Unstructured-grid large-eddy simulation of flow over an airfoil

By Kenneth Jansen

1. Motivation and objectives

Historically, large-eddy simulations (LES) have been restricted to simple geometries where spectral or finite difference methods have dominated due to their efficient use of structured grids. Structured grids, however, not only have difficulty representing complex domains and adapting to complicated flow features, but also are rather inefficient for simulating flows at high Reynolds numbers. The lack of efficiency stems from the need to resolve the viscous sublayer, which requires very fine resolution in all three directions near the wall. Structured grids make use of a stretching to reduce the normal grid spacing but must carry the fine resolution in the streamwise and spanwise directions throughout the domain. The unnecessarily fine grid for much of the domain leads to disturbingly high grid estimates. Chapman (1979), and later Moin & Jimenéz (1993), pointed out that, in order to advance the technology to airfoils at flight Reynolds numbers, structured grids must be abandoned in lieu of what are known as nested or unstructured grids. Fig. 1 illustrates the ability of an unstructured mesh to refine only the near-wall region. Note the large number of points near the wall (where the fine vortical features need better resolution) and the coarseness in all directions away from the wall (where the scales are much larger). The important difference between this approach and the usual structured grid stretching is that the number of elements used to discretize the spanwise and streamwise features of the flow is reduced in each successive layer coming off the wall. This is due to the fact that the elements not only grow in the normal direction but in the other directions as well. This greatly reduces the total number of points or elements required for a given Reynolds number flow.

The finite element method can efficiently solve the Navier-Stokes equations on unstructured grids. Although the CPU cost per time step per element is somewhat higher than structured grid methods, this effect is more than offset by the reduction in the number of elements. The use of unstructured grids, coupled with the advances in dynamic subgrid-scale modeling such as those made by Germano *et al.* (1991) and Ghosal *et al.* (1994), make LES of an airfoil tractable. We have chosen the NACA 4412 airfoil at maximum lift as the first simulation since this flow has not been successfully simulated with the Reynolds-averaged Navier-Stokes equations. Coles and Wadcock (1979) performed a detailed experimental study of this flow. Subsequently, Hasting and Williams (1987) also performed an experimental study. Finally, Wadcock (1987) re-examined the Coles and Wadcock data and the Hastings and Williams data. He synthesized the existing data with some recent measurements and concluded that the maximum lift configuration for the NACA 4412 airfoil at Reynolds number based on chord $Re_c = u_\infty c/\nu = 1.64 \times 10^6$ is 12^o angle of attack.

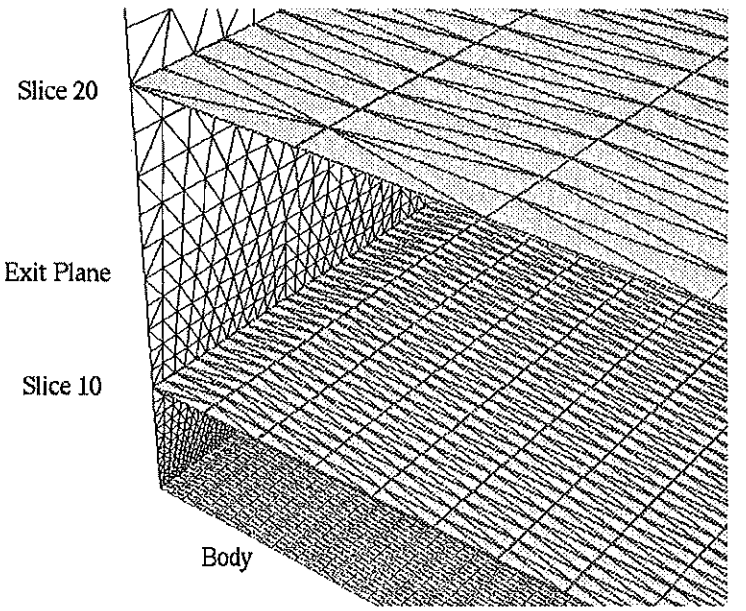

FIGURE 1. A portion of an unstructured grid that illustrates the varying resolution in the streamwise and spanwise directions in layers coming off the wall. Note that the y direction has been scaled to allow visualization of each layer.

