is fairly good for both codes. Although the results shown here are worse than one would like, they are better than earlier results.[4]

The simulations employing the Sarkar compressibility correction were generally no better than those not. In the 40 K jet case the correction makes the PAB3D predictions significantly worse, and the ISAAC predictions are slightly high by about as much as they were low for the computations without the compressibility correction. In the 843 K jet case the compressibility correction significantly improves the PAB3D results but makes the ISAAC results dramatically worse.

Figure 1 shows a grouping of the results from each of the two codes; this grouping occurred in the other case as well. It suggests the differences in results can be caused more by differences in the codes than to differences in the models. To further investigate these differences, the k–ε model in PAB3D was implemented in ISAAC (except for the PAB3D form for χ_w, which involves derivatives not readily available in the ISAAC code), as was the PAB3D implementation of the Gatski–Speziale algebraic stress model (ASM). Comparisons between the ISAAC implementations of the models, the PAB3D implementations of the models, and the PAB3D models implemented in ISAAC are shown in Figs. 2 and 3 for the 40 K case. The results from ISAAC with its own models and with the PAB3D models were very similar for the k–ε model and the Gatski–Speziale algebraic-stress model, indicating that the differences between the ISAAC and PAB3D results are primarily caused by the codes themselves.

Summary

In summary, overall agreement of the computations with experiment is good. The two codes each gave fairly consistent results with the different turbulence models, and the differences between the codes seemed to be greater than the differences between the models. Additional evidence for this was given by the computations with ISAAC using the PAB3D versions of the models, which gave results much closer to the ISAAC results with its own models than to the PAB3D results. Possible reasons for these differences between the results of the two codes include different handling of viscous fluxes (thin shear layer in PAB3D vs full Navier Stokes in ISAAC), first-order advection in the PAB3D turbulence equations vs second-order advection in all equations in ISAAC, and other differences in the numerical algorithms employed in the two codes.

Finally, we note that the aerodynamic input is only part of the story, and the MGB noise prediction can emphasize or deemphasize different aspects of the input error.

Acknowledgments

This work was supported by NASA through the NASA Langley Research Center. The assistance of J. H. Morrison with ISAAC and P. Pao, J. Carlson, and K. S. Abdol-Hamid with PAB3D is gratefully acknowledged. P. Pao and J. Carlson supplied the grid. The ISAAC and PAB3D CFD codes were made available by NASA Langley Research Center.

References

[1]Seiner, J. M., "A New Rational Approach to Jet Noise Reduction," *Theoretical and Computational Fluid Dynamics*, Vol. 10, Nos. 1–4, 1998, pp. 373–383.

[2]Mani, R., Gliebe, P. R., and Balsa, J. F., "High-Velocity Jet Noise Source Location and Reduction," Task 2, Federal Aviation Administration Rept., FAA-RD-76-79-II, 1978.

[3]Lighthill, M. J., "On Sound Generated Aerodynamically. I. General Theory," *Proceedings of the Royal Society of London*, Vol. A211, 1952, pp. 564–587.

[4]Barber, T. J., Chiapetta, L. M., DeBonis, J. R., Georgiadis, N. J., and Yoder, D. A., "Assessment of Parameters Influencing the Prediction of Shear-Layer Mixing," *Journal of Propulsion and Power*, Vol. 15, No. 1, 1999, pp. 45–53.

[5]Seiner, J. M., Ponton, M. K., Jansen, B. J., and Lagen, N., "The Effects of Temperature on Supersonic Jet Noise Emission," AIAA Paper 92-02-046, June 1992.

[6]Morrison, J. H., "A Compressible Navier–Stokes Solver with Two-Equation and Reynolds-Stress Turbulence Closure Models," NASA CR 4440, May 1992.

[7]Abdol-Hamid, K. S., "Implementation of Algebraic-Stress Models in a General 3-D Navier–Stokes Method (PAB3D)," NASA CR 4702, Dec. 1995.

[8]Wilcox, D. C., *Turbulence Modeling for CFD*, DCW Industries, Inc., La Cañada, CA, 1993, Chap. 4.

[9]Gatski, T. B., and Speziale, C. G., "On Explicit Algebraic-Stress Models for Complex Turbulent Flows," *Journal of Fluid Mechanics*, Vol. 254, Sept. 1993, pp. 59–78.

[10]Girimaji, S. S., "Fully-Explicit and Self-Consistent Algebraic Reynolds-Stress Model," Inst. for Computer Applications in Science and Engineering, Rept. 95–82, Hampton, VA, Dec. 1995.

