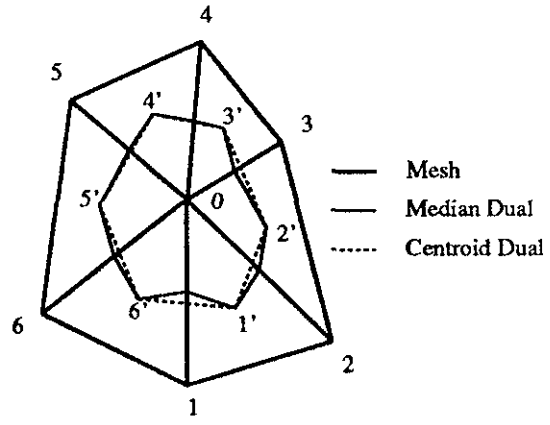

Figure 1.0 Local Mesh with centroid and median duals.

If the discrete solution u^h varies linearly in each triangle T then the gradient is constant and the integration exact.

$$\int_{\Omega_0} \nabla u^h \; da = \sum_{T \in \Omega_0} (\nabla u^h)_T A_T \qquad (1.0)$$

Equation (1.0) would suggest computing the gradient in each triangle sharing v_0 and accumulating the area weighted sum. If integral averaged gradients are required at all vertices then the gradient in each triangle could be computed and the area weighted result scattered to the three vertices of the triangle for accumulation. We refer to this as the element-by-element approach. A Green's formula would suggest a different approach for the same task.

$$\int_{\Omega_0} \nabla u \; da = \oint_{\partial\Omega_0} u\,\mathbf{n}\; dl \qquad (1.1)$$

Identical results are obtained by approximating the right-hand-side of (1.1) by trapezoidal quadrature (exact for piecewise linear u^h)

$$\oint_{\partial\Omega_0} u^h\,\mathbf{n}\; dl = \sum_{i \in \mathcal{I}_0} \frac{1}{2}(u_i^h + u_{i+1}^h)\vec{\mathbf{n}}_{i+1/2}, \;\; \mathcal{I}_0 = \{1, 2, ..., 6\} \qquad (1.2)$$

where $\vec{\mathbf{n}}_{i+1/2}$ is the vector perpendicular to the edge $e(v_i, v_{i+1})$ with magnitude equal to the length of the edge. The summation can be rearranged to yield

$$\oint_{\partial\Omega_0} u^h\,\mathbf{n}\; dl = \sum_{i \in \mathcal{I}_0} \frac{1}{2} u_i^h (\vec{\mathbf{n}}_{i+1/2} + \vec{\mathbf{n}}_{i-1/2}). \qquad (1.3)$$

A constant solution can be added to (1.3) since the gradient of a constant function is exactly zero in this discretization. In particular, we add the value of u^h at vertex v_0.

$$\oint_{\partial\Omega_0} u^h\,\mathbf{n}\; dl = \sum_{i \in \mathcal{I}_0} \frac{1}{2}(u_0^h + u_i^h)(\vec{\mathbf{n}}_{i+1/2} + \vec{\mathbf{n}}_{i-1/2}) \qquad (1.4)$$

For any closed curve we have that $\oint \mathbf{n}\, dl = \oint d\vec{\mathbf{n}} = 0$ which implies that

$$\vec{\mathbf{n}}_{i+1/2} + \vec{\mathbf{n}}_{i-1/2} = \int_{v_{i-1}}^{v_{i+1}} d\vec{\mathbf{n}}$$

for *any* path connecting v_{i-1} and v_{i+1}. One can further show that the path integral represents a vector which is parallel in direction and three times the magnitude of the vector $\vec{\mathbf{n}}$ obtained by computing the integral for any simple path connecting the centroids of the two triangles which share the edge $e(v_0, v_i)$. $\int_{v_{i-1}}^{v_{i+1}} d\vec{\mathbf{n}} = 3\int_{v_{i'-1}}^{v_{i'}} d\vec{\mathbf{n}} = 3\vec{\mathbf{n}}_{0i}$. This reduces the gradient formula to the following form:

$$\oint_{\partial\Omega_0} u^h \mathbf{n}\, dl = \sum_{i\in\mathcal{I}_0} \frac{3}{2}(u_0^h + u_i^h)\vec{\mathbf{n}}_{0i} \qquad (1.5)$$

The vertex lumped average gradient at vertex is then given by

$$(\nabla u^h)_{v_0} = \frac{3}{A_{\Omega_0}} \sum_{i\in\mathcal{I}_0} \frac{1}{2}(u_0^h + u_i^h)\vec{\mathbf{n}}_{0i}. \qquad (1.6)$$

It is well known that the region bounded by the "median" dual at vertex v_0, (shown in figure 1.0) has an area A_0 which is exactly $\frac{1}{3}A_{\Omega_0}$. Therefore using the median dual we obtain a formula which appears to represent some approximate quadrature of the right-hand-side of (1.1) on nonoverlapping regions.

$$(\nabla u^h)_{v_0} = \frac{1}{A_0} \sum_{i\in\mathcal{I}_0} \frac{1}{2}(u_0^h + u_i^h)\vec{\mathbf{n}}_{0i} \qquad (1.7a)$$

A naive interpretation of equation (1.7) would probably conclude that (1.7a) is a rather poor approximation to (1.1). It is not obvious from (1.7a) that the gradient of a linear function u is computed exactly. From the origin of this formula we now know that this formula can be obtained from a trapezoidal quadrature on a slightly larger region and is exact within the class of linear polynomials.

Keep in mind that a constant solution could have been subtracted instead of added from equation (1.3) which would have given a different but equivalent form of (1.7a).

$$(\nabla u^h)_{v_0} = \frac{1}{A_0} \sum_{i\in\mathcal{I}_0} \frac{1}{2}(u_i^h - u_0^h)\vec{\mathbf{n}}_{0i} \qquad (1.7b)$$

This formula does not appear to resemble any approximate quadrature of (1.1).

Equation (1.7a) suggests an algorithm using an edge data structure which is quite different from the element-by-element method (1.0). The edge-based calculation consists of the following steps:

Sample Gradient Computation

(1) (Precomputation) For each edge $e(v_i, v_j)$ *gather* the centroid coordinates of the two adjacent cells, v_i' and v_j'.

(2) (Precomputation) For each edge *compute* the dual edge normal $\vec{\mathbf{n}}_{ij}$ from the centroid coordinates $\vec{\mathbf{n}}_{ij} = \int_{v_i'}^{v_j'} d\vec{\mathbf{n}}$. (Orient from v_i to v_j if $i < j$).

(3) For each edge $e(v_i, v_j)$ *gather* the values of the function at the two vertices, u_i^h and u_j^h.

(4) For each edge *compute* the arithmetic average and multiply by the dual edge normal, $\frac{1}{2}(u_i + u_j)\mathbf{n}_{ij}$.

(5) For each edge *scatter and accumulate* the result at vertex v_i.

(6) For each edge *negate, scatter and accumulate* the same result at vertex v_j.

(7) For each vertex *compute* the final gradient by dividing the accumulated result by area of the median dual, A_0.

This algorithm conforms perfectly within the edge data structure. In practice all the geometrical factors could be precomputed and stored in memory by edge thereby eliminating a gather. The sample algorithm described above serves as a template for all the remaining algorithms described in the rest of this section.

The gradient and divergence operators are related so that it is not surprising that the discretization of the divergence operator produces a similar formula (see also Roe [15]):

$$\int_{\Omega_0} div(\mathbf{F}^h) da = \sum_{i\in\mathcal{I}_0} \frac{3}{2}(\mathbf{F}_0^h + \mathbf{F}_i^h)\cdot\vec{\mathbf{n}}_{0i} \qquad (1.8)$$

The Galerkin weighted finite element integrals with linear elements produce essentially identical results. In this case a piecewise linear weighting function ϕ^h is introduced (see figure 1.1). The gradient and divergence formulas

$$\int_{\Omega_0} \phi^h \nabla u^h da = \sum_{i\in\mathcal{I}_0} \frac{1}{2}(u_0^h + u_i^h)\vec{\mathbf{n}}_{0i} \qquad (1.9)$$

and

$$\int_{\Omega_0} \phi^h\, div(\mathbf{F}^h) da = \sum_{i\in\mathcal{I}_0} \frac{1}{2}(\mathbf{F}_0^h + \mathbf{F}_i^h)\cdot\vec{\mathbf{n}}_{0i} \qquad (1.10)$$

differ from the previous formulas by a constant factor of 1/3. If a lumped approximation to the left-hand-side of (1.9) is assumed, (1.7) is recovered since

$$\int_{\Omega_0} \phi^h \nabla u^h da \approx (\nabla u^h)_{v_0} A_0.$$

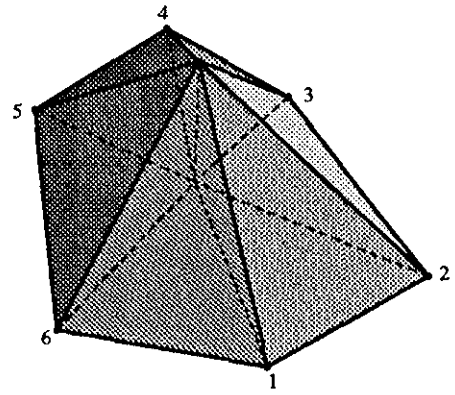

Figure 1.1 Global weighting function for vertex v_0 (not labeled).

Edge-based Hessian and Laplacian Formulas

Similar edge formulas for Hessian and Laplacian operators are possible. In Appendix A, we derive the edge formula for the Hessian-like matrix of second derivatives

$$\nabla\mu(\nabla u)^T = \begin{bmatrix} (\mu u_x)_x & (\mu u_y)_x \\ (\mu u_x)_y & (\mu u_y)_y \end{bmatrix} \tag{2.1}$$

Values of these derivatives at vertices of the mesh can be approximated as vertex lumped approximations to the Galerkin weighted integral.

$$\int_{\Omega_0} \phi \nabla\mu(\nabla u)^T da \approx \nabla\mu(\nabla u)^T A_0 \tag{2.2}$$

The reader is referred to Appendix A for details of the general formulas. It is useful to consider some model equations obtainable from these general equations. The simplest is the Laplacian operator $\nabla^2 u = Trace(\nabla\mu(\nabla u)^T)$ with unit μ. The canonical edge formula for the discrete Laplacian operator is

$$(\nabla^2 u^h)_{v_0} = \frac{1}{A_0} \sum_{i \in \mathcal{I}_0} \frac{1}{2}[\text{cotan}(\alpha_{L_i}) + \text{cotan}(\alpha_{R_i})](u_i - u_0) \tag{2.3}$$

where the angles α_{L_i} and α_{R_i} are depicted below.