2. Accomplishments

2.1 Mesh generation

A mesh generator has been developed which achieves special requirements of large-eddy simulation. These requirements arise from the need to resolve the near-wall structures. In this region the elements should have a streamwise spacing of 200 wall units ($\Delta_x^+ = 200$), a spanwise spacing of 50 wall units ($\Delta_z^+ = 50$), and a normal direction spacing of 1 wall unit ($\Delta_y^+ = 1$). A wall unit is a function of the friction velocity (u_τ) and, therefore, is also a function of position on the airfoil. The experimental friction velocity was used to determine the appropriate spacing in each direction at each point on the airfoil. This fine near-wall spacing is continued in the normal direction for approximately 30 wall units. Only the normal direction spacing is allowed to grow in this interval. Once outside of the near-wall region ($y^+ = 30$), the turbulent scales that need to be resolved become larger and the grid is smoothly coarsened in all directions. Great care is taken to ensure a smooth transition as preliminary studies have shown that non-smooth coarsening in the presence of gradients can greatly reduce accuracy. The domain can be made reasonably short in the spanwise direction by employing periodicity. Moin & Jimenéz predicted unstructured or zonal grids of the type described above would lead to meshes with 1.2×10^6 points for airfoils with a chord Reynolds number of $Re_c = 1.0 \times 10^6$, assuming a span of one-fifth of the chord. The mesh generated

for this span contains 1.0×10^6 points, which is 20 percent less than the prediction even though the Reynolds number is 64 percent higher. The additional savings are the result of the use of the local wall units to determine spacing rather than a global wall unit assumption as was done in the past (Chapman, Moin & Jimenéz, and Jansen (1993a)). A structured grid mesh with the same near-wall resolution would require over 2.6×10^7 points. The difference becomes even more dramatic at flight Reynolds numbers.

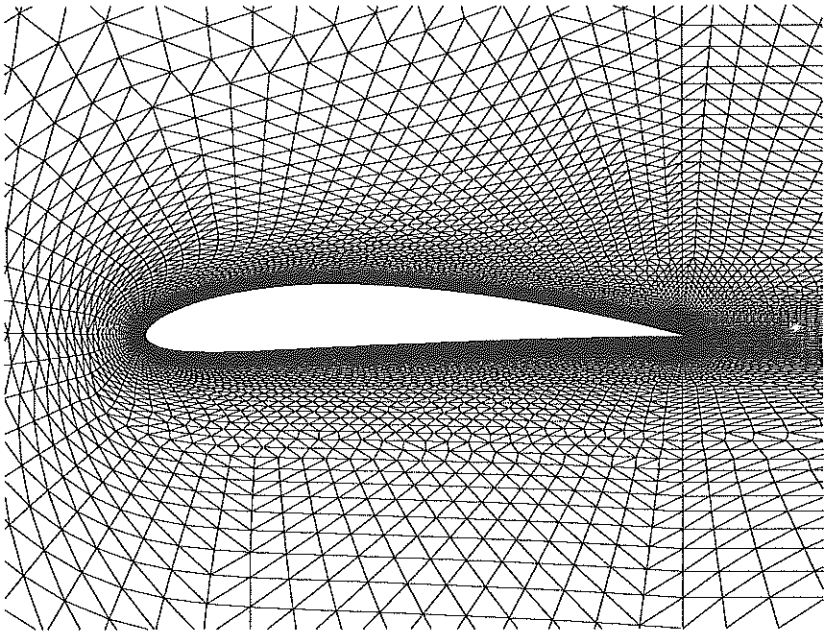

FIGURE 2. The periodic plane of the three-dimensional airfoil mesh (the mesh has been magnified to show the resolution near the airfoil).

The periodic plane of the three-dimensional airfoil mesh generated for this flow can be seen in Fig. 2. Note the smooth variation in element size. Note also the rapid growth in Δ_y in the wake. This not only reduces the number of points required, but also reduces stiffness associated with fine spacing in a region of fairly large vertical flow (large vortical motions shedding off of the tail). Fig. 3 is a plan view of the grid at approximately 30 wall units off of the upper surface. Note the variation in spanwise and streamwise spacing as a function of chord position. The figure has been broken in three pieces to afford a closer look at the very narrow domain. The spanwise domain has been reduced by a factor of four to allow more rapid computation of preliminary results presented in section 2.4. This grid contains only 0.25×10^6 points.

