AIAA JOURNAL
Vol. 38, No. 12, December 2000

Unstructured Grid Arbitrarily Shaped Element Method for Fluid Flow Simulation

Yong G. Lai*
University of Iowa, Iowa City, Iowa 52242

A three-dimensional unstructured grid arbitrarily shaped element method is developed for solving the Navier–Stokes equations. The proposed method is generally applicable to arbitrarily shaped elements and, thus, offers the potential to unify many of the different grid topologies into a single formulation. Examples of grid topologies include a structured hexahedral mesh; unstructured tetrahedral mesh; hybrid mesh using a combination of hexahedra, tetrahedra, prisms, and pyramids; Cartesian mesh with elements (cubes) cut by flow boundaries; and mixed arbitrary multiblock meshes. The concept of such an arbitrary element method is developed and implemented into a pressure-based finite volume solver. It utilizes a collocated and cell-centered storage scheme. Expressions for second-order discretizations of the convection and diffusion terms are derived and presented for an arbitrarily shaped element. The developed code, as a first step, is applied to two selected three-dimensional viscous flows using a structured hexahedral mesh and an unstructured tetrahedral mesh. It is demonstrated that the concept of the arbitrarily shaped element method can be viable and efficient.

Introduction

UNSTRUCTURED grid technology has become very popular in recent years, mainly because it offers the possibility of a much reduced grid generation effort for flows with complex geometry. With tetrahedral elements, an unstructured grid also enjoys the advantage of flexible grid adaptation based on local flow features. These potential advantages, coupled with the need for solving inviscid flows around a complete aircraft, provided the first impetus to developing unstructured grid solvers. Some of the pioneering work includes that of Barth,[1] Jameson et al.,[2] Mavriplis,[3] Lohner et al.,[4] and Frink et al.,[5] among many others. Inspired by these earlier works, unstructured grid technology has since made significant advancements. Some of the recent research is reported in Refs. 6–10.

Despite its wide acceptance, the unstructured gird method with tetrahedral elements suffers from some shortcomings that have to be and are being addressed. One of the major disadvantages, compared to a structured grid, is that the use of highly stretched tetrahedral elements for boundary layers is very inefficient and difficult. An analytical study by Baker[11] even showed that additional errors could be induced on triangulated quadrilateral grids. For this reason, a hybrid approach using different shapes of elements was proposed by a number of investigators.[12–14] With the hybrid approach, prismatic or hexahedral elements can be used to resolve the boundary layer, and tetrahedral elements can be used to cover the rest of the domain. This way, highly stretched hexahedra or prisms are generated first near walls, and tetrahedra are then generated for the remaining flow domains. This approach worked quite well for a number of problems.[12–14] In another development, a Cartesian grid approach was proposed and demonstrated to solve the Euler and Navier–Stokes equations.[15,16] With a Cartesian grid, errors due to grid skew are minimized and grid generation is reduced to the task of identifying the element shape after a cube is intercepted or cut by flow boundaries. Of course, it is no easy task to develop such a cut algorithm for general three-dimensional flows, and one has to deal with boundary elements that may assume arbitrary shapes. Finally, an ideal and convenient approach is to use hybrid multiblock meshes. With hybrid multiblock capability, a complex problem can be divided into a number of blocks, and each block can be meshed with the element shape suitable for the block geometry. The difference from the regular hybrid method is that the mesh topology at the block interface does not have to be the same. With this approach, one has to deal with an arbitrary interface match between blocks. Again, a solver capable of handling arbitrarily shaped elements is viable in this kind of mesh.

It is seen from the preceding discussion that different shapes of mesh elements are being employed for the solution of Euler and Navier–Stokes equations. The relative advantages and disadvantages of different grid topologies have been the subject of debate for a number of years, and this debate will probably continue. The author feels that different grid topologies will probably coexist for awhile, and a consensus on the best meshing strategy may be quite difficult to reach. It is more probable that different grid topologies are suitable for different applications. It is clear, however, that the unstructured way of solving flow problems will definitely prevail, based on a structured grid being just a subset of an unstructured grid. For the same structured mesh, the difference between unstructured and structured methods lies mainly in the style of programming: an unstructured grid uses indirect addressing, whereas a structured grid uses direct addressing to access grid connectivity information. With modern day compiler technology, the penalty associated with indirect addressing is minimized and is not going to be an important factor.

In view of the preceding discussion, it is quite desirable that a unified unstructured grid method be developed so that the same solution algorithm and strategy can be applied to different grid strategies. To achieve this goal, the present study proposes to develop an arbitrarily shaped element method (ASEM). With this method, the same core part of the solver is applicable to most grid topologies currently in use. They include, for example, a structured grid (quadrilateral or hexahedral mesh); tetrahedron- or prism-based unstructured grid; hybrid or mixed-element method using combinations of hexahedra, tetrahedra, prisms, and pyramids; Cartesian grid method, which may have arbitrarily shaped elements due to boundary cut; and, finally, the hybrid multiblock mesh.

This paper reports a first step toward developing such an ASEM to solve fluid flows. To achieve the objective, the concept of an ASEM is developed and demonstrated with the pressure-based finite volume Navier–Stokes solver. The solver uses an element-centered storage scheme so that all dependent variables are located at element centers. In the following, the general numerical formulation that is applicable to arbitrarily shaped elements (or control volumes) is presented first for three-dimensional Navier–Stokes equations. Two-dimensional formulation can be similarly obtained or deduced from three-dimensional formulas. In addition, general expressions for calculating convective and diffusive fluxes on an arbitrarily shaped element face are derived and presented, which has not been reported in the past. Then some other numerical details are discussed. It is acknowledged that demonstration of the ASEM for the

Received 7 June 1999; revision received 27 March 2000; accepted for publication 5 May 2000.

*Associate Research Engineer, Iowa Institute of Hydraulic Research. Member AIAA.

whole spectrum of grid topologies mentioned is beyond the current scope of this paper. Instead, as a first step, the ASEM-based solver will be applied to three-dimensional flow problems with structured hexahedron elements and with unstructured tetrahedron elements to demonstrate the viability of the method. The development and demonstration of the method to other grid topologies will be subject of future study.