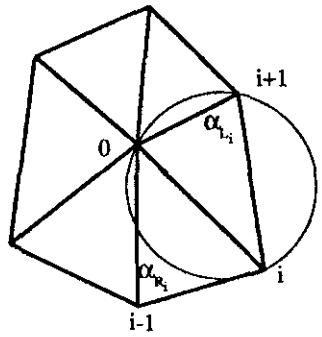

Figure 1.2. Circumcircle test for adjacent triangles.

Although equation (2.3) is not compatible with the edge data structure, equation (A.13) appearing in Appendix A gives an alternate form which is only slightly more complicated and completely compatible with the edge data structure.

When considering elliptic equations, a property of theoretical importance in proving stability and uniform convergence of discrete approximations is the existence of a discrete maximum principle. It is well known that a discrete maximum principle exists for arbitrary point distributions and boundary data if and only if the discrete operator is a nonnegative operator, i.e., if

$$\mathcal{L}(u^h)_{v_0} = \sum_{i \in \mathcal{I}_0} w_i u_i^h \tag{2.4}$$

and

$$w_0 < 0,\ w_i \geq 0, i > 0,\ w_0 + \sum_{i \in \mathcal{I}_0} w_i = 0 \tag{2.5}$$

for any interior vertex v_0. For schemes of the form

$$\mathcal{L}(u^h)_{v_0} = \sum_{i \in \mathcal{I}_0} W_i(u_i^h - u_0^h) \tag{2.6}$$

nonnegativity requires that $W_i \geq 0$ for all $i \in \mathcal{I}_0$. This guarantees a maximum principle. Equating equation (2.6) to zero, we obtain

$$u_0^h = \frac{\sum_{i \in \mathcal{I}_0} W_i u_i^h}{\sum_{i \in \mathcal{I}_0} W_i} \tag{2.7}$$

and therefore

$$\min(u_1^h, u_2^h, ..., u_{d(v_0)}^h) \leq u_0^h \leq \max(u_1^h, u_2^h, ..., u_{d(v_0)}^h)$$

A natural question to be addressed concerns the existence and uniqueness (or nonuniqueness) of triangulations of an arbitrary point set such that (2.3) guarantees a discrete maximum principle. In two dimensions a unique triangulation always exists. The main result is summarized in the following theorem†:

The discrete Laplacian operator (2.3) exhibits a discrete maximum principle for arbitrary point sets in two space dimensions iff the triangulation of these points is a Delaunay triangulation.

† It has been recently brought to the attention of the author that this result has been independently published by M.D. Reese, Oxford Computing Laboratory, NAG Report 88/2, 1988.

The key elements of the proof are given below:
Rearrangement of the weights appearing in (2.3) yields

$$\begin{aligned} W_i &= \frac{1}{2}\left[\text{cotan}(\alpha_{L_i}) + \text{cotan}(\alpha_{R_i})\right] \\ &= \frac{1}{2}\left[\frac{\cos(\alpha_{L_i})}{\sin(\alpha_{L_i})} + \frac{\cos(\alpha_{R_i})}{\sin(\alpha_{R_i})}\right] \\ &= \frac{1}{2}\left[\frac{\sin(\alpha_{L_i}+\alpha_{R_i})}{\sin(\alpha_{L_i})\sin(\alpha_{R_i})}\right] \end{aligned} \tag{2.8}$$

Since $\alpha_{L_i} < \pi$, $\alpha_{R_i} < \pi$, the denominator is always *positive* and *nonnegativity requires that* $\alpha_{L_i} + \alpha_{R_i} \leq \pi$. Some trigonometry reveals that for the configuration of figure 1.2 with circumcircle passing through $\{v_0, v_i, v_{i+1}\}$ the sum $\alpha_{R_i} + \alpha_{L_i}$ depends on the location of v_{i-1} with respect to the circumcircle in the following way:

$$\begin{aligned} &\alpha_{R_i} + \alpha_{L_i} < \pi, \quad v_{i-1} \text{ exterior} \\ &\alpha_{R_i} + \alpha_{L_i} > \pi, \quad v_{i-1} \text{ interior} \\ &\alpha_{R_i} + \alpha_{L_i} = \pi, \quad v_{i-1} \text{ cocircular} \end{aligned} \tag{2.9}$$

Also note that we could have considered the circumcircle passing through $\{v_0, v_i, v_{i-1}\}$ with similar results for v_{i+1}. The condition of nonnegativity implies a circumcircle condition for all pairs of adjacent triangles whereby the circumcircle passing through either triangle cannot contain the fourth point. This is precisely the unique characterization of the Delaunay triangulation which would complete the proof.

For a regular triangulation ($\alpha_L = \alpha_R = 60°$) we have

$$(\nabla^2 u)_{v_0} A_0 = \frac{1}{2\sqrt{3}} \sum_i (u_i - u_0) \text{ (regular triangulation)} \tag{2.10}$$

Equation (2.10) is essentially the smoothing formula used by Jameson [6] for triangular meshes. For irregular meshes, this form of smoothing fails to vanish when the solution varies linearly over the support of the discretization which is a shortcoming of the approximation. The previous analysis would suggest a smoothing term of the form

$$(\mathcal{D}u)_{v_0} = \frac{1}{2}\sum_i \max\left(\epsilon, \frac{\sin(\alpha_{L_i}+\alpha_{R_i})}{\sin(\alpha_{L_i})\sin(\alpha_{R_i})}\right)(u_i - u_0) \tag{2.11}$$

for some nonnegative ϵ. Note that this formula is actually conservative owing to the symmetry of the weighting term.

Keep in mind that from equation (2.3) we have that $\text{cotan}(\alpha) \geq 0$ if $\alpha \leq \pi/2$. Therefore a sufficient but not necessary condition for nonnegativity (and a Delaunay triangulation) is that all angles of the mesh be less than or equal to $\pi/2$. This is a standard result in finite element theory and applies in two or more space dimensions. The construction of nonnegative operators has important implications with respect to the diagonal dominance of implicit schemes, the eigenvalue spectrum of the discrete operator, stability of relaxation schemes, etc. These are all classic results in numerical analysis and not within the scope of the present paper.

We can ask if the result concerning Delaunay triangulation and the maximum principle extends to three space dimensions. As we will show, the answer is no. Appendix A gives the corresponding edge formulas for Hessian and Laplacian discretizations in 3-D. The resulting formula for the three dimensional Laplacian is

$$\int_{V_{\Gamma_0}} \phi^h \nabla^2 u^h \, dv = \sum_{i \in \mathcal{I}_0} W_i (u_i - u_0) \tag{2.12}$$

where

$$W_i = \frac{1}{6} \sum_{k=1}^{d(v_0,v_i)} |\Delta \mathbf{R}_{k+1/2}| \, \text{cotan}(\alpha_{k+1/2}). \tag{2.13}$$

In this formula V_{Γ_0} is the volume formed by the union of all tetrahedra that share vertex v_0. $\mathcal{I}_0$ is the set of indices of all adjacent neighbors of v_0 connected by incident edges, k a local cyclic index describing the associated vertices which form a polygon of degree $d(v_0, v_i)$ surrounding the edge $e(v_0, v_i)$, $\alpha_{k+1/2}$ is the face angle between the two faces associated with $\vec{\mathbf{S}}_{k+1/2}$ and $\vec{\mathbf{S}}'_{k+1/2}$ which share the edge $e(v_k, v_{k+1})$ and $|\Delta \mathbf{R}_{k+1/2}|$ is the magnitude of the edge (see figure below).

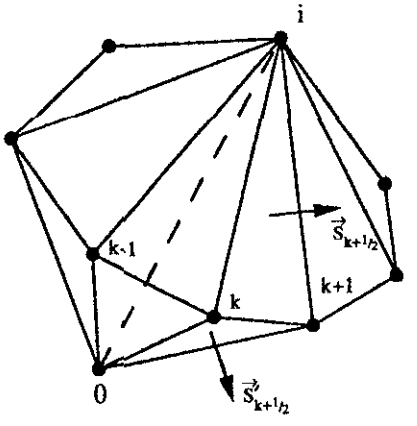

Figure 1.3. Set of tetrahedra sharing edge $e(v_0, v_i)$ with local cyclic index k.

A maximum principle is guaranteed if all $W_i \geq 0$. We now will procede to describe a valid Delaunay triangulation with one or more $W_i < 0$. It will suffice to consider the Delaunay triangulation of N points in which

a single point v_0 lies interior to the triangulation and the remaining $N-1$ points describe vertices of boundary faces which completely cover the convex hull of the point set.

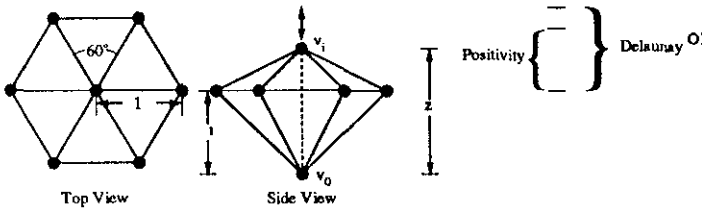

Figure 1.4. Subset of 3-D Delaunay Triangulation which does not maintain nonnegativity.

Consider a subset of the N vertices, in particular consider an interior edge incident to v_0 connecting to v_i as shown in figure 1.4 by the dashed line segment and all neighbors adjacent to v_i on the hull of the point set. In this experiment we consider the height of the interior edge, z, as a free parameter. Although it will not be proven here, the remaining $N-8$ points can be placed without conflicting with any of the conclusions obtained for looking at the subset.

It is known that a necessary and sufficient condition for the 3-D Delaunay triangulation is that the circumsphere passing through the vertices of any tetrahedron must be point free [16]; that is to say that no other point of the triangulation can lie interior to this sphere. Furthermore a property of locality exists [16] so that we need only inspect adjacent tetrahedra for the satisfaction of the circumsphere test. For the configuration of points shown in figure 1.4, convexity of the point cloud constrains $z \geq 1$ and the satisfaction of the circumsphere test requires that $z \leq 2$.

$$1 \leq z \leq 2 \quad \text{(Delaunay Triangulation)}$$

From (2.13) we find that $W_i \geq 0$ if and only if $z < 7/4$.

$$1 \leq z \leq \frac{7}{4}, \quad \text{(Nonnegativity)}$$

This indicates that for $7/4 < z \leq 2$ we have a valid Delaunay triangulation which (2.13) does not satisfy a discrete maximum principle. In fact, the Delaunay triangulation of 400 points randomly distributed in the unit cube revealed that approximately 25% of the interior edge weights were of the wrong sign (negative).