2.2 Computer code

The finite element formulation being used in this work is based upon the work

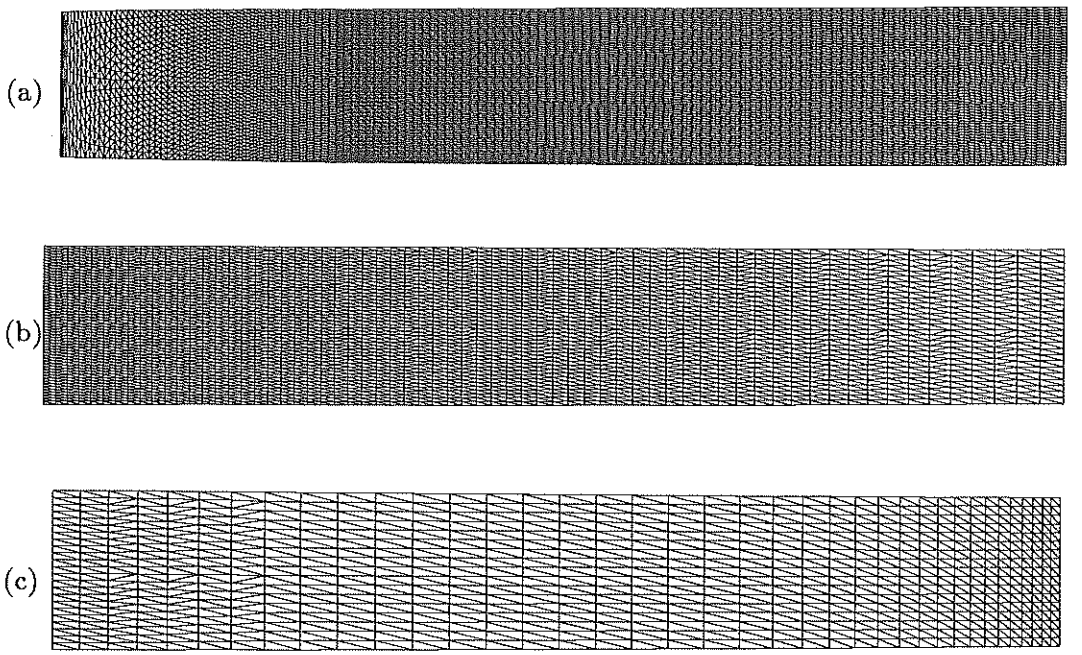

FIGURE 3. Plan view of a slice through a three-dimensional airfoil grid at $y^+ = 30$ from the upper surface. (a) displays the first third of the chord ($0.0 < x/c < 0.33$), (b) the second third ($0.33 < x/c < 0.67$) and (c) the final third ($0.67 < x/c < 1.0$). Note that the large variation in spanwise and streamwise spacing match the resolution requirements of the flow locally.

of Jansen *et al.* (1993b) and Johan *et al.* (1993). The code was extended to time-accurate calculations in the past year and was validated by solving the problem of vortex shedding behind a cylinder. Implicit time integration is required due to the very high acoustic CFL numbers encountered in flows of this type. Different integration schemes were studied and the trapezoidal rule was found to be the most efficient for external flow problems. It should be noted that this time-integration scheme was observed to be a poor choice for internal flows such as channel flows due to undamped acoustic waves in a bounded domain. To perform this type of simulation properly would require development of a new time integration scheme that would damp temporally unresolved accoustic waves. The computational domain of the airfoil has open boundaries far from the airfoil surface, and no such difficulties arise in this case.

The code has proven to be very efficient on parallel architectures such as the CM5. For large problems, such as the one we consider here, very high flop rates can be achieved (25 MFLOPS per processor). The CM5 has also been a far more available resource than the Cray C90 in the past year.

2.3 Filtering operators for the finite element method

The dynamic model requires a "test filtering" operation defined as

$$\widehat{f}(\mathbf{x}) = \int f(\mathbf{x}')G(\mathbf{x},\mathbf{x}')d\mathbf{x}'$$

as part of the procedure to determine the model coefficient (Germano *et al.* and Ghosal *et al.*). If G is a "top-hat" filter and the above equation is integrated with Simpson's rule, the filtering operation leads to the following formula for each internal node A ($A = 1, \cdots, n_p$) in the mesh with n_p such internal nodes

$$\widehat{f}^A = \frac{1}{6}(4f^A + f_E^A + f_W^A)$$

where the subscripts E and W denote the point due East and West for node A. In multi-dimensions the process is simply repeated in each direction. On an unstructured grid there often is not a point due East or West. Furthermore, the location of the points that might approximate the due East or West neighbor is not simply determined through a recursive formula as is the case in a structured grid. One could pre-process these approximate neighbors and store a pointer list for each point in the grid. This would require additional memory which is unattractive. Also, for parallel machines this approach is very inefficient. The inefficiency stems from the fact that the nodal data is scattered among the processors and retrieval of that data requires substantial communication. The time required to perform these communication operations is often much larger than the time required to perform the actual calculations on parallel machines. For this reason it is attractive to explore element-based filtering procedures which minimize the amount of communication required. Four alternative filtering operators have been developed and are described next.