Numerical Method

Governing Equations

The present study deals with fluid flows governed by the Navier–Stokes equations in complex geometry. The fluid is assumed to be steady and incompressible for presentation convenience, although the method can be readily extended to unsteady and compressible flows. The mass and momentum equations can be expressed in vector form as

$$\nabla \cdot (\rho \boldsymbol{V}) = 0 \qquad (1)$$

$$\nabla \cdot (\rho \boldsymbol{V}\boldsymbol{V}) = -\nabla P + \nabla \cdot (\mu \nabla \boldsymbol{V}) \qquad (2)$$

Here ρ is the fluid density, $\boldsymbol{V}$ is the fluid velocity vector, μ is the fluid viscosity, and P is the static pressure.

Integration of Governing Equations

Numerical solution of fluid flows involves use of mesh covering the solution domain and the discretization of governing equations. As pointed out in the Introduction, the element shape of the mesh used in this study is assumed to be arbitrary. For a three-dimensional element, a polyhedron is used that may have any number of polygonal faces. For convenience, an element-centered scheme is used vs an alternative possibility of an element-vertex scheme. This means that all dependent variables are located at the centroid of the element instead of at the element vertices. Discretization of the governing equations is carried out using the finite volume approach so that the conservation property is preserved locally and globally. Without a loss of generality, consider a convection–diffusion equation representative of all governing equations, as follows:

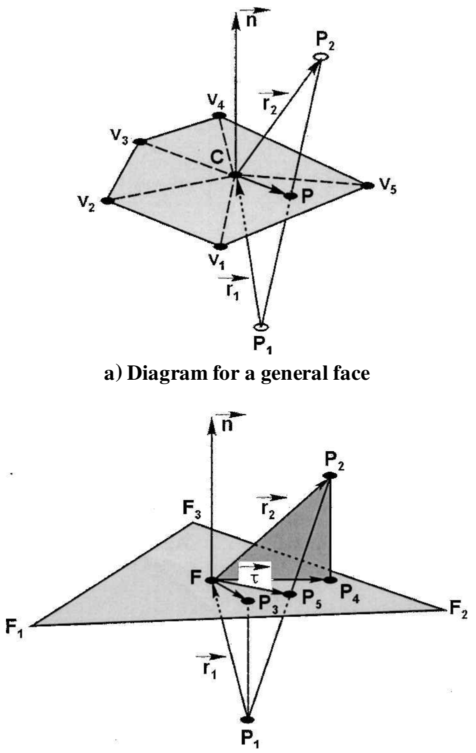
$$\nabla \cdot (\rho \boldsymbol{V} \Phi) = \nabla \cdot (\Gamma \nabla \Phi) + S_\Phi \qquad (3)$$

On integration over an element, the preceding equation can be discretized as

$$\sum_{\text{all faces}} (\rho_f V_f A)\Phi_f = \sum_{\text{all faces}} (\Gamma A \nabla \Phi \cdot \boldsymbol{n}) + S_\Phi \forall \qquad (4)$$

where $V_f = \boldsymbol{V} \cdot \boldsymbol{n}$ is the element face normal velocity component used to satisfy mass conservation, A is the element face area, Φ_f is the face value of the dependent variable Φ, $\boldsymbol{n}$ is the element face unit normal vector, and $\forall$ is the element volume. It is seen that the main task of discretization is to find appropriate expressions for convective and diffusive fluxes on each face of an element.

Before we proceed to flux calculation, note that the definition of the element face shape is uniquely defined in this study by a procedure described hereafter. Consider an element face with N vertices. A sample polygonal face is depicted in Fig. 1a. The face is not completely defined yet because all vertex points may not be on the same plane. To define the face, a center point C of the face is found first by taking the arithmetic average of all vertex position vectors. N triangles are then formed by connecting point C to all vertices. When we take the face in Fig. 1a as an example, five triangles, V_1V_2C, V_2V_3C, V_3V_4C, V_4V_5C, and V_5V_1C, are formed. A collection of these triangles uniquely defines the shape of the face, and thereby the element shape is also defined. All geometric quantities such as the element volume, face area, and face unit normal vector can be calculated based on the preceding definition. The use of an exactly defined element shape is consistent with the essence of the finite volume method and is also important to preserve the conservation property.



a) Diagram for a general face

b) Diagram for a triangular face

Fig. 1 Illustration of an arbitrary element face.

Cell Face Diffusive Flux

First, let us consider diffusive flux at an element face, that is, the term $\nabla\Phi \cdot \boldsymbol{n}$. Without a loss of generality, consider only a triangular face as shown in Fig. 1b (note that a general face consists of triangles). The triangular face has vertex points F_1, F_2, and F_3, with unit normal of $\boldsymbol{n}$ and center point F. It is assumed that Φ at two neighboring element centers, P_1 and P_2, and at face vertices are known. The diffusion term can be calculated by locally establishing a general nonorthogonal coordinate system on the face with its three covariant coordinate bases ε_1, ε_2, and ε_3. Here ε_1 is chosen as the vector from point P_1–P_2, and ε_2 and ε_3 are on the element face. Then, a gradient can be expressed in this local coordinate system as

$$\nabla\Phi = \frac{\partial\Phi}{\partial\xi}\varepsilon^1 + \frac{\partial\Phi}{\partial\eta}\varepsilon^2 + \frac{\partial\Phi}{\partial\zeta}\varepsilon^3 \qquad (5)$$

where ξ, η, and ζ are coordinates along ε_1, ε_2, and ε_3, respectively; and ε^1, ε^2, and ε^3 are three contravariant coordinate bases. Therefore, $\nabla\Phi \cdot \boldsymbol{n}$ can be split into a normal term and a cross term. The normal term is the first term on the right-hand side of Eq. (5), and it is approximated by the difference of Φ values between points P_1 and P_2. The cross term consists of the last two terms on the right-hand side of Eq. (5) and is expressed as an area integral on the face. It can be shown that the area integral of the cross term can be further transformed to a line integral along the perimeter of the face using Green's theorem. Once discretized for a triangular face, the diffusion term for an arbitrary face is obtained through summation over all triangles. Some mathematical manipulation leads to the following for the diffusion term on an arbitrary element face:

$$\nabla\Phi \cdot \boldsymbol{n}A = D_n(\Phi_2 - \Phi_1) + \sum_{\text{all edges}} D_c^{\text{edge}}\Phi_{\text{edge}} \qquad (6)$$

$$D_n = \frac{A}{(\boldsymbol{r}_1 + \boldsymbol{r}_2) \cdot \boldsymbol{n}}, \qquad D_c^{\text{edge}} = -\frac{(\boldsymbol{r}_1 + \boldsymbol{r}_2) \cdot (\delta\boldsymbol{r}_{\text{edge}} \times \boldsymbol{n})}{(\boldsymbol{r}_1 + \boldsymbol{r}_2) \cdot \boldsymbol{n}} \qquad (7)$$