Keep in mind that from (2.13) we can obtain a sufficient but not necessary condition for nonnegativity that all face angles be less than or equal to $\pi/2$.

The formulas for the prototypical viscous term $\nabla \cdot \mu \nabla u$ are only slightly more complicated than the Laplacian formulas. In 2-D we have the following weights

$$W_i = \frac{1}{2}\left[\overline{\mu}_{L_i}\text{cotan}(\alpha_{L_i}) + \overline{\mu}_{R_i}\text{cotan}(\alpha_{R_i})\right] \qquad (2.14)$$

or in 3-D

$$W_i = \frac{1}{6}\sum_{k=1}^{d(v_0,v_i)} \overline{\mu}_{k+1/2}|\Delta \mathbf{R}_{k+1/2}|\ \text{cotan}(\alpha_{k+1/2}) \qquad (2.15)$$

where $\overline{\mu}$ is the average value of μ in the specified simplex. Since μ and $\overline{\mu}$ are always assumed positive quantities, we have the following theorem:

A discrete maximum principle associated with the discretization of $\nabla \cdot \mu \nabla u$ with weights given by (2.14) and (2.15) is guaranteed iff $W_i \geq 0$ for all interior edges of the mesh. A sufficient but not necessary condition is that all angles (2-D) or faces angles (3-D) be less than or equal to $\pi/2$.

The proof follows immediately from nonnegativity of (2.14) and (2.15). The sufficient but not necessary condition is a minor extension of the result by Ciarlet [12]. These weight formulas will be considered again in the next section when discussing data dependent edge swapping algorithms.

§ Numerical Calculation of Viscous Flows on Unstructured Meshes

The calculation of viscous flow at high Reynolds numbers introduces new difficulties not normally encountered in inviscid or low Reynolds number calculations. In the following paragraphs practical issues associated with computing viscous Navier-Stokes flow will be discussed.

Viscous Mesh Generation

The analysis of the Laplacian operator clearly indicates the special role of Delaunay triangulation with respect to discretization of the Laplacian operator using standard approximations. Unfortunately, this type of analysis does not address aspects related to the accuracy or convergence of the approximation. From an accuracy point-of-view the Delaunay triangulation is not always the method of choice.

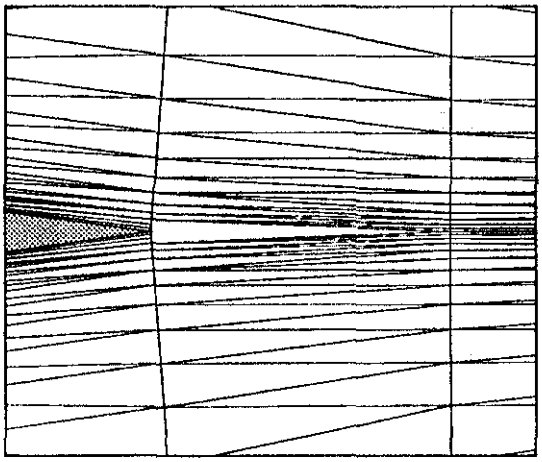

Figure 2.1 Delaunay triangulation near trailing edge of airfoil.

A graphic example is shown in figure 2.1 for the Delaunay triangulation of points surrounding the trailing edge regions of a typical airfoil (points generated from a structured mesh). Upon first inspection, the triangulation appears flawed. An extreme closeup of the trailing edge region (figure 2.2) reveals that the mesh is actually a mathematically correct Delaunay triangulation.

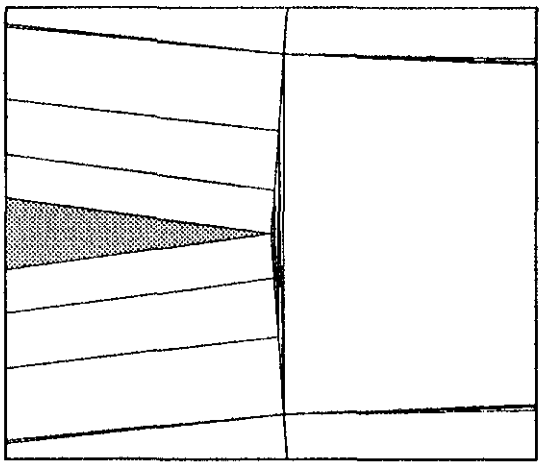

Figure 2.2 Extreme closeup of Delaunay triangulation near trailing edge.

The presence of nearly collapsed triangles near the trailing edge followed by relatively large triangles leaves considerable doubt as to the accuracy of numerical approximations in the trailing edge region. One way to retain the same distribution of points but remove the collapsed triangles is to incorporate directional preferentiality into the Delaunay triangulation [17]. Directional preferentiality can be accomplished by locally mapping points into a nonphysical plane and performing the Delaunay triangulation in this plane. The local mapping from physical to nonphysical planes is accomplished by specification of nonisotropic stretchings along a prespecified set of axes. If the stretching characteristics are chosen properly, the resulting triangulation would connect the trailing edge points downstream thereby eliminating the collapsed triangles. Unfortunately, the choice of stretching characteristics is ad hoc and usually obtained by exploiting the fact that the point set was originally derived from a structured body-fitted mesh with known stretching characteristics.

Other departures from Delaunay triangulation are possible which would eliminate these collapsed cells. One effective technique, more in the spirit of unstructured meshes, is the generation of a triangulation which minimizes the sum of the maximum angles in all triangles (the "Min-Max" triangulation). This technique has a foundation in theory; Babuška and Aziz [18] show that within the framework of the finite element method, the essential property of a triangle is that no angle be too close to 180°. In other words, high aspect ratio triangles with one small angle are acceptable, triangles with two small angles are not.

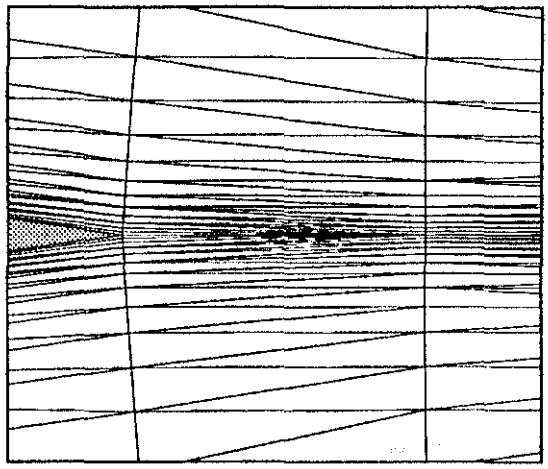

Figure 2.3 Min-Max triangulation near trailing edge of airfoil.

One simple way to generate a Min-Max triangulation is to start with some existing triangulation of the point set (possibly Delaunay) then to generate a Min-Max triangulation via a variant of Lawson's edge swapping algorithm [19]. The algorithm considers pairs of adjacent triangles and swaps the diagonal edge if it decreases the maximum angle of the triangle pair.

Iterative Algorithm: Min-Max Triangulation

```
swapedge = true
While(swapedge)do
  swapedge = false
  Do (all interior edges)
    If (adjacent triangles form convex quadrilateral)then
      Swap diagonal to form T*.
      If (Min-Max criteria locally satisfied)then
        T = T*
        swapedge = true
      EndIf
    EndIf
  EndDo
EndWhile
```

The edge swapping algorithm has been proven to terminate in a finite number of steps. In practice the computer time required to perform this task is very small. Figure 2.3 shows the resulting Min-Max triangulation of the same points as before. All collapsed cells are absent.

Modified forms of Lawson's edge swapping algorithm can be formulated which locally optimize other quantities. For example, consider the edge weights given in (2.14) for the prototype viscous term $\nabla \cdot \mu \nabla u$.

$$W_i = \frac{1}{2}\left[\overline{\mu}_{L_i}\text{cotan}(\alpha_{L_i}) + \overline{\mu}_{R_i}\text{cotan}(\alpha_{R_i})\right]$$

From these weights we can seek a data dependent triangulation that maximizes the minimum weight, W_i, via local edge swapping. Data dependent triangulations of this type are discussed in [20,21]

Contrary to common belief, edge swapping algorithms are possible in 3-D [22]. These algorithms exploit the property that in an n-dimensional space $n+2$ points can be triangulated in at most two ways (subject to certain technical conditions) [16]. Preliminary algorithms based on 3-D edge swapping ideas have been constructed.

Implicit Navier-Stokes Solver

The development of efficient and robust solution strategies for solving the Navier-Stokes equations on unstructured meshes is probably one of the most difficult aspects facing algorithm developers. In this section we will describe a rather brute-force implicit algorithm for the Navier-Stokes equations. The method requires the solution of a large sparse system of linear equations. The linear system arises from the use of a backward Euler time integration scheme. Rapid convergence to steady-state can be obtained by the use of large time steps. To reduce the computational effort in the sparse matrix solution, the true linear system is replace by approximate linear system obtained from the linearization of a lower order accurate scheme for the inviscid Euler terms. The advantage of this approximate linearization is the significant reduction in number of nonzero entries appearing in the sparse linear system. In fact, the approximate linearization for any vertex of the mesh only involves adjacent neighbors of the mesh. This permits a precise calculation of the number of nonzeros of the linear system (but not its inverse!).

We begin with the Navier-Stokes equations in divergence form

$$\mathbf{u}_t + div(\mathbf{F}) = div(\mathbf{G}) \tag{3.0}$$

where $\mathbf{u}$ represents the vector of conserved variables, $\mathbf{F}$ the inviscid Euler flux vector,

$$\mathbf{u} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix} \hat{\mathbf{i}} + \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix} \hat{\mathbf{j}}$$

and $\mathbf{G}$ the viscous flux vector.

$$\mathbf{G} = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ u\tau_{xx} + v\tau_{xy} - q_x \end{pmatrix} \hat{\mathbf{i}} + \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{yx} + v\tau_{yy} - q_y \end{pmatrix} \hat{\mathbf{j}}$$

$$\begin{aligned} \nabla q &= -\kappa \nabla T \\ \boldsymbol{\tau} &= \lambda(u_x + v_y)I + \mu \begin{bmatrix} 2u_x & u_y + v_x \\ v_x + u_y & 2v_y \end{bmatrix} \\ p &= (\gamma - 1)\left(E - \frac{1}{2}\rho(u^2 + v^2)\right) = \rho RT \end{aligned}$$

In these equations λ and μ are diffusion coefficients, κ the coefficient of thermal conductivity, γ the ratio of specific heats, and R the ideal gas law constant.