Method 1.) Start by obtaining the function at the element centroids, f^e, where the superscript e corresponds to the e^{th} element. Then define the following filter operation,

$$\widehat{f}^e = w_0 f^e + \sum_{i=1}^{n_f} w_i f_i^e$$

where f_i^e is the function value at the centroid of the element on the other side of the i^{th} face (there are n_f such faces for each element), and w_i are filter weights. For example, triangles have three faces, therefore, the filtering operation of a given function for a particular triangle involves the function value within the triangle and the function values within the three other triangles which surround it as illustrated in Fig. 4.

The problem with this method is that it requires an additional data structure to determine the elements which lie on the other side of each face of a given element. This data structure is not immediately available from existing finite element data structures. It could be pre-processed and stored, but this is unattractive. This

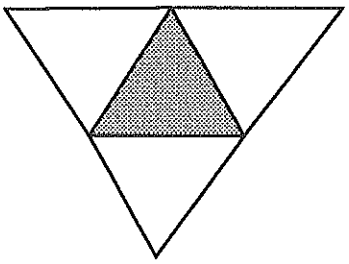

FIGURE 4. In method 1 the dynamic model filter for the function in the shaded triangle is determined through a weighted combination of the function value within the shaded triangle and the function values within the triangles which share a face with it (the 3 white triangles).

method may prove effective for finite volume schemes using an edge-based data structure.

Method 2.) An alternative approach is to approximate the the filtering operator by

$$\widehat{f}(\mathbf{x}) = f(\mathbf{x}) + \frac{1}{m}\tilde{\nabla}^2 f(\mathbf{x}) + O(\delta^4)$$

which extends the work of Carati (1994) to non-isotropic grids. Here $\tilde{\nabla}$ is a non-dimensional gradient (i.e. the gradient in a particular direction multiplied by the discretization width in that direction).

$$\tilde{\nabla} = \Delta_x \frac{\partial}{\partial x}\mathbf{i} + \Delta_y \frac{\partial}{\partial y}\mathbf{j} + \Delta_z \frac{\partial}{\partial z}\mathbf{k}$$

It is easily verified that this procedure gives the same result as a one-dimensional top-hat filter integrated with Simpson's rule ($m = 3$). In general $m = d + 2$ where d is the number of space dimensions.

This filter can be evaluated quite rapidly with the finite element method

$$\widehat{f}^A = f^A - \frac{1}{d+2}\{M^{BA}\}^{-1}\left(\int_\Omega N^B_{,i_n} f_{,i_n}\, d\Omega - \int_\Gamma N^B f_{,i_n} n_{i_n}\, d\Gamma\right)$$

where N^A is the basis function for node A (likewise B can be any node ($B = 1, \cdots, n_p$)), $\{M^{BA}\}$ is the finite element "mass matrix", Ω is the spatial domain, Γ is the boundary of the domain, and the subscript $,i_n$ denotes differentiation in the i^{th} direction multiplied by the length of the element in this direction. Note that we have included the boundary terms (there will be contributions when f is the strain-rate tensor due the non-zero strain-rates at the boundaries).

While this method requires more floating point operations than method 1, it requires no additional data structures (it uses existing finite element data structures) and very little communication. Algorithms of this type are already coded for the viscous terms; consequently these operations were easily parallelized.

The filtering operation defined above requires that all quantities that need to be filtered must be defined at the nodes. The most commonly used basis functions in finite element methods are C^0 continuous piece-wise polynomials. Function spaces of this type yield gradient quantities (such as the strain-rates) that are discontinuous at element boundaries. Before the filtering operator can be applied, it is necessary to project the strain-rates from the elements to the nodes. This is accomplished by a consistent finite element projection operator

$$S_{ij}^A = \{M^{BA}\}^{-1} \sum_{e=1}^{n_{el}} \int_{\Omega^e} N^B S_{ij}^e d\Omega_e$$

here S_{ij}^A is the strain-rate tensor at node A and S_{ij}^e is the strain-rate tensor as defined in element e. Note that the sum is over all the element domains Ω_e (there are n_{el} such domains).