[11]Sarkar, S., Erlebacher, G., Hussaini, M. Y., and Kreiss, H. O., "The Analysis and Modelling of Dilatational Terms in Compressible Turbulence," *Journal of Fluid Mechanics*, Vol. 227, June 1991, pp. 473–493.

P. J. Morris
Associate Editor

Efficient Method for Calculating Wall Proximity

David A. Boger*
Pennsylvania State University,
University Park, Pennsylvania 16804

Introduction

THE problem of calculating the distance to the nearest surface from all of the points throughout a volume mesh is addressed by devising a finite search region and then conducting an efficient geometric search. The method is tested on 10 three-dimensional computational fluid dynamics (CFD) grids with up to 1,700,000 points in the volume and up to 130,000 points on the surface. The expense of the calculation is reduced by orders of magnitude compared to the simplest approach of checking the distance to every surface point. This method is especially useful when the distances need to be updated often, such as multiple-body or moving-appendage problems, and it can be easily retrofitted into existing codes.

Many physical models employed in modern CFD codes require wall proximity in their specification. The most common are low-Reynolds-number turbulence models, many of which rely on wall proximity in near-wall damping functions.[1–3] Researchers have observed that this reliance on explicit wall distance is ambiguous for all but the simplest topologies and that it is "not at all evident that wall distance bears a relationship to the structure of turbulence."[4] As a result, much current research in turbulence modeling is focused on achieving geometry independence.[5,6] The appropriateness of using models that contain wall distance explicitly is not considered here. It is simply observed that such models continue to enjoy widespread use.

For complex geometries the simplest approach, given a point within the volume, is to measure the distance to every point on every solid surface, keeping the minimum. Though simple, this requires $\mathcal{O}(N_v N_s)$ operations, where N_v and N_s are the number of volume and surface points, respectively. In many applications the calculation is performed once and saved, but when the distance function changes with time the efficiency of wall proximity computations can become important. The current note formulates a method for calculating wall proximity with a cost of $\mathcal{O}(N_v \log N_s)$, a gain of several orders of magnitude in the cases shown here.

At least three other methods for computing wall proximity appear in the literature. In the first an approximate distance to the wall was inferred from the solution of a Poisson equation for a length scale L, where $L = 0$ on solid boundaries.[7,8] Described as a "convenient

Received 19 May 2001; revision received 8 July 2001; accepted for publication 13 August 2001.

*Associate Research Engineer, Applied Research Laboratory.

alternative to crude search procedures," the method was designed to return smooth contours of the wall distance around sharp corners, which, for mixing-length turbulence models, helps to return smooth values of the turbulent viscosity and so was observed to help convergence. Despite using the solution to the Poisson equation, the turbulent viscosity needed "further smoothing," diminishing one of the main arguments for using the method.

"Fast-marching methods" were also applied to computing the distance to the nearest wall.[9] Here, the problem was recast as the solution for the "time of first arrival" for waves emanating at unit velocity from the solid surfaces or $|\nabla L| = 1$. Because the equation is not continuous in some regions (e.g., near corners), a diffusion term was added. The finite difference form of the resulting convection-diffusion equation was solved numerically.

The third method derived from an early model of the turbulent mixing length. Buleev postulated that the "characteristic linear scale L" used to determine the mean size of turbulent eddies should be "qualitatively connected with the reciprocal of the distance . . . to the channel wall" for flow in pipes of arbitrary cross section.[10] The resulting formula for L recommended by Buleev was

$$\pi L^{-1} = \int_D s^{-1}\,\mathrm{d}\phi$$

where s is the distance to the wall in the direction ϕ. For a point in space bounded by an infinite plane, L becomes equal to the actual distance from the point to the plane. Other researchers empirically modified Buleev's equation by multiplying constants and adding exponents to s and L (Refs. 11–13). But more recently the formula was used directly to approximate the wall proximity required in the Baldwin–Lomax model.[14]

Relative to these three methods, the algorithm proposed here might fall into the category of "crude search methods"; however, it is only approximate in that the surface has been discretized, is simple to implement, and is fast.

Determination of a Search Region

The brute force approach to calculating wall proximity is costly because it uses an infinite search region so that every surface point must be searched. Here, a finite search region is devised that encompasses only a small number of surface points so that it becomes worthwhile to conduct an advanced geometric search.