Using a Galerkin weighted integral form of equation (3.1), we obtain

$$\int_{\Omega_0} \phi^h \left(\mathbf{u}_t^h + div(\mathbf{F}^h) - div(\mathbf{G}^h)\right)\, da = 0. \tag{3.1}$$

Recall from section 2 that the discretization of divergence terms results in the following formulas:

$$\begin{aligned} \int_{\Omega_0} \phi^h div(\mathbf{F}^h)\, da &= \sum_{i \in \mathcal{I}_0} \frac{1}{2}(\mathbf{F}_0^h + \mathbf{F}_i^h) \cdot \vec{\mathbf{n}}_{0i} \\ \int_{\Omega_0} \phi^h div(\mathbf{G}^h)\, da &= \sum_{i \in \mathcal{I}_0} \mathbf{M}_i(\mathbf{u}_i - \mathbf{u}_0) \end{aligned} \tag{3.2}$$

where $\mathbf{M}_i$ is calculated from (A.13) in Appendix A.

Upwinded convective formulas can be derived by modifying the basic Galerkin discretization formula for

the $div(\mathbf{F})$. The simplest upwind scheme is obtained by adding an additional term to the Galerkin formula:

$$\int_{\Omega_0} \phi^h div(\mathbf{F}^h)\, da = \sum_{i \in \mathcal{I}_0} \frac{1}{2}(\mathbf{F}_0^h + \mathbf{F}_i^h)\cdot \vec{\mathbf{n}}_{0i} - \frac{1}{2}|A(\mathbf{u}_0, \mathbf{u}_i; \vec{\mathbf{n}}_{0i})|(\mathbf{u}_i - \mathbf{u}_0) \tag{3.3}$$

where $|A(\mathbf{u}_0, \mathbf{u}_i; \vec{\mathbf{n}}_{0i})|$ is a positive semi-definite matrix formed from the Jacobian of $\mathbf{F}$, $A = \mathbf{dF}(\overline{\mathbf{u}})$, evaluated at an intermediate state $\overline{\mathbf{u}} = \overline{\mathbf{u}}(\mathbf{u}_0, \mathbf{u}_i)$, see reference [23]. The additional term is motivated from finite volume considerations and corresponds the simplest upwind finite volume scheme with piecewise constant distributions in control volumes defined by a median dual.

In reference [1] higher order accurate upwind finite volume schemes are constructed which assume discontinuous piecewise linear distributions in each control volume. Similar schemes can be derived as modified forms of Galerkin discretizations. If gradients are calculated at all vertices of the mesh

$$(\nabla \mathbf{u})_{v_0} = \frac{1}{A_0} \sum_{i \in \mathcal{I}_0} \frac{1}{2}(\mathbf{u}_0 + \mathbf{u}_i)\vec{\mathbf{n}}_{0i} \tag{3.4}$$

then a solution state at a point midway between the vertices v_0 and v_i can be extrapolated from either of the following two formulas:

$$\begin{aligned} \mathbf{u}_0^+ &= \mathbf{u}_0 + (\nabla \mathbf{u})_{v_0}\cdot \Delta \mathbf{r}_i \\ \mathbf{u}_i^- &= \mathbf{u}_i - (\nabla \mathbf{u})_{v_i}\cdot \Delta \mathbf{r}_i \end{aligned} \tag{3.5}$$

with $\Delta \mathbf{r}_i = \frac{1}{2}(\mathbf{r}_i - \mathbf{r}_0)$, $\mathbf{r} = (x, y)^T$. Note that for smooth solutions we expect $|u_i - u_0| = O(|\Delta \mathbf{r}_i|)$ and $|u_i^- - u_0^+| = O(|\Delta \mathbf{r}_i|^2)$ since $u_i^- - u_0^+ = 0$ when u varies linearly over the support. This motivates the construction of a higher order accurate upwind scheme of the form

$$\int_{\Omega_0} \phi^h div(\mathbf{F}^h) da = \sum_{i \in \mathcal{I}_0} \frac{1}{2}(\mathbf{F}_0^h + \mathbf{F}_i^h)\cdot \vec{\mathbf{n}}_{0i} - \frac{1}{2}|A(\mathbf{u}_0^+, \mathbf{u}_i^-; \vec{\mathbf{n}}_{0i})|(\mathbf{u}_i^- - \mathbf{u}_0^+) \tag{3.6}$$

Equation (3.6) differs from the Galerkin discretization at the level of higher order terms. For flows containing shockwaves or steep gradients, a flux limiting procedure can be applied as discussed in [1].

Denoting the spatial discretization by the residual vector $\mathbf{R}(\mathbf{u}^h)$, equation (3.1) is rewritten in the following semi-discrete form:

$$\int_{\Omega_0} \phi^h \mathbf{u}_t^h \, da = \mathbf{R}(\mathbf{u}^h). \tag{3.7}$$

For steady-state calculations, a vertex lumped approximation can be assumed for time derivatives.

$$(\mathbf{u}_t^h)_{v_0} A_0 \approx \int_{\Omega_0} \phi^h \mathbf{u}_t^h \, da = \mathbf{R}(\mathbf{u}^h) \tag{3.8}$$

A general two level family of integration schemes for solving (3.8) is given by

$$[I/\Delta t - \theta \mathbf{dR}(\mathbf{u}^h(t))]\,(\mathbf{u}(t+\Delta t) - \mathbf{u}(t)) = \mathbf{R}(\mathbf{u}(t)) \tag{3.9}$$

so that $\theta = 0$ corresponds to a forward Euler (explicit) timestep and $\theta = 1$ corresponds to the backward Euler (implicit) timestep. The backward Euler implicit scheme approaches Newton's method for $\Delta t \to \infty$. For steady-state computations, Δt is treated as a relaxation parameter and is usually a function of the level of convergence to steady-state, see Venkatakrishnan and Barth [24]. In this reference a cell-centered finite volume scheme is developed for the Euler and Navier-Stokes equations. The spatially discretized equations with backward Euler time stepping are solved using a sparse matrix solver. In the present application we use a hybrid scheme consisting of sparse matrix solves using an approximate form of the left-hand-side of (3.9) and explicit time stepping. The backward Euler time stepping with approximate Jacobian is effective in removing low frequency error components. A 4-stage Runge-Kutta scheme with coefficients chosen to optimize high frequency damping characteristics [26] is used in the explicit phase of the solution strategy. The residual vector for the scheme consists of the Galerkin discretization of viscous terms and the high order upwind discretization (3.6) for convective terms. Rather than form the true Jacobian linearization of the residual vector, the linearization of (3.6) is approximated by the linearization of the lower order accurate form (3.3). Details of the exact Jacobian linearization of (3.3) are given in [25]. The resulting Jacobian linearization at each vertex of the mesh involves adjacent neigbors of the mesh and greatly reduces the number of nonzero entries of the linear system when compared to the true Jacobian linearization. The reduction in nonzeros reduces the storage and matrix inversion costs considerably. Unfortunately, the use of the approximate Jacobian slows the rate of convergence to steady-state and destroys any hope of quadratic convergence for $\Delta t \to \infty$.

Note that because we are using an approximate Jacobian linearization involving adjacent neighbors of the mesh, we can precisely compute the storage requirement of the implicit sparse matrix (left-hand-side of (3.9)). The storage requirements for a Navier-Stokes algorithm differ from a scalar differential equation of the same form as equation (3.0) by a factor of $4 \cdot 4$ (2-D) or $5 \cdot 5$ (3-D). For an algorithm based on a scalar differential equation, the number of nonzero entries in

the i-th row of the sparse matrix is equal to the number of adjacent neigbors to vertex v_i in addition to the vertex itself.

$$\mathcal{N}_i = 1 + d(v_i)$$

where $d(v_i)$ is the vertex degree. The total number of nonzeros in the matrix is the sum of nonzeroes in each row.

$$\mathcal{N} = \sum_{i=1}^{n(v)} \mathcal{N}_i = \sum_{i=1}^{n(v)} (1 + d(v_i)) = n(v) + \sum_{i=1}^{n(v)} d(v_i)$$

Since every edge has two vertices and every vertex v_i has $d(v_i)$ incident edges, the following equality exists.

$$\sum_{i=1}^{n(v)} d(v_i) = 2n(e)$$

So without approximation, the number of nonzero entries of the sparse matrix is exactly

$$\mathcal{N} = n(v) + 2n(e).$$

Note that this result holds in three dimensions as well. When combined with the formula of section 2 relating edges to vertices of triangular meshes in 2-D, we have the following exact formula in 2-D for the number of nonzeros appearing in the sparse linear system:

$$\mathcal{N} = 7n(v) + 6(n(c) - 2) - 2n(e)_{boundary}.$$

Formulas of this type are not particularly useful for the present solution strategy because they do not predict the storage required for actually solving the linear system. The previous formula is useful when considering solution methods utilizing approximate inverses such as variants of incomplete $L - U$ factorization which retain the same sparsity in the approximate L and U factors as appears in the original matrix.

In actual computations, we find that a hybrid strategy consisting of a sparse matrix solve followed by $20 - 50$ explicit 4-stage Runge-Kutta timesteps gives satisfactory convergence to steady-state. Some sample Navier-Stokes calculations with timings and memory requirements are given in a later section. In future work, the sparse matrix solver will be replaced by a preconditioned minimum residual method.

High Reynolds Number Flows

Simulating the effects of turbulence on unstructured meshes via the Reynolds-averaged Navier-Stokes equations and turbulence modeling is a relatively unexplored topic. In early work by Rostand [27], an algebraic turbulence model was incorporated into an unstructured mesh flow solver. This basic methodology was later refined by Mavriplis [17] for the flow about multi-element airfoil configurations. Both of these implementations utilize locally structured meshes to produce one-dimensional-like boundary-layer profiles from which algebraic models can determine an eddy viscosity coefficient for use in the Reynolds-averaged Navier-Stokes equations. The use of local structured meshes limits the general applicability of the method.

The next level of turbulence modeling involves the solution of one or more auxiliary differential equations. Ideally these differential equations would only require initial data and boundary conditions in the same fashion as the Reynolds-averaged mean equations. The use of turbulence models based on differential equations greatly increases the class of geometries that can be treated "automatically." Unfortunately this does not make the issue of turbulence modeling a "solved" problem since most turbulence models do not perform well across the broad range of flow regimes usually generated by complex geometries. Also keep in mind that most turbulence models for wall-bounded flow require knowledge of distance to the wall for use in "damping functions" which simulate the damping effect of solid walls on turbulence. The distance required in these models is measured in "wall units" which means that physical distance from the wall y is scaled by the local wall shear, density, and viscosity.

$$y^+ = \sqrt{\frac{\tau_{wall}}{\rho_{wall}}} \frac{y}{\nu} \tag{3.1}$$

Scaling by wall quantities is yet another complication but does not create serious implementation difficulties for unstructured meshes as we will demonstrate shortly.