With the strain-rate tensor globally projected to the nodes, we next interpolate $S_{ij}(\mathbf{x})$ with the basis functions,

$$S_{ij}(\mathbf{x}) = \sum_{A=1}^{n_p} N^A(\mathbf{x}) S_{ij}^A$$

The above procedure has been implemented in the parallel code. The tests thus far have used a "lumped mass" for $\{M^{BA}\}$, making inversion trivial. The cost of the dynamic model calculation of the eddy viscosity is less than one-fifth of a non-linear iteration. Therefore, even when only performing two non-linear iterations, the cost of the dynamic model is less than 10 percent of the total cost. This is as cheap or cheaper than many three-dimensional structured grid filtering operators. As mentioned before it also requires no additional memory.

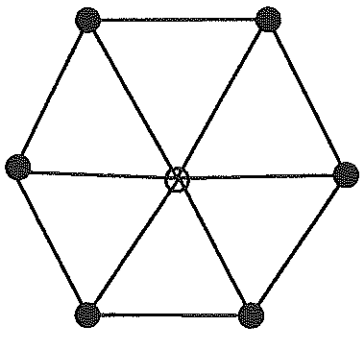

FIGURE 5. Generalized box filter for node A is defined as $G(\mathbf{x}^A) = 1$ for all the triangles surrounding node A.

Method 3.) The third method generalizes the notion of a top-hat or box filter to unstructured grids. Since the element domains do not necessarily form boxes,

it is more practical to define the filter function for node A as $G(\mathbf{x}^A) = 1$ for all the triangles which have node A as a vertex (see Fig. 5). In three dimensions this generalized box filter is an approximation to an ellipsoid (for isotropic grids it is approximately a sphere). This filter function is easily integrated against the function we desire to filter with the rectangle integration rule (equivalent to trapezoidal rule if the function is linear). This method has been implemented and is slightly cheaper than method 2 but may be less accurate since rectangle rule is less accurate than Simpson's rule. The new generalized box filter is not formed by structured grid lines, making Simpson's rule integration impossible. Studies are underway to quantify the differences between methods 2 and 3.

Method 4.) The fourth method is only appropriate when using a higher-order function space. Consider for example a one-dimensional quadratic element as shown in Fig. 6. It is possible to construct a filter from a combination of two interpolations. In our one-dimensional example the filtered value of a function f at the center of the element can be a weighted combination of the quadratic interpolation (which involves all three points) and linear interpolation of the endpoints viz.

$$\widehat{f} = \beta(f_{lin} + \alpha f_{quadr})$$

α and β can be determined to represent the filter of choice. For example $\alpha = \frac{1}{4}, \beta = \frac{2}{3}$ is equivalent to a top-hat filter integrated with Simpson's rule. Quadrilaterals (in two-dimensions) and hexahedra (in three-dimensions) pose no additional difficulty as they are constructed from tensor products of these one-dimensional functions. Triangles and tetrahedra are not quite so trivial but can none the less be constructed (see Fig. 7). The only difference here is that there is not a node at the center of the element.

Implicit in this method is the assumption that it is sufficient to calculate the dynamic model coefficient once in each element. This filtering method is only meaningful on the interior of the element since at the endpoints (or corners in multi-dimensions) the different functional representations yield the same value. At these positions no filtering would be accomplished. This is of little concern for tetrahedral meshes since in this case there are roughly the same number of quadratic tetrahedra as there are nodal points. Therefore, the number of points where the dynamic model coefficient is evaluated is roughly the same. What has changed is the point in space where the dynamic model coefficient is calculated. We have simply moved the position where we evaluate the dynamic model coefficient from the nodes to the element centroids. The reason for doing this is that no communication is required to determine the dynamic model coefficient at the centroid using this method. This approach promises to be far less costly than methods 2 and 3.

2.4 Preliminary simulations

To obtain a reasonable initial condition, the two-dimensional Reynolds-averaged Navier-Stokes equations were solved with a one-equation eddy-viscosity model. Three-dimensional turbulence fluctuations from Choi's (1994) structured grid simulation were then added to this two-dimensional Reynolds-averaged solution to obtain a reasonable three-dimensional starting field.