In the preceding equations,

$$\sum_{\text{all edges}}$$

stands for summation over all edges of the face, Φ_{edge} is the Φ value at the edge center and is calculated using vertex values, $\delta\boldsymbol{r}_{\text{edge}}$ is the distance vector of the edge, and $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are vectors from P_1 to C and C to P_2 (see Fig. 1a). The cross term in the preceding expression can be shown to have second-order accuracy by the use of Taylor series expansion, whereas the normal term is second order only if the vectors from P_1 to C and C to P_2 are equal. Otherwise, the normal term discretization error is proportional to the difference of the two distances that is nonzero due to nonuniform and nonregular meshes. Note that the normal and cross diffusion coefficients at each face involve only geometric quantities and that they are calculated only once in the beginning because the mesh does not change during the solution process.

Cell Face Value Calculation

Next, let us consider calculation of cell face value with cell-center quantities given. This center-to-face operation is used very often during the solution process. In the following, a second-order accurate expression is derived and presented. Again, refer to Fig. 1b for a triangular face $F_1F_2F_3$ with two neighboring element center points P_1 and P_2. Define point P_5 such that it is the intercept point between line P_1P_2 and the face; also define points P_3 and P_4 on the face such that P_1P_3 and P_2P_4 are normal to the face. Then a second-order accurate interpolation expression for point P_5 is given by

$$\Phi_{P_5} = \frac{\delta_1\Phi_2 + \delta_2\Phi_1}{\delta_1 + \delta_2} \tag{8}$$

where $\delta_1 = \boldsymbol{r}_1 \cdot \boldsymbol{n}$ and $\delta_2 = \boldsymbol{r}_2 \cdot \boldsymbol{n}$. Φ_{P_5} obtained here has often been used to approximate the value at center F. In general, however, this does not guarantee second-order accuracy unless $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are parallel. A truly second-order expression is derived using Taylor expansion as

$$\Phi_f = \Phi_{P_5} - \nabla_\tau\Phi \cdot \boldsymbol{\tau} \tag{9}$$

where $\boldsymbol{\tau}$ is the distance vector from F to P_5, and ∇_τ is the gradient operator defined on the face. With some geometric and mathematical manipulation, Eq. (9) can be written as

$$\Phi_f = \Phi_{P_5} - \frac{\delta_1\boldsymbol{r}_2 - \delta_2\boldsymbol{r}_1}{(\delta_1 + \delta_2)A}\sum(\delta\boldsymbol{r}_{\text{edge}} \times \boldsymbol{n})\Phi_{\text{edge}} \tag{10}$$

It can be shown with Taylor expansion that the preceding discretization has second-order accuracy.

The Φ_f value of the convective term in Eq. (4) needs further discussion. If the second-order central difference scheme is applied directly, spurious oscillation may occur for flows with a high enough cell Peclet number. Therefore, damping is added to the second-order scheme in a manner similar to the concept of artificial viscosity. The damped convective scheme is derived as follows:

$$\Phi_f = \Phi_f^{\text{CN}} + d\left(\Phi_f^{\text{UP}} - \Phi_f^{\text{CN}}\right) \tag{11}$$

where

$$\Phi_f^{\text{UP}} = \tfrac{1}{2}(\Phi_1 + \Phi_2) - \tfrac{1}{2}\text{sign}(V_f)(\Phi_2 - \Phi_1) \tag{12}$$

Φ_f^{CN} is the second-order interpolation scheme as presented, and d represents the amount of damping used ($d = 0.1$ is used for all calculations in this study).

With diffusion and convection terms discretized, the final discretized governing equation for an element P can be organized concisely as the following linear equation:

$$A_P\Phi_P = \sum_{\text{nb}} A_{\text{nb}}\Phi_{\text{nb}} + S_\Phi^* \tag{13}$$

where nb refers to all neighbor elements that share the same vertices with element P.

Mass Conserving Face Velocity

For a nonstaggered grid, a special procedure is required to obtain the element face normal velocity used to enforce mass conservation. Otherwise, the well-known checkerboard instability may occur. In this study, the procedure proposed by Rhie and Chow[17] and Peric et al.[18] is adopted. That is, the face normal velocity is obtained by averaging the momentum equation from element centers to element faces. A detailed derivation is omitted, and interested readers are referred to previous papers, for example, Refs. 17–19. It is sufficient to point out here that the element face normal velocity is calculated as

$$V_f = \langle \boldsymbol{V}\rangle \cdot \boldsymbol{n} + \langle \forall / A_P\rangle[\langle \nabla P\rangle \cdot \boldsymbol{n} - \nabla P \cdot \boldsymbol{n}] \tag{14}$$

where $\langle\ \rangle$ stands for the averaging operator from the element center to face and $\forall$ is the element volume. When the averaging operator is applied to a vector, it implies application to each Cartesian component of the vector.

Pressure Correction Equation

The pressure correction equation is derived from the mass conservation equation using the PISO algorithm.[20] PISO is basically a predictor–corrector method, and it can be described as follows.

Predictor Step

With a known pressure field P^o, a new velocity field $\boldsymbol{V}^*$ is predicted by solving the momentum equation

$$A_P\boldsymbol{V}^* = H\left(\boldsymbol{V}_{\text{nb}}^*\right) - a\nabla P^o + S_V \tag{15}$$

where H stands for a linear operator and

$$H = \sum_{\text{nb}} A_{\text{nb}}$$

First Corrector Step

Next, new pressure and velocity fields P^* and $\boldsymbol{V}^{**}$ are sought such that following two equations are satisfied:

$$\nabla \cdot \boldsymbol{V}^{**} = 0 \tag{16}$$

$$A_P\boldsymbol{V}^{**} = H\left(\boldsymbol{V}_{\text{nb}}^*\right) - a\nabla P^* + S_V \tag{17}$$

The pressure correction equation is obtained from this as

$$\nabla \cdot (a/A_P)\nabla P' = \nabla \cdot \boldsymbol{V}^* \tag{18}$$

with $P' = P^* - P^o$. The pressure correction equation is solved to obtain new fields of P^* and $\boldsymbol{V}^{**}$.