In a recent report with Baldwin [28] we proposed and tested (on structured meshes) a single equation turbulence transport model. In this report, the model was tested on various subsonic and transonic flow problems: flat plates, airfoils, wakes, etc. The model consists of a single advection-diffusion equation with source term for a field variable which is the product of turbulence Reynolds number and kinematic viscosity, $\nu\tilde{R}_T$. This variable is proportional to the eddy viscosity except very near a solid wall (complete details and definitions are in Appendix B). The model equation appearing in Appendix B is

$$\begin{aligned}\frac{D(\nu\tilde{R}_T)}{Dt} =& (c_{\epsilon_2} f_2(y^+) - c_{\epsilon_1})\sqrt{\nu\tilde{R}_T P} \\ &+ (\nu + \frac{\nu_t}{\sigma_R})\nabla^2(\nu\tilde{R}_T) - \frac{1}{\sigma_\epsilon}(\nabla\nu_t)\cdot\nabla(\nu\tilde{R}_T).\end{aligned} \tag{3.2}$$

In this equation P is the production of turbulent kinetic energy and is related to the mean flow velocity rate-of-strain and the kinematic eddy viscosity ν_t.

Equation (3.2) depends on distance to solid walls in two ways. First, the damping function f_2 appearing in equation (3.2) depends directly on distance to the wall (in wall units). Secondly, ν_t depends on $\nu\tilde{R}_t$ and damping functions which require distance to the wall.

$$\nu_t = c_\mu D_1(y^+)D_2(y^+)\nu\tilde{R}_t$$

It is important to realize that the damping functions f_2, D_1, and D_2 deviate from unity only when very near a solid wall. For a typical turbulent boundary-layer (see figure 2.4) accurate distance to the wall is only required for mesh points which fall below the logarithmic region of the boundary-layer.

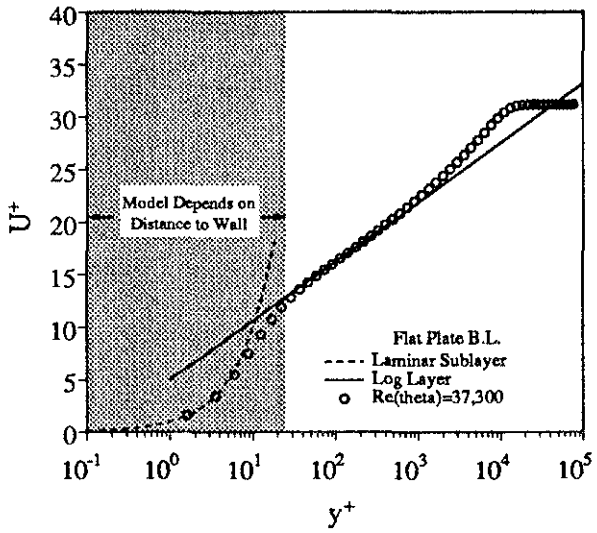

Figure 2.4. Typical flat plate boundary-layer showing dependence of turbulence model on distance to wall.

The relative insensitivity of distance to the wall means that accurate estimation of distance to the wall is only required for a small number of points that are extremely close to a boundary surface. The remaining points can be assigned any approximate value of physical distance since the damping functions are essentially at their asymptotic values. A general procedure for calculation of distance to the wall in wall units is to precompute and store, for each vertex of the mesh, the minimum distance from the vertex to the closest solid wall (examples are shown later in figures 2.6b and 2.7b). This strategy can only fail if two bodies are in such close proximity that the near wall damping functions never reach their asymptotic values. Realistically speaking, the validity of most turbulence models would be in serious question in this situation anyway. In general, the minimum distance from vertex to boundary edge does not occur at the end points of a boundary edge but rather interior to a boundary edge. For each vertex, information concerning which boundary edge achieves the minimum distance and the weight factor for linear interpolation along the boundary edge must be stored. Data can then be interpolated to the point of minimum distance whenever needed. In the course of solving (3.2), distance to a solid wall in wall units is calculated by retrieving physical distance to the wall and the local wall quantities needed for (3.1) as interpolated along the closest boundary edge.

Numerical Examples of Navier-Stokes Flow

The numerical calculations presented in this section represent a successful application of the ideas discussed in previous sections. Figures 2.6a and 2.7a show examples of Min-Max triangulations about single and multi-element airfoils. The first geometry consists of a single RAE 2822 airfoil. Navier-Stokes flow is computed about this geometry assuming turbulent flow with the following free-stream conditions: $M_\infty = .725, \alpha = 2.31°, Re = 6.5$ million. Wind tunnel experiments for the RAE 2822 geometry at these test conditions have been performed by Cook, McDonald, and Firmin [29]. The RAE 2822 airfoil mesh shown in figure 2.6a contains approxiamtely 14000 vertices and 41000 edges. The second geometry consists of a two element airfoil configuration with wind tunnel walls. The inflow conditions assume turbulent flow with $M_\infty = .09$ and $Re = 1.8$ million. Details of the geometry and wind tunnel test results can be found in the report by Adair and Horne [30]. The two element mesh shown in figure 2.7a contains approximately 18000 vertices and 55000 edges.

Both meshes were constructed in two steps. The first step was to generate a Delaunay triangulation of the point cloud. As mentioned earlier, the method of Delaunay triangulation can generate poor triangulations for highly stretched point distributions. Both meshes suffered from nearly collapsed triangles with two small interior angles. As a second step, a Min-Max triangulation was constructed by edge swapping the Delaunay triangulation. Edge swapping repaired both triangulations. Both airfoil geometries were calculated assuming turbulent flow using the one-equation turbulence model (3.2). Level sets of the generalized distance function used in the turbulence model are shown in figures 2.6b and 2.7b.

Navier-Stokes calculations on the RAE 2822 airfoil were performed using the hybrid implicit-explicit time stepping scheme with 20 explicit time steps per sparse matrix solve. Figure 2.5 shows a temporal convergence history for RAE 2822 airfoil calculation. Computations performed on the CRAY Y-MP computer required approximately 48 million words of memory and 32 CPU seconds for each iteration consisting of sparse matrix solve with 20 Runge-Kutta time steps.

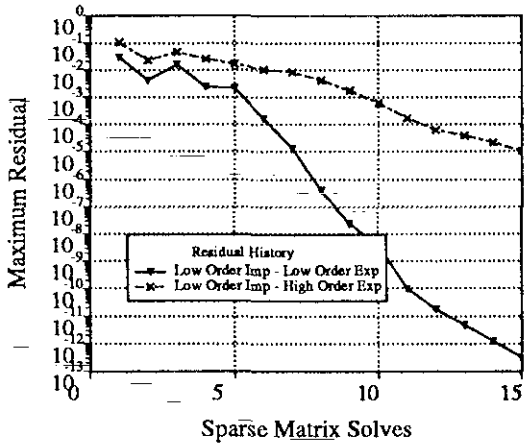

Figure 2.5. Residual history. 20 Explicit R-K 4 steps per sparse matrix solve.

In this figure we graph the convergence histories using low order accurate (3.3) or high order accurate (3.6) right-hand-side residual vectors in (3.9). As expected, the low order accurate residual vector results in rapid convergence since it more nearly matches the left-hand-side implicit operator. The convergence rate slows significantly when the high order residual vector is used because of the mismatch of implicit and explicit operators. Even so, plotting accuracy is obtained in approximately 15 sparse matrix solves. In these calculations the linearization of terms arising from the coupling of the turbulence model to the mean equation have been ignored. The impact of this coupling appears to be a secondary effect.

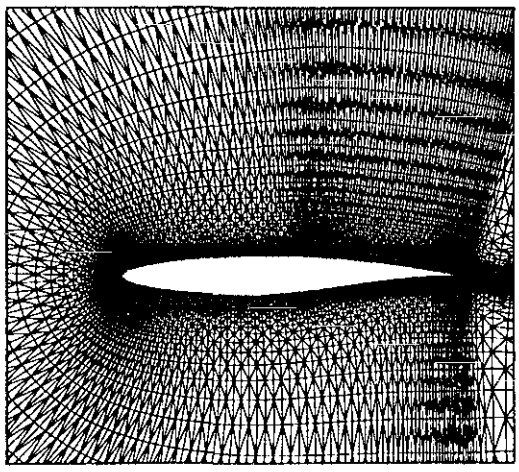

Figure 2.6a. Mesh near RAE 2822 airfoil.

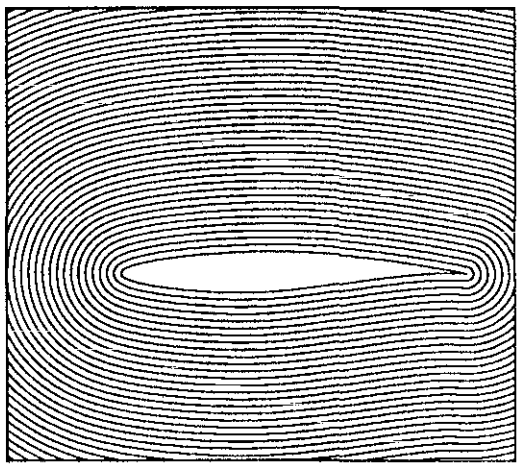

Figure 2.6b. Contours of distance function for turbulence model.

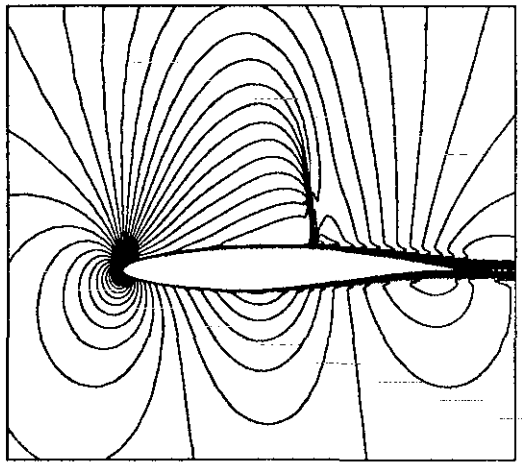

Figure 2.6c. Closeup of Mach number contours near airfoil.

Figures 2.6c-d plot Mach number contours and pressure coefficient distributions for the RAE 2822 airfoil. The pressure coefficient distribution compares favorably with the experiment of Cook, McDonald, and Firmin [29]. Leading edge trip strips were used on the experimental model but not simulated in the computations. This may explain the minor differences in the leading edge pressure distribution.