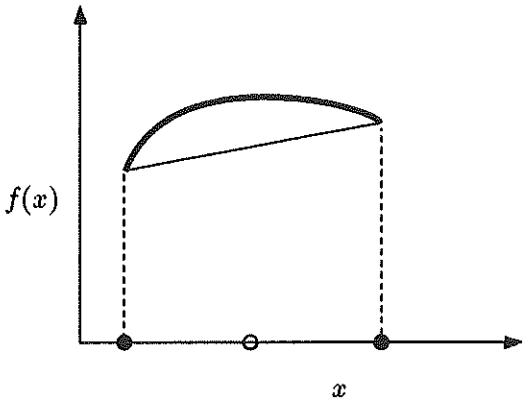

FIGURE 6. One-dimensional quadratic element is comprised of three nodes which quadratically interpolate (bold line) the nodal values of the function $f(x)$. By eliminating the center node (unfilled circle), a linear interpolant can be constructed (thin line) from the end nodes (filled circles).

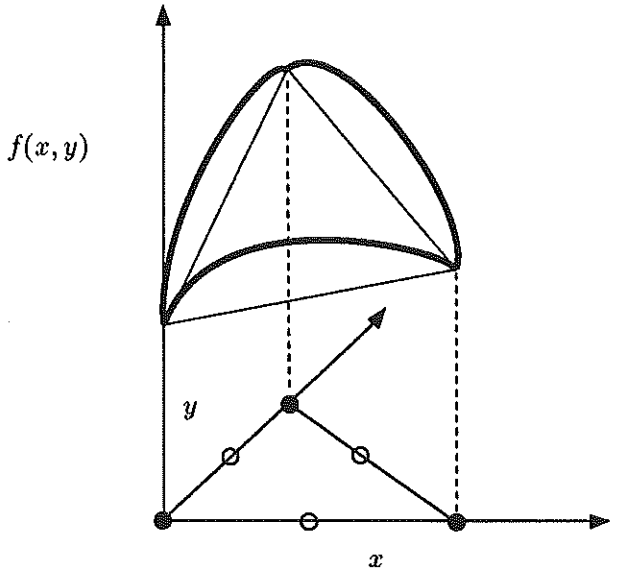

FIGURE 7. Quadratic triangle element is comprised of six nodes which quadratically interpolate the nodal values of the function $f(x,y)$. By eliminating the center node on each edge (unfilled circles), a linear interpolant can be constructed from the end nodes (filled circles).

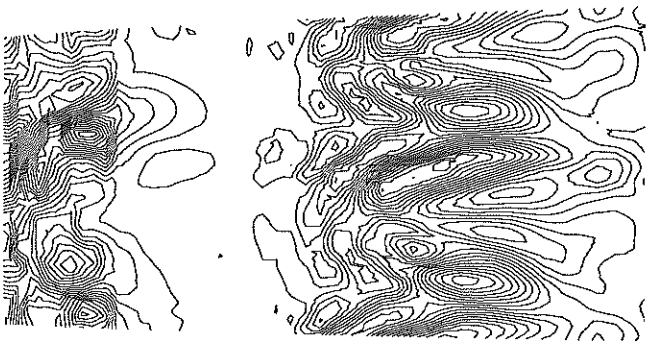

FIGURE 8. No sub-grid scale model simulation. Plan view spanwise velocity fluctuations at approximately 30 wall units away from the upper surface of the airfoil.

The first simulation was performed with no sub-grid scale model to establish a baseline case. The plan view of the spanwise velocity fluctuations at approximately 30 wall units away from the upper surface of the airfoil is shown in Fig. 8. Note the strong fluctuations beginning just after the nose, followed by a calm region and then continued fluctuations. The calm region is associated with a separation bubble which was not observed in the experiment. The calculation was discontinued after it became clear that the separation bubble was not a transient.