Second Corrector Step

One more corrector is performed to obtain new fields of P^{**} and $\boldsymbol{V}^{***}$ so that the following two equations are satisfied:

$$\nabla \cdot \boldsymbol{V}^{***} = 0 \tag{19}$$

$$A_P\boldsymbol{V}^{***} = H\left(\boldsymbol{V}_{\text{nb}}^{**}\right) - a\nabla P^{**} + S_V \tag{20}$$

The pressure correction equation is derived as

$$\nabla \cdot (a/A_P)\nabla P' = \nabla \cdot \boldsymbol{V}^{**} + H\left(\boldsymbol{V}_{\text{nb}}^{**} - \boldsymbol{V}_{\text{nb}}^*\right) \tag{21}$$

with $P' = P^{**} - P^*$. The preceding equation is solved to obtain new corrected fields of P^{**} and $\boldsymbol{V}^{***}$.

Note that more corrector steps can be carried out if necessary but that two correctors are usually sufficient.

Data Structure

For an unstructured grid approach, the design of a data structure is important, and it is discussed hereafter. In this study, three across-the-field operations are encountered. The most frequently used operation is a loop over all elements. Therefore, the element-based data storage is used because it is natural to the element-centered storage scheme. Operations such as property update and linear equation solvers are all element based, and they represent a major portion of the CPU time. With element-based data, connectivity integer arrays are created that address mesh relations from element to neighboring elements and from elements to element faces. The second type of data structure is face based and is created to compute the convective and diffusive fluxes and the associated coefficients. A face-based data structure requires the creation of a connectivity array that provides information from the face to the neighboring elements. Finally, the vertex-based data structure, which provides information from a vertex to all neighboring elements is also used. A vertex-based data structure is used to compute vertex values from the element center values of a variable and is needed by the cross diffusion term and the pressure correction equation. The face-based and vertex-based data structures are less frequently used and consume much less computing time. Therefore, the element-based data structure dominates the efficiency of the proposed method. In terms of memory requirement, the proposed method requires less than 1000 bytes per element (with 64 bit floating point precision and 32 bit integer) based on a wide range of practical three-dimensional turbulent flows simulated.

Boundary Conditions

The boundary conditions encountered in this study are related to the treatment at the inlet, outlet, no-slip wall, and symmetric boundary. At an inlet and a no-slip wall, the Cartesian velocity components are specified at boundary element face centers, and the pressure is extrapolated from the interior. They are needed to solve the momentum equations. The solution of the pressure correction equation, however, requires no pressure boundary condition because mass fluxes on these boundaries are specified and unchanged during the solution process. At an outlet, pressure is specified at the element face center, and Cartesian velocity components are extrapolated from the interior for the solution of momentum equations. For the pressure correction equation, the pressure increment is set to be zero at the outlet because pressure should not change during the solution. Finally, at a symmetry boundary, the velocity component normal to the boundary is set to zero whereas tangential components and pressure are extrapolated from the interior for the momentum equations. For the pressure correction equation, no pressure condition is needed at the symmetry boundary as zero mass flux is enforced.

Summary of Solution Process

All governing equations are solved sequentially, that is, they are solved in an equation-by-equation manner. For a typical steady-state simulation, momentum equations are solved first assuming a known pressure field. This represents the predictor step of the PISO algorithm. The predicted velocity field is then used to calculate the element face normal velocity with Eq. (14). This velocity will not usually satisfy mass conservation. Therefore, corrector steps of the PISO algorithm are applied by solving pressure correction equations (18) and (21). After each solution of the pressure correction equation, new pressure and velocity fields are obtained. The given solution process completes one iteration of the solution cycle, and it is repeated until a preset convergence criterion is met. To establish convergence criterion, residuals for continuity and other equations are computed as follows:

$$R_p = \frac{\sum_{\text{all elements}} |\nabla \cdot V|}{\text{Mass Residual Scale}}$$

$$R_\Phi = \frac{\sum_{\text{all elements}} \left| A_P \Phi_P - \sum_{\text{nb}} A_{\text{nb}} \Phi_{\text{nb}} + S_\Phi^* \right|}{\sum_{\text{all elements}} |A_P \Phi_P|} \tag{22}$$

where summation is over all elements of the mesh and the mass residual scale is selected to be the maximum mass imbalance in the first five iterations. The preceding definition guarantees that the mass and momentum conservation equations are satisfied to the preset convergence criterion. In general, convergence is considered to be achieved when R_p and R_Φ are below 10^{-3}. In this paper, however, all calculations are performed such that residuals for mass and u and v equations are below 10^{-6} to demonstrate the robustness of the solver.

Note that the preceding solver requires the solution of sparse linear equations. A direct solver is impractical because of the excessive demand on computer resources. However, the choice of iterative solvers is quite limited for the general unstructured grid method. In this study, the preconditioned conjugate gradient method is used if the mesh is unstructured, whereas the strongly implicit procedure (SIP) of Stone[21] is employed if the mesh is structured or block structured. As mentioned in the Introduction, a structured grid is embedded in an unstructured grid code and a hexahedron element is treated as a special element shape.

Validation and Discussion

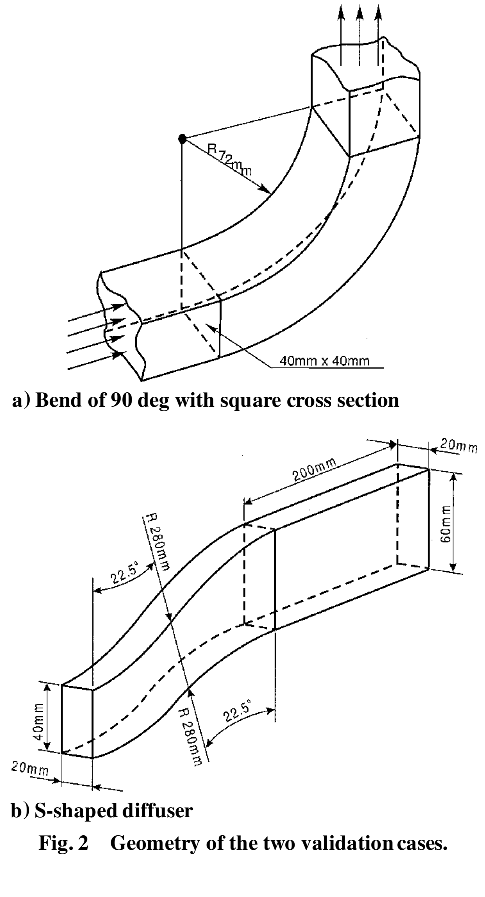
Despite that the ASEM is applicable to many grid topologies, a test and a demonstration of all grid topologies are beyond the scope of this study. As a first step, only pure tetrahedron elements or structured hexahedron elements are used. The demonstration and the validation of proposed ASEM to other grid topologies are under investigation and will be reported in the future.