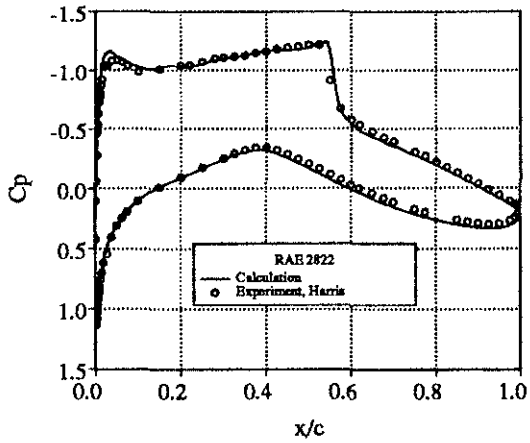

Figure 2.6d. Pressure coefficient distribution on airfoil.

Navier-Stokes computations for the two element airfoil configuration as shown in figures 2.7c-d. The effects of wind tunnel walls have been modeled in the computation by assuming an inviscid wall boundary condition. The total memory required for calculation on the two element mesh was slightly less that 64 million words. CPU time for each sparse matrix solve followed by 50 explicit time steps was approximately 50 seconds on the CRAY Y-MP. Mach number contours are shown in figure 2.7c. Observe that the contours appear very smooth, even in regions where the mesh becomes very irregular. This is due to the insistance that linear functions be accurately treated in the flow solver reguardless of mesh irregularities.

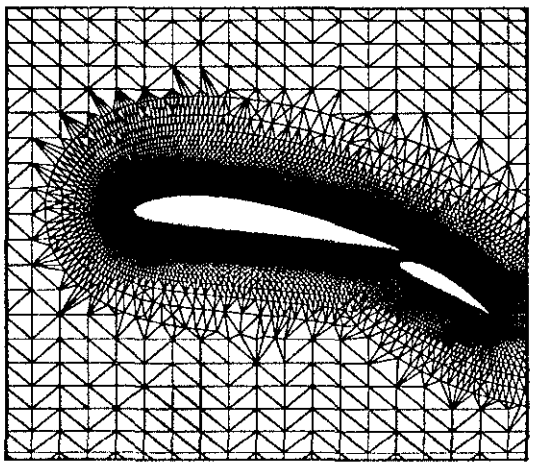

Figure 2.7a. Mesh near Multi-element airfoil.

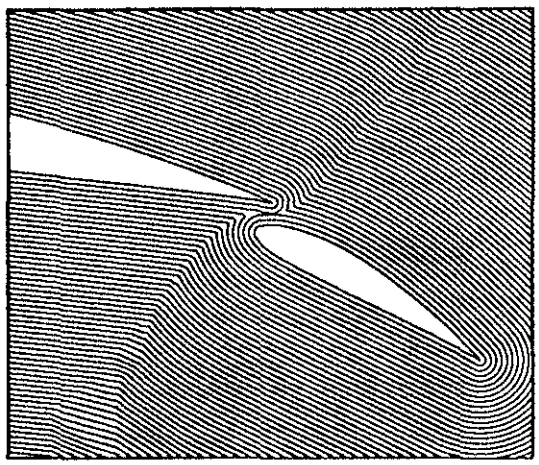

Figure 2.7b. Contours of distance function for turbulence model.

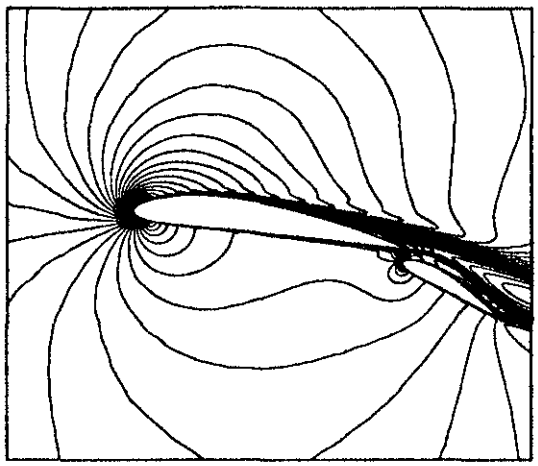

Figure 2.7c. Closeup of Mach number contours near airfoil.

Pressure coefficient distribution on the main airfoil and flap are graphed in figure 2.7d. Comparison of calculation and experiment on the main element is very good. The suction peak values of pressure coefficient on the flap element are slightly below the experiment. The experimentors also note a small separation bubble at the trailing edge of the flap which was not found in the computations.

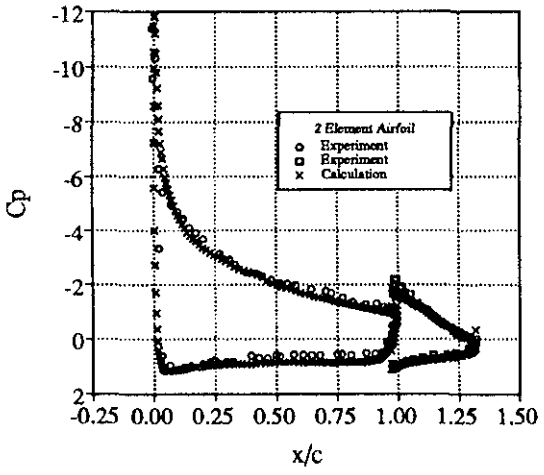

Figure 2.7d. Pressure coefficient distribution on airfoil.

§ Conclusions

The preceding analysis and sample applications have demonstrated the simplicity of the edge data structure and the corresponding numerical algorithms. A byproduct of the derivation of edge formulas is some new theory concerning a discrete maximum principle for elliptic operators. The new theory can be used to determine "optimal" meshes in the sense of a discrete maximum principle. Unfortunately, maximum principle analysis does not take into account issues associated with accuracy or convergence. Analysis which addresses these issues arrives at somewhat different conclusions concerning requirements of an "optimal" mesh. Both forms of analysis assume that the distribution of points is given and the task is to find the best triangulation. The differing requirements dictated by the two forms of analysis suggest that the ultimate mesh generation strategy should allow for either movement of points or the addition of a small number of additional points. With this flexibility, meshes can be generated which meet both requirements.

Although results for the implicit solution of turbulent flow have been presented, significant advances must be made in this area. The use of sparse matrix solvers limits computations to two space dimensions because of the large computer memory requirements. Results from conjugate gradient and minimum residual methods show some promise in this area.

§ Appendix A: 2-D and 3-D Edge Formulas for Hessian and Laplacian Discretizations

2-D Hessian and Laplacian Formualas

We begin by approximating the following matrix of second derivatives

$$\nabla\mu(\nabla u)^T = \begin{bmatrix} (\mu u_x)_x & (\mu u_y)_x \\ (\mu u_x)_y & (\mu u_y)_y \end{bmatrix} \qquad (A.1)$$

using a standard Galerkin approximation for the region Ω_0 formed from the union of all triangles that share the vertex v_0. Multiplying by the weight function ϕ and integration by parts over Ω_0 assuming $\phi = 0$ on $\partial\Omega_0$ produces the so-called weak form of the left-hand-side of (2.2)

$$\begin{aligned}\int_{\Omega_0} \phi\nabla\,\mu(\nabla u)^T\,da &= -\int_{\Omega_0} \mu(\nabla\phi)\,(\nabla u)^T\,da \\ &= -\sum_{i\in\mathcal{I}_0}\int_{T_{i+1/2}} \mu(\nabla\phi)\,(\nabla u)^T\,da\end{aligned} \qquad (A.2)$$

where $T_{i+1/2} \equiv \textit{simplex}(v_0, v_i, v_{i+1})$.

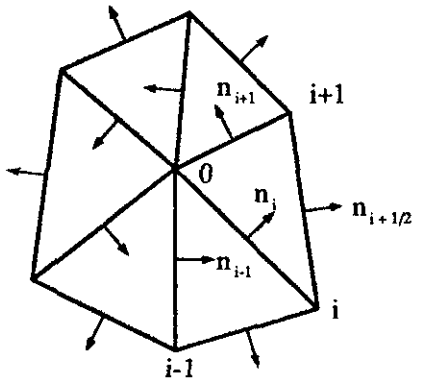

Figure A.1. Vertex v_0 and adjacent neighbors.

Using the notation of figure A.1, gradients of the piecewise linear functions ϕ^h (figure 1.1) and u^h are

$$(\nabla\phi^h)_{T_{i+1/2}} = -\frac{1}{2A_{i+1/2}}\vec{n}_{i+1/2} \qquad (A.3)$$

and

$$(\nabla u^h)_{T_{i+1/2}} = -\frac{1}{2A_{i+1/2}}(u_0^h\vec{n}_{i+1/2} + u_i^h\vec{n}_{i+1} - u_{i+1}^h\vec{n}_i) \qquad (A.4)$$

where $A_{i+1/2}$ is the area of $T_{i+1/2}$ and $\vec{n}_{i+1/2}$ is the vector normal to the edge $e(v_i, v_{i+1})$ with magnitude equal to the length of the edge.

For piecewise linear u^h the gradient is constant in each triangle. The integral average matrix of second derivatives simplifies to the following form:

$$\begin{aligned}\int_{\Omega_0}\phi^h\nabla\,\mu(\nabla u^h)^T\,da &= \sum_{i\in\mathcal{I}_0}\frac{1}{2A_{i+1/2}}\vec{n}_{i+1/2}\int_{T_{i+1/2}}\mu(\nabla u^h)^T da \\ &= \sum_{i\in\mathcal{I}_0}\frac{1}{2A_{i+1/2}}\vec{n}_{i+1/2}(\nabla u^h)^T_{T_{i+1/2}}\int_{T_{i+1/2}}\mu da \\ &= \sum_{i\in\mathcal{I}_0}\frac{\overline{\mu}_{i+1/2}}{2}\vec{n}_{i+1/2}(\nabla u^h)^T_{T_{i+1/2}}\end{aligned} \qquad (A.5)$$

where $\overline{\mu}_{i+1/2}$ is the integral average of μ in $T_{i+1/2}$. Inserting the triangle gradient formula we obtain a discretized formula for the Galerkin integral.

$$\begin{aligned}&\int_{\Omega_0}\phi^h\nabla\,\mu(\nabla u^h)^T\,da = \\ &-\frac{1}{4}\sum_{i\in\mathcal{I}_0}\frac{\overline{\mu}_{i+1/2}}{A_{i+1/2}}\vec{n}_{i+1/2}(u_0^h\vec{n}_{i+1/2}^T + u_i^h\vec{n}_{i+1}^T - u_{i+1}^h\vec{n}_i^T)\end{aligned} \qquad (A.6)$$

Regrouping of terms and removal of a constant solution yields the following simplified form

$$\begin{aligned}\int_{\Omega_0}\phi^h\nabla\,\mu(\nabla u^h)^T\,da &= \int_{\Omega_0}\nabla\mu(\nabla(u^h - u_0^h))^T\,da \\ &= \sum_{i\in\mathcal{I}_0} M_i(u_i^h - u_0^h)\end{aligned} \qquad (A.7)$$

with

$$M_i = -\frac{1}{4}\left[\frac{\overline{\mu}_{i+1/2}}{A_{i+1/2}}\vec{n}_{i+1/2}(\vec{n}_{i+1})^T - \frac{\overline{\mu}_{i-1/2}}{A_{i-1/2}}\vec{n}_{i-1/2}(\vec{n}_{i-1})^T\right] \qquad (A.8)$$

Even though this formula is very simple, it is not compatible with the edge data structure mentioned in section 2. Using some simple identities we will now rewrite the weight formula in a form which is compatible with the edge data structure.