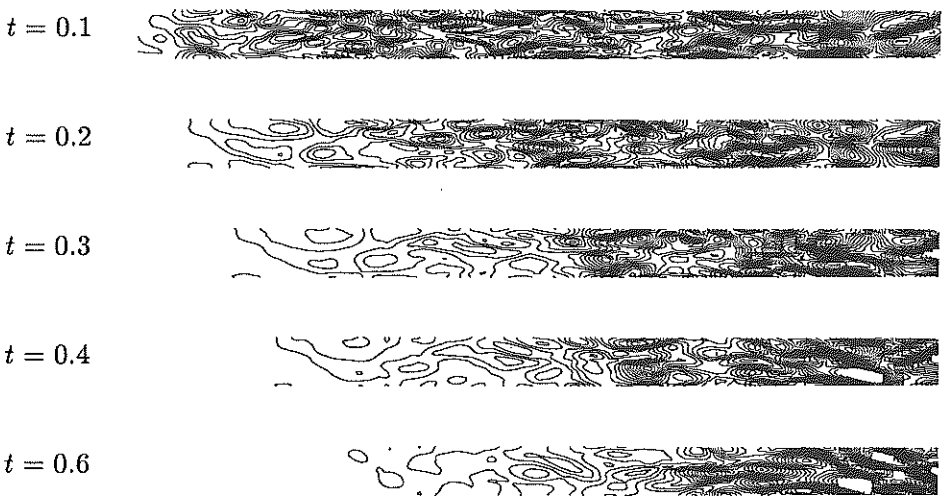

FIGURE 9. Constant-coefficient Smagorinsky model large-eddy simulation. Plan view spanwise velocity fluctuations at approximately 30 wall units away from the upper surface of the airfoil at various fractions of flow-over-chord times.

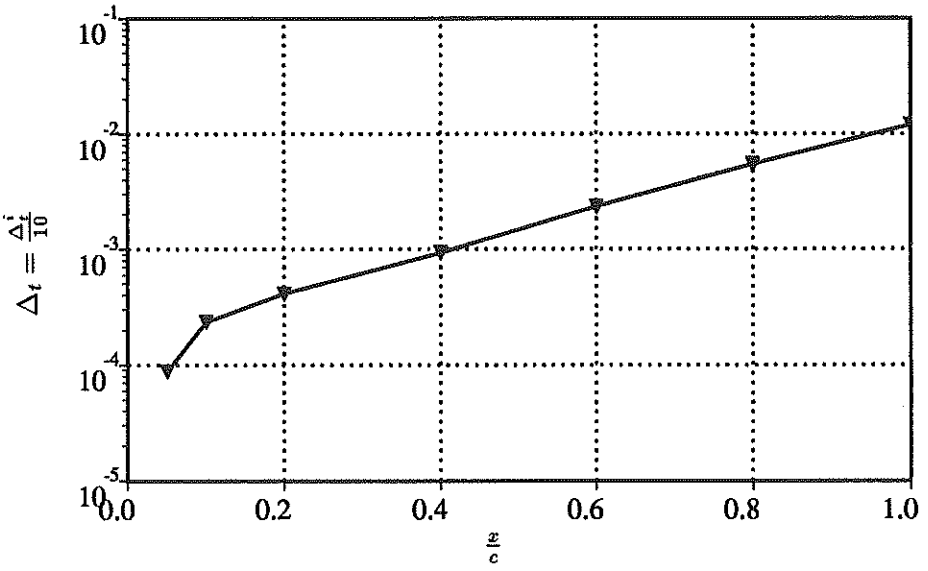

FIGURE 10. The time step which satisfies the one-tenth of an inertial time scale criterion as a function of position on the airfoil upper surface.

The second simulation performed utilized a constant-coefficient Smagorinsky model with wall damping. The additional eddy viscosity caused the flow to stay attached at the nose, but other difficulties were encountered. Fig. 9 shows a plan view of the fluctuating spanwise velocity field at approximately 30 wall units away from the upper surface. The full upper surface is shown at various fractions of flow-over-chord time. Note that the turbulence is convecting and interacting, but no new turbulence is being generated in the first two-thirds of the airfoil. The area of turbulent activity simply moves down the plate until it reaches the separation zone. Perhaps this should be the expected result as we have provided no disturbances to the boundary layer to sustain the turbulence. This simulation lacks the temporal and spatial resolution to undergo a natural transition. Strategies to force a cost-efficient transition will be investigated in future work. It is encouraging to note that the separated boundary layer maintains its turbulence as it should since it is an absolute instability (boundary layers are only convectively unstable).

A dynamic model simulation is under way. Preliminary results are quite promising. It seems that the laminar separation bubble at the leading edge that was observed in the coarse DNS is present at certain times in the dynamic model calculation. The highly transient separation seems to be causing enough of a disturbance to maintain turbulence over the entire airfoil upper surface. It is too early to tell if this disturbance is enough like the tripped boundary layer in the experiment to expect agreement further downstream. The calculation will be continued, and should it fail, more precise forcing strategies will be explored.