Test Case Description

Two three-dimensional viscous flows are chosen to evaluate and validate the proposed unstructured grid method. The first is a laminar flow through a 90-deg square duct that was experimentally measured by Humphrey et al.[22] The second three-dimensional case is the flow in an S-shaped diffuser with rectangular cross section that was experimentally studied by Rojas et al.[23] Both cases have also been studied numerically by other researchers[22,24,25] and, therefore, can be used for comparison purposes.

The 90-deg square cross-sectioned bend flow was experimentally studied by Humphrey et al.[22] and the geometry is illustrated in Fig. 2a. As shown, the bend has a cross section of 40 × 40 mm and a mean radius of 92 mm. The experiment was carried out at a Reynolds number, based on the hydraulic diameter D_h and bulk velocity, of 7.9×10^2 (corresponding to a Dean number of 368). The computational domain consists of three sections: an entrance with a straight duct $10D_h$ in length, a 90-deg curved bend, and an exit with a straight duct $10D_h$ in length. Because of the symmetry of the problem, only one-half of the duct is calculated.

The second case simulated is the flow in an S-shaped diffuser with rectangular cross section and the geometry is displayed in Fig. 2b. The case was experimentally studied by Rojas et al.[23] It is seen that the diffuser cross section is a 40 × 40 mm square at the inlet. It is



a) Bend of 90 deg with square cross section

b) S-shaped diffuser

Fig. 2 Geometry of the two validation cases.

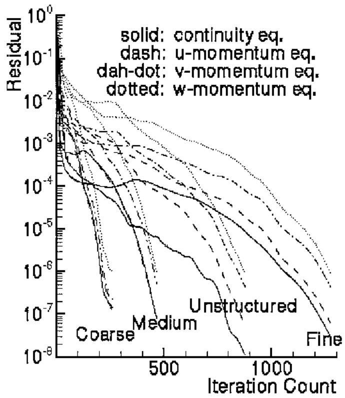


Fig. 3 Convergence plots of three hexahedral meshes and one tetrahedral mesh.

then expanded linearly in the plane of curvature to a rectangle of 60 × 40 mm at the exit. The S curvature is formed from two 22.5-deg bends of 280-mm mean radius of curvature. Note that a straight duct with cross section of 60 × 40 mm and length of 200 mm is attached to the exit of the diffuser. Because of symmetry, only one-half of the diffuser needs to be calculated. The calculation corresponds to the flow Reynolds number of 7.9×10^2 based on the hydraulic diameter at the inlet and the mean bulk velocity.

Evaluation of Convergence and Accuracy

Before proceeding to validation, an effort was made to investigate the order of accuracy, convergence, and efficiency of the solver using the 90-deg square duct case. For the purpose, three hexahedral meshes were generated with meshes of $46 \times 20 \times 10$, $92 \times 40 \times 20$, and $184 \times 80 \times 40$, respectively (designated as coarse, medium, and fine grids). The grid spacing within the bend is quite uniform, whereas grids in the entrance and exit are nonuniformly distributed. All three computations used the same relaxation parameters and the same initial conditions (zero velocity and pressure are used everywhere). The convergence plots of three hexahedral meshes are shown in Fig. 3. CPU times per iteration on the SGI Octane workstation are 0.888, 9.43, and 87.3 s for the coarse, medium, and fine grids, respectively. The finest mesh has 588,800 elements, and six orders of error reduction required about 31 h on the SGI octane workstation and about 26 h on a Pentium III 500 MHz personal computer on a single processor and with double precision.

Formal Taylor series analysis indicates that the solver is approximately second order as discussed in the preceding section. However, the actual accuracy will be less than second order due to the use of the damping in the convective flux as well as the accuracy of a normal diffusion term on nonregular mesh. To quantify the accuracy for the case studied, the approach suggested by Ferziger and Peric[26] is used. This approach requires solutions on three different meshes with finer mesh obtained by doubling the mesh elements of the coarser mesh in each direction. The three meshes generated earlier serve this purpose. The order of accuracy can be estimated using the $L1$ or $L2$ norms as follows:

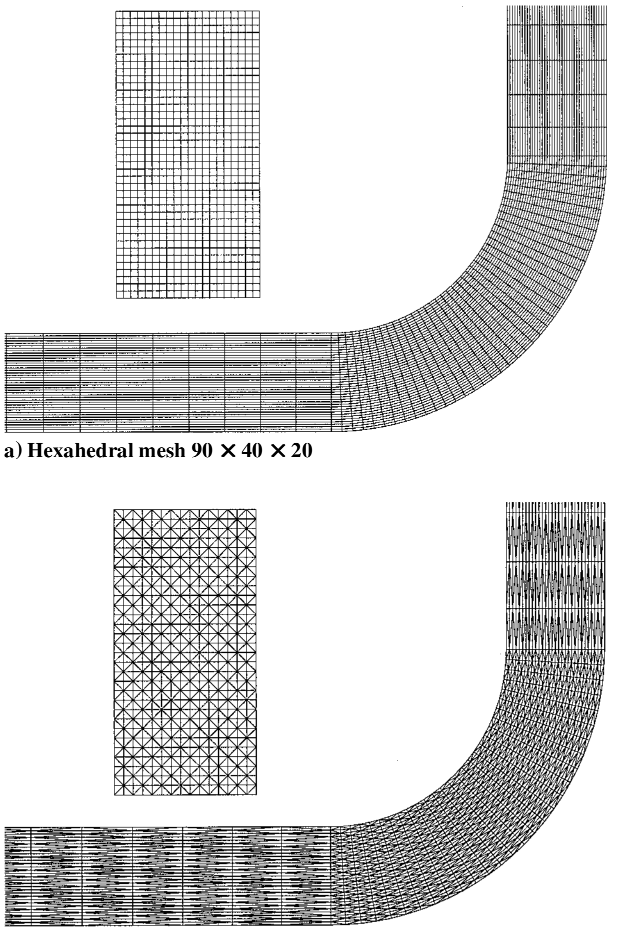
$$p_1 = \log \frac{\sum_{4h} |\Phi_{4h} - \Phi_{2h}|/N_{4h}}{\sum_{2h} |\Phi_{2h} - \Phi_h|/N_{2h}} \Bigg/ \log 2$$