Define the surface S as a set of points $\boldsymbol{s}_j$ for $j = 1, N_s$. The distance between a point $\boldsymbol{x}$ and S can then be expressed as $\mathrm{d}(\boldsymbol{x}, S) = \min_j\{\mathrm{d}(\boldsymbol{x}, \boldsymbol{s}_j)\}$, where $\mathrm{d}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$ is simply the distance between two points. If $\mathrm{d}(\boldsymbol{x}_1, S)$ is known for some point $\boldsymbol{x}_1$, then by the triangle inequality, $\mathrm{d}(\boldsymbol{x}_2, S) \le \mathrm{d}(\boldsymbol{x}_2, \boldsymbol{x}_1) + \mathrm{d}(\boldsymbol{x}_1, S)$. The distance $r = \mathrm{d}(\boldsymbol{x}_2, \boldsymbol{x}_1) + \mathrm{d}(\boldsymbol{x}_1, S)$ thus defines a region about $\boldsymbol{x}_2$ in which a geometric search algorithm for S can be implemented.

Alternating Digital Tree

The alternating digital tree (ADT) is a type of binary tree, a data structure that can be used to access stored information quickly. The ADT has played an important role in unstructured grid generation by advancing front methods[15] (also Ives, D. C., private communication, Sept. 2000) and more recently in automated hole cutting for Chimera techniques.[16] The ADT is summarized briefly here; a more thorough discussion can be found in Ref. 15.

The ADT is constructed beginning with a node known as the root. This node can have up to two children, one to its left and one to its right. Each of these children may in turn have up to two children of its own, and so on. Any node can be considered the root of a subtree consisting of itself and all of its descendents. Access to any particular node is gained only by beginning at the root and visiting each of the ancestors of the node in turn.

Each node in the tree is associated with a rectangular prism $[\boldsymbol{c}, \boldsymbol{d})$. For the root $[\boldsymbol{c}, \boldsymbol{d})$ is the three-dimensional bounding box of the domain. The children of the root are then associated with subdivisions $[\boldsymbol{c}_L, \boldsymbol{d}_L)$ and $[\boldsymbol{c}_R, \boldsymbol{d}_R)$ obtained by dividing $[\boldsymbol{c}, \boldsymbol{d})$ in half with respect to the x direction. The children of these children are associated, in turn, with regions created by dividing these subdivisions in half in the y direction and so on. Requiring that each point lie inside of the region corresponding to the node where the point is stored completes the definition of the ADT. Therefore, when a new point is to be inserted, the tree is followed downward from the root until a blank node is found. At each level of the tree, only the left or right branch is pursued depending on whether the new point lies within the region associated with the left or right child. If the tree is empty, the new point is associated with the root.

Once the points have been sorted into a tree in this manner, searching through the tree requires $\mathcal{O}(\log N)$ operations. The search for points contained in a region $[\boldsymbol{a}, \boldsymbol{b}]$ progresses by traversing the tree recursively according to three steps: 1) test whether the point associated with the root is an element of $[\boldsymbol{a}, \boldsymbol{b}]$; 2) if there is a point associated with the left child and $[\boldsymbol{a}, \boldsymbol{b}] \cap [\boldsymbol{c}_L, \boldsymbol{d}_L) \neq \emptyset$, then traverse the left subtree; 3) if there is a point associated with the right child and $[\boldsymbol{a}, \boldsymbol{b}] \cap [\boldsymbol{c}_R, \boldsymbol{d}_R) \neq \emptyset$, then traverse the right subtree. In this manner entire subtrees can be eliminated with one test, leading to the stated efficiency.

Efficiency

In practice, several factors limit the efficiency of the search algorithm. For example, the cost reverts back to $\mathcal{O}(N_s)$ in the limit as $r \to \infty$. It is therefore desirable to minimize $r = \mathrm{d}(\boldsymbol{x}_2, \boldsymbol{x}_1) + \mathrm{d}(\boldsymbol{x}_1, S)$. On structured grids $\mathrm{d}(\boldsymbol{x}_2, \boldsymbol{x}_1)$ is minimized by taking $\boldsymbol{x}_1$ to be a neighbor of $\boldsymbol{x}_2$. (For the first point, when no neighbors have been calculated, a brute force search is needed to start the algorithm.) In general, it is worth the extra cost to choose the closest of the possible neighbors, especially for high-aspect-ratio grids and in cases where the solid boundaries can be in any direction. Because $\mathrm{d}(\boldsymbol{x}_1, S)$ is the distance to the surface from the neighboring point, it is likely that it is a good approximation to $\mathrm{d}(\boldsymbol{x}_2, S)$ and cannot be significantly reduced. As a result, the algorithm will be more efficient while $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are near a wall and become less efficient as $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ stray into the far field.

Another factor is the shape of the search region. Optimally, the search region should be a sphere of radius r centered at $\boldsymbol{x}_2$, which provides the minimum volume under the given constraints. In the context of the ADT, however, it is simpler to work with rectangular regions so that the search takes place in the cube $[\boldsymbol{a}, \boldsymbol{b}]$, where $a^i = x_2^i - r$, $b^i = x_2^i + r$, for $i = 1, 3$.