2.5 Time-step constraints

The current implementation of the code requires 10 seconds per non-linear iteration on the 128 node CM5. A real challenge in this flow is the very restrictive time step imposed by the flow in, and immediately following, transition. It is common when performing large-eddy simulations to choose the time step to be one tenth of an inertial time scale $\Delta t^i = \frac{\delta}{u_e}$. In this problem, the boundary layer thickness, δ, and the boundary layer edge velocity, u_e, vary with x/c. The effect of this variation on the desired time step is plotted in Fig. 10. The simulations described above were calculated at a time step of 2.0×10^{-4} non-dimensional time units (normalized by chord and freestream velocity). This time step is only time accurate beyond $\frac{x}{c} = 0.1$. At this time step one flow over the chord requires 5000 time steps. It seems clear that time-accurate resolution of the transition process would be very expensive (experimental trip was placed at $x/c = .02$).

3. Future plans

3.1 Dynamic model

The new filtering operators developed in section 2.3 should be validated on simple, well understood flows such as decaying isotropic turbulence. This flow is very sensitive to errors in the filter width ratio as the decay rate is very sensitive to this quantity. Certain simple triangulation patterns may lend themselves to a closed form analysis of the filter width, but mixed triangulations occur in practice. While studying this problem, we might also examine the influence of the least-squares stabilization operator and the cost effectiveness of higher order elements.

3.2 Transition

The preliminary simulations have demonstrated a need to aid transition of the flow as was done in the experiment. It would be more efficient to do these studies on a flat plate where the mesh could be kept smaller. The goal here is not to give a spatially accurate transition, but rather to create and sustain turbulence beyond a certain percent of chord. An accurate resolution of the transition process is a costly alternative that hopefully may be avoided.

3.3 Improve code performance

New results from Aliabadi & Tezduyar (1994) indicate that further simplification of the least-squares stabilization operator may be possible. These ideas offer the potential to reduce the number of floating point operations per time step by a factor of two. The effect of these simplifications on accuracy must be studied.

3.4 Airfoil simulation

The airfoil simulation will continue to be the main focus of the unstructured-grid large-eddy simulation program. The above topics are parallel projects that are designed to constantly improve the quality of the airfoil simulation.

REFERENCES

ALIABADI, S. K. & TEZDUYAR, T. E. 1994 Parallel fluid dynamics computations in aerospace applications. *University of Minnesota Supercomputer Institute Research Report UMSI 94/166.*

CARATI, D. 1994 Private communication.

CHAPMAN, D. R. 1979 Computational aerodynamics development and outlook. *AIAA J.* **17,** 1293.

CHOI, H. 1994 Private communication.

COLES, D., & WADCOCK, A. J. 1979 A flying-hot-wire study of two-dimensional mean flow past an NACA 4412 airfoil at maximum lift. *AIAA J.* **17,** 321.

GERMANO, M., PIOMELLI, U., MOIN, P. & CABOT, W. H. 1991 A dynamic subgrid-scale eddy viscosity model. *Phys Fluids A.* **3,** 1760.

GHOSAL, S., LUND, T. S., MOIN, P. & AKSELVOLL K. 1992 A dynamical localization model for large eddy simulation of turbulent flows. *J. Fluid Mech.* To appear.

HASTINGS, R. C.& WILLIAMS B. R. 1987 Studies of the flow field near a NACA 4412 aerofoil at nearly maximum lift. *Aero. J.* **91,** 29.

JANSEN, K. 1993 Unstructured grid large eddy simulation of wall bounded turbulent flows. *Annual Research Briefs 1993.* Center for Turbulence Research, NASA Ames/Stanford Univ.

JANSEN, K., JOHAN, Z., & HUGHES, T. J. R. 1993 Implementation of a one-equation turbulence model within a stabilized finite element formulation of a symmetric advective-diffusive system. *Comp Meth Appl Mech Eng.* **105,** 405.

JOHAN, Z., HUGHES, T. J. R., MATHUR, K. K., & JOHNSSON, S. L. 1992 A data parallel finite element method for computational fluid dynamics on the Connection Machine system. *Comp Meth Appl Mech Eng.* **99,** 113.

MOIN, P. & JIMENÉZ, J. 1993 Large-eddy simulation of complex turbulent flows. *AIAA-03-3099, AIAA 24th Fluid Dynamics Conference.*

WADCOCK, A. J. 1987 Investigation of low-speed turbulent separated flow around airfoils. *NACA CR 177450.*