$$p_2 = \log \frac{\sqrt{\sum_{4h} (\Phi_{4h} - \Phi_{2h})^2/N_{4h}}}{\sqrt{\sum_{2h} (\Phi_{2h} - \Phi_h)^2/N_{2h}}} \Bigg/ \log 2 \qquad (23)$$

where Φ_{4h}, Φ_{2h}, and Φ_h are solutions of variable Φ at coarse, medium, and fine meshes, respectively; N_{2h} and N_{4h} are the number of grid points on medium and coarse meshes; and

$$\sum_{2h}, \qquad \sum_{4h}$$

are summations over all medium and coarse mesh points. With the preceding equations, the estimated order of accuracy is $P_1 = 1.79$ and $p_2 = 1.83$ for u velocity and $p_1 = 1.85$ and $p_2 = 1.95$ for v velocity, respectively. It is seen that the order of accuracy is less than, but close to, 2.0.



a) Hexahedral mesh 90 × 40 × 20

b) Tetrahedral mesh with 144,000 elements

Fig. 4 Cross-sectional view of meshes at the inlet and on symmetry plane.

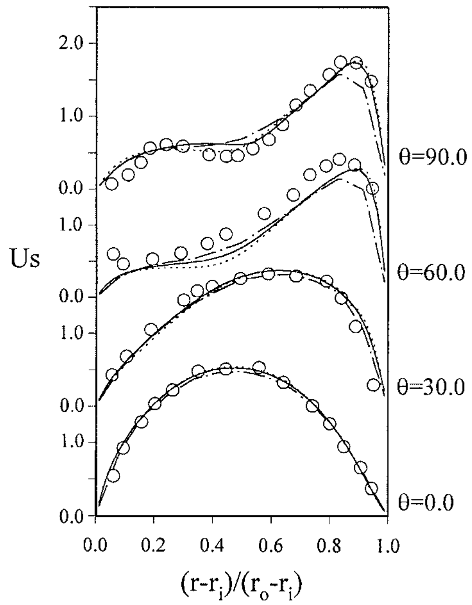
Bend of 90 Degrees with Square Cross Section

Three meshes are generated for validation purposes, one with structured hexahedral elements (designated as HEX) and two with tetrahedral elements (designated as TET1 and TET2). HEX mesh has mesh of $90 \times 40 \times 20$ (72,000 elements), where 90 cells are along the streamwise, 40 cells in the radial, and 20 cells in the spanwise direction. This structured mesh is comparable to previous numerical studies, for example, $60 \times 15 \times 10$ by Humphrey et al.[22] and $92 \times 33 \times 18$ by Kim and VanOverbeke.[25] TET1 is a very coarse tetrahedral mesh with about 13,440 elements and 3432 vertices, whereas TET2 is a finer tetrahedral mesh with 144,000 elements and 32,240 vertices. Plots of some sections of the mesh are shown in Fig. 4.

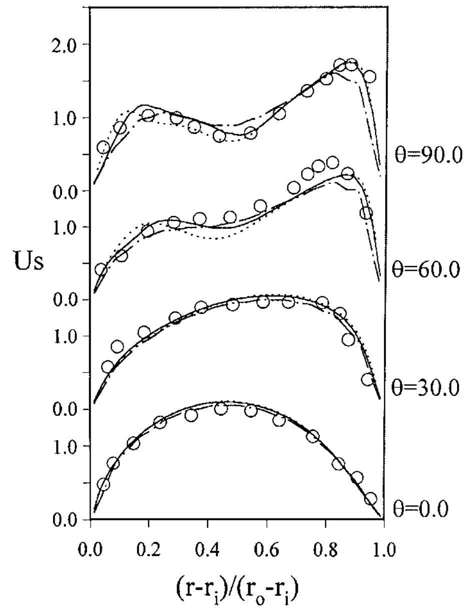
The flow inlet is located at $10D_h$ upstream of the bend section, and the velocity profile of a fully developed square duct with the same cross section is prescribed at the inlet. The fully developed solution is obtained from another straight duct simulation, and the same solution is imposed at the inlet for all three meshes. At the exit, the static pressure is assumed to be constant, and all other dependent variables are extrapolated from interior.

Just for comparison purposes, the convergence history with the TET2 mesh is also shown in Fig. 3 together with other hexahedral meshes. In terms of convergence and efficiency, all three calculations obtained six orders of error reduction within 800 iterations. The CPU times per iteration are 9.2 and 15.7 s for the HEX and TET2 meshes, respectively.

Calculated streamwise velocity profiles in the radial direction are shown in Fig. 5 for two spanwise planes and four streamwise locations, and they are also compared to the experimental data. Note



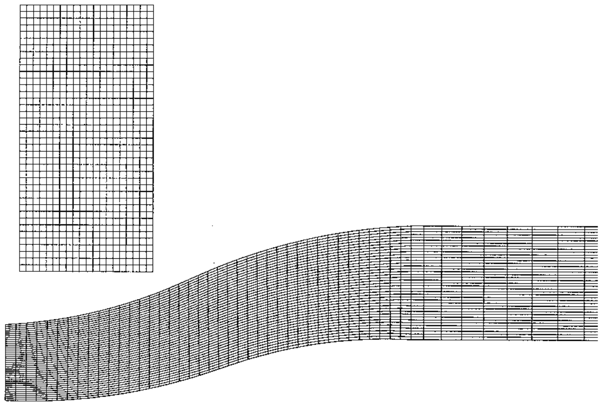
a) On symmetry plane ($Z/D_h = 0.5$)



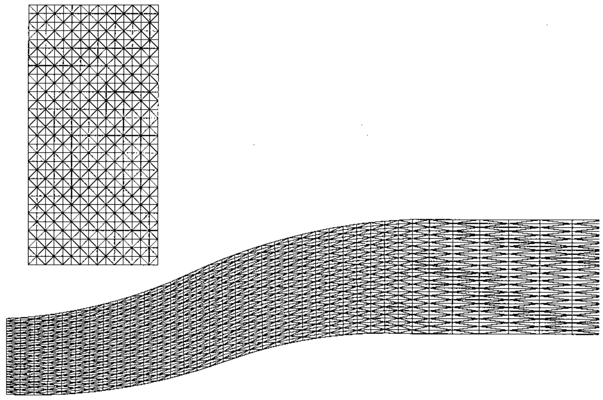
b) On quarter plane ($Z/D_h = 0.25$)