Finally, the outcome of the ADT sort directly influences the efficiency of the search. For N_s nodes the optimal tree has $\log_2(N_s)$ layers, but a highly degenerate tree can have $\mathcal{O}(N_s)$ layers. The efficiency of the sort is difficult to control. In general, acceptable efficiency will be obtained if the points are distributed throughout the domain in a reasonably uniform fashion.[16]

Results

The algorithm just described was tested on 10 structured three-dimensional CFD grids. The calculations were conducted on a Silicon Graphics Octane workstation using the MIPSpro f77 compiler with extensive optimization (−O2). The present examples are serial calculations, although parallelization is straightforward through domain decomposition.

The first grid ($N_v = 1{,}664{,}800$; $N_s = 53{,}264$) represented a rectangular water-tunnel test section with a circular cylinder and an airfoil-shaped strut running perpendicular to the flow. Wall proximity was calculated using a naïve search and the ADT method. The calculation required 7680 s in the former case and only 63 s in the latter. Similar results were achieved for eight other cases. The first nine cases share the trait that they are predominantly internal flows. Except for the first example just stated, all of the grids were for turbomachinery blades where the surfaces of the adjacent blades were included in the search for the nearest wall. Figure 1 shows a compilation of the results of the nine cases, comparing the CPU time required to find the distance to the nearest surface from all of the points in the volume using both the naïve and ADT methods. Over the range of cases, the ADT method is roughly two orders of magnitude faster than the naive method. The methods were also applied to one external flow case. In that case the CFD grid discretized the volume about the aft end of a marine vehicle with a propeller. The radius of the domain was approximately 20 times the radius of the vehicle. As a result, points in the far field needed to search nearly the entire surface, and whenever this is true the ADT search

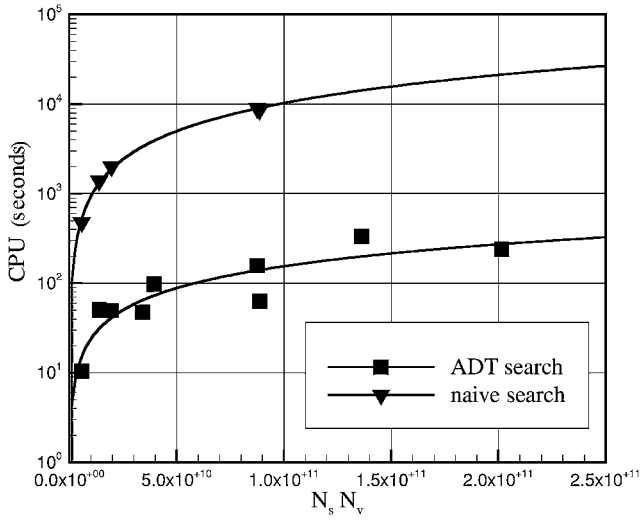

Fig. 1 Comparison of CPU time required to compute the distance from N_v points in a volume to the nearest of N_s surface points using an alternating digital tree search and a naïve search for nine internal flows.

becomes less efficient than an ordered search of the entire surface. The CPU time required by the ADT search was still lower than the naïve search but only by a factor of about four.

Conclusions

An $\mathcal{O}(N_v \log N_s)$ method for calculating wall proximity is derived by recasting the problem in a form that takes advantage of an alternating digital tree, an efficient geometric search algorithm. The method is easy to implement and is shown to reduce the CPU time by two orders of magnitude over the naive approach in the present examples.

References

[1]Baldwin, B. S., and Lomax, H., "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper 78-257, Jan. 1978.

[2]Chien, K.-Y., "Predictions of Channel and Boundary Layer Flows with a Low-Reynolds-Number Turbulence Model," *AIAA Journal*, Vol. 20, No. 1, 1982, pp. 33–38.

[3]Lai, Y. G., and So, R. M. C., "On Near-Wall Turbulent Flow Modelling," *Journal of Fluid Mechanics*, Vol. 221, 1990, pp. 641–673.

[4]Goldberg, U., Peroomian, O., and Chakravarthy, S., "A Wall-Distance-Free k–ε Model with Enhanced Near-Wall Treatment," *Journal of Fluids Engineering*, Vol. 120, No. 3, 1998, pp. 457–462.

[5]So, R. M. C., and Yuan, S. P., "A Geometry Independent Near-Wall Reynolds-Stress Closure," *International Journal of Engineering Science*, Vol. 37, No. 1, 1999, pp. 33–57.