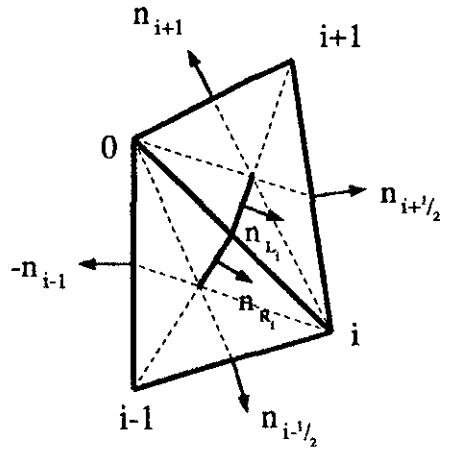

Figure A.2. Local geometry configuration.

Referring to figure A.2, we have the following vector identities:

$$\vec{n}_{i-1/2} = 3\vec{n}_{R_i} - \frac{1}{2}\vec{n}_i, \quad \vec{n}_{i-1} = 3\vec{n}_{R_i} + \frac{1}{2}\vec{n}_i \quad (A.9)$$

and similarly

$$\vec{n}_{i+1/2} = 3\vec{n}_{L_i} + \frac{1}{2}\vec{n}_i, \quad \vec{n}_{i+1} = -3\vec{n}_{L_i} + \frac{1}{2}\vec{n}_i \quad (A.10)$$

It is useful to decompose the tensor product terms into symmetric and skew-symmetric parts, for example:

$$-\vec{n}_{i-1/2}(\vec{n}_{i-1})^T = \underbrace{(\frac{1}{4}\vec{n}_i\vec{n}_i^T - 9\vec{n}_R\vec{n}_R^T)}_{Symmetric} + \underbrace{\frac{3}{2}(\vec{n}_i\vec{n}_R^T - \vec{n}_R\vec{n}_i^T)}_{Skew-symmetric}$$

and

$$\vec{n}_{i+1/2}(\vec{n}_{i+1})^T = \underbrace{(\frac{1}{4}\vec{n}_i\vec{n}_i^T - 9\vec{n}_L\vec{n}_L^T)}_{Symmetric} + \underbrace{\frac{3}{2}(\vec{n}_i\vec{n}_L^T - \vec{n}_L\vec{n}_i^T)}_{Skew-symmetric}$$

Upon dividing by the area terms, some simple algebra reveals that

$$-\frac{\vec{n}_{i-1/2}(\vec{n}_{i-1})^T}{A_{i-1/2}} = \frac{(\frac{1}{4}\vec{n}_i\vec{n}_i^T - 9\vec{n}_{R_i}\vec{n}_{R_i}^T)}{A_{i-1/2}} + \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad (A.11)$$

similarly

$$\frac{\vec{n}_{i+1/2}(\vec{n}_{i+1})^T}{A_{i+1/2}} = \frac{(\frac{1}{4}\vec{n}_i\vec{n}_i^T - 9\vec{n}_{L_i}\vec{n}_{L_i}^T)}{A_{i+1/2}} + \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad (A.12)$$

So in summary

$$\begin{aligned} M_i &= -\frac{1}{4}\left[\frac{\overline{\mu}_{i+1/2}}{A_{i+1/2}}\vec{n}_{i+1/2}(\vec{n}_{i+1})^T - \frac{\overline{\mu}_{i-1/2}}{A_{i-1/2}}\vec{n}_{i-1/2}(\vec{n}_{i-1})^T\right] \\ &= -\frac{1}{4}\left[\overline{\mu}_{i+1/2}\frac{\frac{1}{4}\vec{n}_i\vec{n}_i^T - 9\vec{n}_{L_i}\vec{n}_{L_i}^T}{A_{i+1/2}}\right. \\ &\qquad + \overline{\mu}_{i-1/2}\frac{\frac{1}{4}\vec{n}_i\vec{n}_i^T - 9\vec{n}_{R_i}\vec{n}_{R_i}^T}{A_{i-1/2}} \\ &\qquad \left. + (\overline{\mu}_{i+1/2} - \overline{\mu}_{i-1/2})\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}\right] \end{aligned} \quad (A.13)$$

The second form is compatible with an edge data structure where edge vertices and adjacent cell centroids are known.

2-D Discretization of $\nabla \cdot \mu \nabla u$

$$\begin{aligned} \int_{\Omega_0} \phi^h \nabla \cdot \mu \nabla u^h \, da &= \text{Trace} \int_{\Omega_0} \phi^h \nabla \mu (\nabla u^h)^T \, da \\ &= \sum_{i \in \mathcal{I}_0} W_i (u_i^h - u_0^h) \end{aligned} \quad (A.14)$$

$$W_i = -\frac{1}{4}\left[\overline{\mu}_{i+1/2}\frac{\vec{n}_{i+1/2}\cdot\vec{n}_{i+1}}{A_{i+1/2}} + \overline{\mu}_{i-1/2}\frac{\vec{n}_{i-1/2}\cdot\vec{n}_{i-1}}{A_{i-1/2}}\right] \quad (A.15)$$

The area of a triangle can be expressed in terms of the magnitude of the cross product of the scaled edge normals

$$A_{i+1/2} = \frac{1}{2}|\vec{n}_{i+1/2} \times \vec{n}_{i+1}|$$

$$A_{i-1/2} = \frac{1}{2}|\vec{n}_{i-1/2} \times \vec{n}_{i-1}|$$

$$W_i = -\frac{1}{2}\left[\overline{\mu}_{i+1/2}\frac{(\vec{n}_{i+1/2}\cdot\vec{n}_{i+1})}{|\vec{n}_{i+1/2}\times\vec{n}_{i+1}|} - \overline{\mu}_{i-1/2}\frac{(\vec{n}_{i-1/2}\cdot\vec{n}_{i-1})}{|\vec{n}_{i-1/2}\times\vec{n}_{i-1}|}\right] \quad (A.16)$$

Finally we can express the dot and cross products in terms of the local angles as sketched in figure A.3.

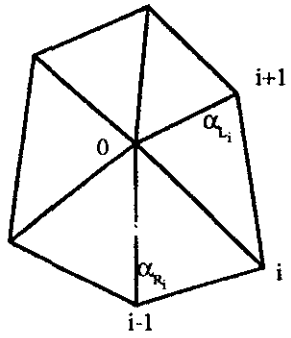

Figure A.3. Local angles for triangles sharing edge $e(v_0, v_i)$.

$$\frac{\vec{n}_{i+1/2}\cdot\vec{n}_{i+1}}{|\vec{n}_{i+1/2}\times\vec{n}_{i+1}|} = -\frac{\cos(\alpha_{L_i})}{\sin(\alpha_{L_i})} = -\text{cotan}(\alpha_{L_i}) \quad (A.17)$$

and

$$-\frac{\vec{n}_{i-1/2}\cdot\vec{n}_{i-1}}{|\vec{n}_{i-1/2}\times\vec{n}_{i-1}|} = -\frac{\cos(\alpha_{R_i})}{\sin(\alpha_{R_i})} = -\text{cotan}(\alpha_{R_i}) \quad (A.18)$$

Inserting these formulas yields a particularly simple form of the weight factors W_i:

$$W_i = \frac{1}{2}\left[\overline{\mu}_{i+1/2}\text{cotan}(\alpha_{L_i}) + \overline{\mu}_{i-1/2}\text{cotan}(\alpha_{R_i})\right] \quad (A.19)$$

Equation (A.19) is particularly useful in theoretical studies.

3-D Hessian and Laplacian Formulas

As in the 2-D case we begin with the Galerkin integral equation for the Hessian-like matrix of derivatives.

$$\int_\Gamma \phi^h \nabla \mu (\nabla u^h)^T \, dv = - \int_\Gamma \mu (\nabla \phi^h) \, (\nabla u^h)^T \, dv$$

In this formula V_Γ is the volume formed by the union of all tetrahedra that share vertex v_0. Following a procedure identical to the 2-D case, we can derive the analogous 3-D edge formula for the matrix of second derivatives

$$\int_{V_{\Gamma_0}} \phi^h \nabla \mu (\nabla u^h)^T \, dv = \sum_{i \in \mathcal{I}_0} M_i (u_i - u_0) \qquad (A.20)$$

where

$$M_i = -\frac{1}{9} \sum_{k=1}^{d(v_0,v_i)} \frac{\overline{\mu}_{k+1/2}}{V_{k+1/2}} \vec{\mathbf{S}}_{k+1/2} (\vec{\mathbf{S}}'_{k+1/2})^T \qquad (A.21)$$

$\mathcal{I}_0$ is the set of indices of all adjacent neighbors of v_0 connected by incident edges, k a local cyclic index describing the associated vertices which form a polygon of degree $d(v_0, v_i)$ surrounding the edge $e(v_0, v_i)$. The subscript $k + 1/2$ indicates quantities associated with the tetrahedron with vertices v_0, v_i, v_k and v_{k+1} as shown in figure A.4.

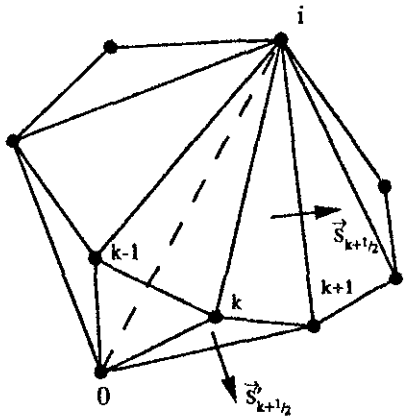

Figure A.4. Set of tetrahedra sharing edge $e(v_0, v_i)$ with local cyclic index k.