Fig. 5 Calculated and measured streamwise velocity profiles for 90-deg bend: ○, measured data[22]; ——, TET2; - - -, HEX; and - -, TET1.

that r_i and r_o in Fig. 5 refer to the inner and outer radii of the bend, U_s is the streamwise velocity, and θ is the angle in degrees of the curved bend section. Figure 5a is for results on the symmetry plane ($z/D_h = 0.5$, with $z/D_h = 0.0$ being the wall), and Fig. 5b is on the plane of $z/D_h = 0.25$. From these results, it is seen that all three meshes obtained very similar results in the earlier part of the bend (e.g., $\theta = 0$ and 30 deg) and that they all compare well with measured data. This is because the influence of bend curvature is not very significant initially. However, the three sets of results start to deviate from each other in the later part of the bend, that is, at $\theta = 60$ and 90 deg). Overall, the results from the TET2 mesh have the best agreement with the data, followed by the HEX mesh, and then the TET1 mesh. This finding is not surprising because TET2 has the greatest number of elements, whereas TET1 has the smallest number. Also note that results from the HEX mesh are very close to those obtained by Kim and VanOverbeke,[25] who used a structured hex mesh and a similar number of elements. Overall, it is demonstrated that both tetrahedron- and hexahedron-based meshes work quite well in terms of accuracy and efficiency, at least for the flow considered.



Hexahedral mesh 55 × 40 × 20



Tetrahedral mesh with 94,500 elements

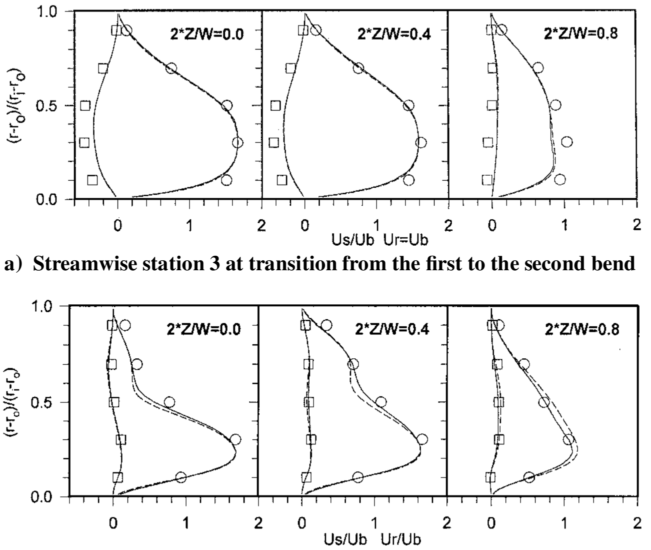
Fig. 6 Cross-sectional view of meshes at the inlet and on symmetry plane.

Flow in an S-Shaped Diffuser

Two meshes are generated for the validation study of the S-shaped diffuser flow: one with structured hexahedral elements and another with unstructured tetrahedral elements. The structured mesh uses a 55 ×40 ×20 grid system, which gives about 44,000 elements, whereas the unstructured mesh uses 94,500 tetrahedron elements. For the structured grid, 55 cells are in the streamwise direction, 40 cells in radial direction, and 20 cells in spanwise direction. This mesh compares to the mesh of 32 ×30 ×14 used by Majumdar et al.[24] for their computation. Plots of some sections of the two meshes are displayed in Fig. 6.

The boundary conditions are specified as follows. At the diffuser inlet, measured streamwise and radial velocity components are imposed. Spanwise velocity is set to zero at the inlet because it is quite small according to experiment. The exit of the solution domain is located 200 mm downstream of the diffuser exit. At the exit, pressure is assumed to be constant, whereas all other dependent variables are extrapolated.

Figure 7 displays calculated transverse profiles of the streamwise and radial velocity components on three spanwise planes and two streamwise cross sections. The measured data are also displayed in Fig. 7. Figure 7a shows results at streamwise station 3, which is located at the transition from the first to the second bend, whereas Fig. 7b is for results at station 5 at the end of the second bend. In Fig. 7, r_i and r_o are the inner and outer radii of each bend, U_s and U_r are the streamwise and radial velocity components, U_b is the mean bulk velocity, W is the width of the diffuser (wall-to-wall spanwise distance), and Z is the spanwise coordinate measured from the wall to the symmetry plane. It is seen that the agreement between the computation and the data is satisfactory. On the other hand, results from the tetrahedral mesh compare better with the experiment than the structured mesh. This is believed to be attributable to more elements being used for the tetrahedral elements. Also, it is found through inspection that the calculated results agree well with the computational results obtained by Majumdar et al.[24]



a) Streamwise station 3 at transition from the first to the second bend

b) Streamwise station 5 at the exit of the second bend

Fig. 7 Calculated and measured streamwise and radial velocity profiles for S-shaped diffuser: ○, measured data[23]; ——, tetrahedral mesh; and - - -, hexahedral mesh.

In terms of convergence and efficiency, hexahedral (44,000 elements) and tetrahedral (94,500 elements) meshes need 809 and 1218 iterations, respectively, to achieve six orders of error reduction in mass and momentum conservation, and the CPU times per iteration are 6.69 and 9.43 s, respectively.

Conclusions

An unstructured ASEM is developed to solve three-dimensional Navier–Stokes equations. The proposed ASEM is attractive in that it has the potential to unify many of the currently used grid topologies into one formulation. Different grid topologies could include a structured hexahedral mesh; unstructured tetrahedral mesh; hybrid or mixed element mesh, which uses a combination of hexahedra, tetrahedra, prisms, and pyramids; the Cartesian grid approach, which uses arbitrary cubes cut by the boundaries; and hybrid multiblock meshes.

The concept of the ASEM is implemented into the pressure-based finite volume numerical method to solve three-dimensional incompressible viscous flows. In the implementation, the element-centered storage scheme is adopted and a PISO algorithm is used for velocity and pressure coupling. General expressions are derived and presented for the convective and diffusive fluxes at each element face with an arbitrary polygonal shape. The developed ASEM solver is applied both to the structured hexahedral mesh and the unstructured tetrahedral mesh as a first-step validation and demonstration. In the future, the ASEM-based solver will be applied to other grid topologies.