[6]Craft, T. J., and Launder, B. E., "A Reynolds Stress Closure Designed for Complex Geometries," *International Journal of Heat and Fluid Flow*, Vol. 17, No. 3, 1996, pp. 245–254.

[7]Tucker, P. G., "Assessment of Geometric Multilevel Convergence Robustness and a Wall Distance Method for Flows with Multiple Internal Boundaries," *Applied Mathematical Modelling*, Vol. 22, No. 4–5, 1998, pp. 293–311.

[8]Tucker, P. G., "Prediction of Turbulent Oscillatory Flows in Complex Systems," *International Journal for Numerical Methods in Fluids*, Vol. 33, No. 6, 2000, pp. 869–895.

[9]Sethian, J. A., "Fast Marching Methods," *SIAM Review*, Vol. 41, No. 2, 1999, pp. 199–235.

[10]Buleev, N. I., "Theoretical Model of the Mechanism of Turbulent Exchange in Fluid Flows," Atomic Energy Research Establishment, AERE Translation 957, Harwell, England, U.K., May 1963.

[11]Launder, B. E., and Ying, W. M., "Prediction of Flow and Heat Transfer in Ducts of Square Cross-Section," *Proceedings of the Institution of Mechanical Engineers*, Vol. 187, Pt. 1, No. 37, 1973, pp. 455–461.

[12]Leslie, D. C., "Discussion of Prediction of Flow and Heat Transfer in Ducts of Square Cross-Section," *Proceedings of the Institution of Mechanical Engineers*, Vol. 187, Pt. 1, No. 37, 1973, pp. D147, D148.

[13]Gessner, F. B., and Po, J. K., "A Reynolds Stress Model for Turbulent Corner Flows, Part II: Comparisons Between Theory and Experiment," *Journal of Fluids Engineering*, Vol. 98, No. 2, 1976, pp. 269–277.

[14]Chima, R. V., and Yokota, J. W., "Numerical Analysis of Three-Dimensional Viscous Internal Flows," NASA TM 100878, July 1988.

[15]Bonet, J., and Peraire, J., "An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection Problems," *International Journal for Numerical Methods in Engineering*, Vol. 31, No. 1, 1991, pp. 1–17.

[16]Wang, Z. J., and Parthasarathy, V., "Fully Automated Chimera Methodology for Multiple Moving Body Problems," *International Journal for Numerical Methods in Fluids*, Vol. 33, No. 7, 2000, pp. 919–938.

R. M. C. So
Associate Editor

Improved Rhie–Chow Interpolation for Unsteady Flow Computations

Wen Zhong Shen,* Jess A. Michelsen,* and Jens Nørkær Sørensen*
Technical University of Denmark, DK-2800 Lyngby, Denmark

I. Introduction

FINITE difference and finite volume methods for solving the incompressible Navier–Stokes equations are based on either of two different grid categories: staggered grid or nonstaggered grid. Using staggered grids[1–3] typically results in stable and robust solutions.

For general nonorthogonal meshes, however, the staggered methods tend to become rather complex, either storing all velocity components on every cell face or introducing derivatives of the grid curvature through the Christoffel symbol.

To avoid these difficulties, several researchers, for example, see Ref. 4, investigated nonstaggered methods. These early schemes had no means of avoiding pressure oscillations arising from the use of 2δ differences of the pressure.

About 20 years ago, Rhie and Chow[5] proposed a procedure using a momentum-based interpolation for the cell face mass fluxes in the continuity equation that closely imitates the staggered practice by letting mass conservation be expressed in terms of mass fluxes across cell interfaces. The mass fluxes are driven by 1δ pressure differences across the faces. Hence, velocity–pressure decoupling cannot occur.

The Rhie–Chow procedure[5] gives excellent results for steady-state problems where a large local time step is used. Majumdar[6] improved the flux interpolation to avoid solution dependency on the underrelaxation parameter of the original Rhie–Chow interpolation.[5] For unsteady flow calculation, however, when using small time steps, pressure oscillations may still appear. This phenomenon was observed by Ferziger and Peric.[7] In the present paper, the origin of the pressure oscillations is analyzed, and a remedy is proposed.

II. Numerical Method

The incompressible Navier–Stokes equations are solved by a predictor–corrector method as follows.

A. Predictor Step

The momentum equations are discretized using a second-order backward differentiation scheme in time and second-order central differences in space, except for the convective terms that are discretized by the QUICK upwinding scheme. The resulting equations

Received 22 May 2001; revision received 14 August 2001; accepted for publication 17 August 2001.

*Associate Professor, Department of Mechanical Engineering, Building 403.