3-D Discretization of $\nabla \cdot \mu \nabla u$

$$\begin{aligned} \int_{\Gamma_0} \phi^h \nabla \cdot \mu \nabla u^h \, d\, v &= \text{Trace} \int_{\Gamma_0} \phi^h \nabla \mu (\nabla u^h)^T \, dv \\ &= \sum_{i \in \mathcal{I}_0} W_i (u_i - u_0) \end{aligned} \qquad (A.22)$$

where

$$W_i = -\frac{1}{9} \sum_{k=1}^{d(v_0,v_i)} \frac{\overline{\mu}_{k+1/2}}{V_{k+1/2}} \vec{\mathbf{S}}_{k+1/2} \cdot \vec{\mathbf{S}}'_{k+1/2} \qquad (A.23)$$

It can be shown that the volume of a tetrahedron is given by

$$V_{k+1/2} = \frac{2}{3} \frac{|\vec{\mathbf{S}}_{k+1/2} \times \vec{\mathbf{S}}'_{k+1/2}|}{|\Delta \mathbf{R}_{k+1/2}|} \qquad (A.24)$$

where $|\Delta \mathbf{R}_{k+1/2}|$ is the length of the edge shared by the faces associated with $\vec{\mathbf{S}}_{k+1/2}$ and $\vec{\mathbf{S}}'_{k+1/2}$.

$$W_i = -\frac{1}{6} \sum_{k=1}^{d(v_0,v_i)} \overline{\mu}_{k+1/2} |\Delta R_{k+1/2}| \frac{\vec{\mathbf{S}}_{k+1/2} \cdot \vec{\mathbf{S}}'_{k+1/2}}{|\vec{\mathbf{S}}_{k+1/2} \times \vec{\mathbf{S}}'_{k+1/2}|} \qquad (A.25)$$

Finally we can rewrite the dot and cross product in terms of the cotangent of the face angle.

$$\frac{\vec{\mathbf{S}}_{k+1/2} \cdot \vec{\mathbf{S}}'_{k+1/2}}{|\vec{\mathbf{S}}_{k+1/2} \times \vec{\mathbf{S}}'_{k+1/2}|} = -\frac{\cos(\alpha_{k+1/2})}{\sin(\alpha_{k+1/2})} = -\text{cotan}(\alpha_{k+1/2})$$

As in the 2-D case, the weights W_i now have a particularly simple form:

$$W_i = \frac{1}{6} \sum_{k=1}^{d(v_0,v_i)} \overline{\mu}_{k+1/2} |\Delta R_{k+1/2}| \text{cotan}(\alpha_{k+1/2}) \qquad (A.26)$$

§ Appendix B: Summary of a One-Equation Turbulence Transport Model

This appendix gives a complete summary of the one-equation model as it appeared in Baldwin and Barth [28]. The model consists of a single advection-diffusion-source term equation for the field variable $\nu \tilde{R}_T$ which is directly proportional to the eddy viscosity except very near a solid wall.

$$\begin{aligned} \frac{D(\nu \tilde{R}_T)}{Dt} =& (c_{\epsilon_2} f_2 - c_{\epsilon_1}) \sqrt{\nu \tilde{R}_T P} + (\nu + \frac{\nu_t}{\sigma_R}) \nabla^2 (\nu \tilde{R}_T) \\ & - \frac{1}{\sigma_\epsilon} (\nabla \nu_t) \cdot \nabla (\nu \tilde{R}_T) \end{aligned} \qquad (B.1)$$

In equation (B.1), the following functions are required:

$$
\begin{aligned}
\frac{1}{\sigma_\epsilon} &= (c_{\epsilon_2} - c_{\epsilon_1})\sqrt{c_\mu}/\kappa^2 \\
\sigma_R &= \sigma_\epsilon \\
\nu_t &= c_\mu(\nu\tilde{R}_T)D_1 D_2 \\
\mu_t &= \rho\nu_t \\
D_1 &= 1 - \exp(-y^+/A^+) \\
D_2 &= 1 - \exp(-y^+/A_2^+) \\
P &= \nu_t\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\frac{\partial U_i}{\partial x_j} - \frac{2}{3}\nu_t\left(\frac{\partial U_k}{\partial x_k}\right)^2 \\
f_2(y^+) &= \frac{c_{\epsilon_1}}{c_{\epsilon_2}} + (1 - \frac{c_{\epsilon_1}}{c_{\epsilon_2}})(\frac{1}{\kappa y^+} + D_1 D_2)\Bigg(\sqrt{D_1 D_2} \\
&\quad + \frac{y^+}{\sqrt{D_1 D_2}}\Big(\frac{1}{A^+}\exp(-y^+/A^+)\ D_2 \\
&\quad + \frac{1}{A_2^+}\exp(-y^+/A_2^+)\ D_1\Big)\Bigg)
\end{aligned}
$$

The following constants have been recommended in [28]:

$$
\begin{aligned}
\kappa &= 0.41, \quad c_{\epsilon_1} = 1.2, \quad c_{\epsilon_2} = 2.0 \\
c_\mu &= 0.09, \quad A^+ = 26, \quad A_2^+ = 10
\end{aligned}
$$

We also recommend the following boundary conditions for (B.1):

1. **Solid Walls**: Specify $\tilde{R}_T = 0$.

2. **Inflow** ($\mathbf{V}\cdot\mathbf{n} < 0$): Specify $\tilde{R}_T = (\tilde{R}_T)_\infty < 1$.

3. **Outflow** ($\mathbf{V}\cdot\mathbf{n} > 0$): Extrapolate $\tilde{R}_T$ from interior values.

§ References

1. Barth, T. J., and Jespersen, D. C., "The Design and Application of Upwind Schemes on Unstructured Meshes", AIAA-89-0366, Jan. 9-12, 1989.
2. Barth, T. J., and Frederickson, P. O., "Higher Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction", AIAA-90-0013, Jan. 8-11, 1990.
3. Slack, D. C., Walters, R. W., and Löhner, R., "An Interactive Adaptive Remeshing Algorithm for the Two-Dimensional Euler Equations", AIAA-90-0331, Jan. 8-11, 1990.
4. Whitaker, D. L., Slack, D. C., and Walters, R. W., "Solution Algorithms for the Two-Dimensional Euler Equations on Unstructured Meshes",AIAA-90-0697.
5. Desideri, J. A., and Dervieux, A., "Compressible Flow Solvers Using Unstructured Grids", VKI Lecture Series 1988-05, March 7-11, 1988, pp. 1—115.
6. Jameson, A. and Mavriplis, D., "Finite Volume Solution of the Two-Dimensional Euler Equations on a Regular Triangular Mesh", AIAA paper 85-0435, January 1985.
7. Mavriplis, D., "Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes", ICASE Report No. 87-53.
8. Mavriplis, D. and Jameson, A., "Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes', AIAA paper 87-0353, January 1987.
9. Morgan K., Peraire J.,, "Finite Elements for Compressible Flows", VKI Lecture Notes, 1988.
10. Lohner, R., Morgan K., and Peraire J., "Finite Elements for Compressible Flow", Numerical Methods for Fluid Dynamics (K.W. Morton and M.J. Baines eds.) Oxford University Press, 1986.
11. Shakib, F., "Finite Element Analysis of the Compressible Euler and Navier-Stokes Equations", PhD Thesis, Department of Mechanical Engineering, Stanford University, 1988.
12. Ciarlet, P.G., Raviart, P.-A., "Maximum Principle and Uniform Convergence for the Finite Element Method", Comp. Meth. in Appl. Mech. and Eng., Vol. 2., 1973, pp. 17-31.
13. Hammond, S., and Barth, T.J., "An Efficient Massively Parallel Euler Solver for Unstructured Grids", AIAA paper 91-0441, Reno, 1991.
14. Chrobak, M., and Eppstein, D., "Planar Orientations with Low Out-Degree and Compaction of Adjacency Matrices", Theoretical Computer Science, to appear, 1990.
15. Roe, P.L.,"Error Estimates for Cell-Vertex Solutions of the Compressible Euler Equations", ICASE report 87-6, 1987.
16. Lawson, C. L., "Properties of n-dimensional Triangulations" CAGD, Vol. 3, April 1986, pp. 231–246.
17. Mavriplis, D., "Adaptive Mesh Generation for Viscous Flows Using Delaunay Triangulation," ICASE Report No. 88-47,1988.
18. Babuška, I., and Aziz, A. K., "On the Angle Condition in the Finite Element Method", SIAM J. Numer. Anal., Vol. 13, No. 2, 1976.
19. Lawson, C. L., "Software for C^1 Surface Interpolation", Mathematical Software III, (Ed., John R. Rice), Academic Press, New York, 1977.
20. Nira, D., Levin, D., Rippa, S.,"Data Dependent Triangulations for Piecewise Linear Interpolation", Manuscript, School of Mathematical Sciences, Tel-Aviv University, 1989.
21. Barth, T.J., "On Unstructured Grids and Solvers," Computational Fluid Dynamics, Lecture Series 1990-03, Published by the Von Karman Instit., Belgium, March, 1990.
22. Lawson, C. L., Private communication.
23. Roe, P.L.,"Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes", J. Comput. Phys., Vol 43, 1981.
24. Venkatakrishnan, V., and Barth, T.J., "Application of Direct Solvers to Unstructured Meshes for the Euler and Navier-Stokes Equations Using Upwind Schemes," AIAA Paper 89-0364, 1989.
25. Barth, T.J.,"Analysis of Implicit Local Linearization Techniques for Upwind and TVD Schemes," AIAA Paper 87-0595, 1987.
26. B. van Leer, C.H. Tai, and Powell, K.G., "Design of Optimal-smoothing Multi-stage Schemes for the Euler Equations", AIAA 9th Computational Fluid Dynamics Conference, 1989.
27. Rostand, P., "Algebraic Turbulence Models for the Computation of Two-dimensional High Speed Flows Using Unstructured Grids", ICASE Report 88-63, 1988.
28. Baldwin, B.S., and Barth, T.J.,"A One-Equation Turbulence Transport Model for High Reynolds Number Wall-Bounded Flows," NASA TM-102847, August 1990.
29. Cook, P.H., McDonald M.A., Firmin, M.C.P., "AEROFOIL RAE 2822 Pressure Distributions, and Boundary Layer and Wake Measurements," AGARD Advisory Report No. 139, 1979.
30. Adair, D., and Horne, W.C., "Characteristics of Merging Shear Layers and Turbulent Wakes of a Multi-element Airfoil," NASA TM 100053, Feb. 1988.