Based on this study, it is demonstrated that the ASEM can be developed to solve fluid flows. A code developed as such is quite flexible, so that it can be used to solve flow problems with different grid topologies. Selected three-dimensional flow calculations demonstrated that the developed method is viable, accurate, and efficient.

Acknowledgments

This work was partially sponsored by Southwest Research Institute, and the author would like to express his appreciation to H. S. Zhang for his enthusiasm on the project.

References

[1] Barth, T., "Three-Dimensional Upwind Euler Solver for Unstructured Meshes," AIAA Paper 91-1548, June 1991.

[2] Jameson, A., Baker, T. J., and Weatherill, N. P., "Calculation of Inviscid Transonic Flow Over a Complete Aircraft," AIAA Paper 86-0103, Jan. 1986.

[3] Mavriplis, D. J., "Accurate Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes," *AIAA Journal*, Vol. 28, No. 2, 1990, pp. 213–221.

[4] Lohner, R., Morgan, K., Peraire, J., and Vahdati, M., "Finite Element Flux-Corrected Transport (FCT) for the Euler and Navier–Stokes Equations," *International Journal for Numerical Methods in Fluids*, Vol. 7, No. 10, 1987, pp. 1093–1109.

[5] Frink, N. T., Parikh, P., and Pirzadeh, S., "Fast Upwind Solver for the Euler Equations on Three-Dimensional Unstructured Meshes," AIAA Paper 91-0102, Jan. 1991.

[6] Haselbacher, A., McGuirk, J. J., and Page, G. J., "Finite Volume Discretization Aspects for Viscous Flows on Mixed Unstructured Grids," *AIAA Journal*, Vol. 37, No. 2, 1999, pp. 177–184.

[7] Mavriplis, D. J., "On Convergence Acceleration Techniques for Unstructured Meshes," AIAA Paper 98-2966, July 1998.

[8] Kim, S. E., Mathur, S. R., Murthy, J. Y., and Choudhury, D., "Reynolds-Averaged Navier–Stokes Solver Using Unstructured Mesh-Based Finite Volume Scheme," AIAA Paper 98-0231, Jan. 1998.

[9] Frink, N. T., "Tetrahedral Unstructured Navier–Stokes Method for Turbulent Flows," *AIAA Journal*, Vol. 36, No. 11, 1998, pp. 1975–1982.

[10] Lai, Y. G., "An Unstructured Grid Method for a Pressure-Based Flow and Heat Transfer Solver," *Numerical Heat Transfer*, Pt. B, Vol. 32, Dec. 1997, pp. 267–281.

[11] Baker, T. J., "Discretization of Navier–Stokes Equations and Mesh-Induced Errors," *Proceedings of 5th International Conference on Numerical Grid Generation in Computational Field Simulation*, Mississippi State Univ., Mississippi State, MS, 1996, pp. 209–218.

[12] Ward, S., and Kallinderis, Y., "Hybrid Prismatic Tetrahedral Grid Generation for Complex Three-Dimensional Geometries," AIAA Paper 93-0669, Jan. 1993.

[13] Mavriplis, D. J., and Venkatakrishnan, V., "A Unified Multigrid Solver for the Navier–Stokes Equations on Mixed Element Meshes," Inst. for Computer Applications in Science and Engineering, Rept. 95-53, Hampton, VA, 1995.

[14] Coirier, W. J., and Jorgenson, P. C. E., "Mixed Volume Grid Approach for the Euler and Navier–Stokes Equations," AIAA Paper 96-0762, Jan. 1996.

[15] Melton, J. E., Berger, M. J., Aftosmis, M. J., and Wong, M. D., "Three-Dimensional Applications of a Cartesian Grid Euler Method," AIAA Paper 95-0853, Jan. 1995.

[16] Coirier, W. J., and Powell, K. G., "Cartesian, Cell-Based Approach for Adaptively Refined Solutions of the Euler and Navier–Stokes Equations," AIAA Paper 95-0566, Jan. 1995.

[17] Rhie, C. M., and Chow, W. L., "Numerical Study of the Turbulent Flow Past an Airfoil with Trailing-Edge Separation," *AIAA Journal*, Vol. 21, No. 11, 1983, pp. 1526–1532.

[18] Peric, M., Kessler, R., and Scheuerer, G., "Comparison of Finite-Volume Numerical Methods with Staggered and Collocated Grids," *Computers and Fluids*, Vol. 16, No. 4, 1988, pp. 389–403.

[19] Lai, Y. G., So, R. M. C., and Przekwas, A. J., "Turbulent Transonic Flow Simulation Using a Pressured-Based Method," *International Journal in Engineering Science*, Vol. 33, No. 4, 1995, pp. 469–483.

[20] Issa, R. I., "Solution of the Implicitly Discretized Fluid Flow Equations by Operator-Splitting," *Journal of Computational Physics*, Vol. 62, No. 1, 1986, pp. 40–65.

[21] Stone, H. L., "Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations," *SIAM Journal on Numerical Analysis*, Vol. 5, No. 5, 1968, pp. 530–558.

[22] Humphrey, J. A. C., Taylor, A. M. K., and Whitelaw, J. H., "Laminar Flow in a Square Duct of Strong Curvature," *Journal of Fluid Mechanics*, Vol. 83, Pt. 3, 1977, pp. 509–527.

[23] Rojas, J., Whitelaw, J. H., and Yianneskis, M., "Developing Flow in S-Shaped Diffuser," Dept. of Mechanical Engineering, Rept. FS/83/12, Imperial College of Science and Technology, London, 1983.

[24] Majumdar, S., Rodi, W., and Zhu, J., "Three-Dimensional Finite-Volume Method for Incompressible Flows with Complex Boundaries," *Journal of Fluids Engineering*, Vol. 114, No. 4, 1992, pp. 496–503.

[25] Kim, S. W., and VanOverbeke, T., "On the Anomaly of Velocity-Pressure Decoupling in Collocated Mesh Solutions," NASA TM 103769, March 1991.

[26] Ferziger, J. H., and Peric, M., "Finite Difference Methods," *Computational Methods for Fluid Dynamics*, Springer, New York, 1999, pp. 59–60.

G. M. Faeth
Editor-in-Chief