NASA-TM-107135
19960012165

NASA Technical Memorandum 107135
AIAA–96–0762

A Mixed Volume Grid Approach for the Euler and Navier-Stokes Equations

William J. Coirier and Philip C.E. Jorgenson
Lewis Research Center
Cleveland, Ohio

Prepared for the
34th Aerospace Sciences Meeting and Exhibit
sponsored by the American Institute of Aeronautics and Astronautics
Reno, Nevada, January 15–18, 1996

National Aeronautics and
Space Administration

A MIXED VOLUME GRID APPROACH FOR THE EULER AND NAVIER-STOKES EQUATIONS

William J. Coirier*
Philip C.E. Jorgenson**
Computational Fluid Dynamics Branch
Internal Fluid Mechanics Division
NASA Lewis Research Center, Cleveland Ohio

Abstract

An approach for solving the compressible Euler and Navier-Stokes equations upon meshes composed of nearly arbitrary polyhedra is described. Each polyhedron is constructed from an arbitrary number of triangular and quadrilateral face elements, allowing the unified treatment of tetrahedral, prismatic, pyramidal and hexahedral cells, as well the general cut cells produced by Cartesian mesh approaches. The basics behind the numerical approach and the resulting data structures are described. The accuracy of the mixed volume grid approach is assessed by performing a grid refinement study upon a series of hexahedral, tetrahedral, prismatic and Cartesian meshes for an analytic inviscid problem. A series of laminar validation cases are made, comparing the results upon differing grid topologies to each other, to theory and experimental data. A computation upon a prismatic/tetrahedral mesh is made simulating the laminar flow over a wall/cylinder combination.

I Introduction

Unstructured grids are rapidly becoming more useful for the simulation of inviscid flows in complex geometries. The promise of easing the burden of grid generation for complex geometries is being met. By exploiting certain geometric properties of tetrahedra and convex unit aspect ratio hexahedra (Cartesian cells), efficient methods can be found that fill the volume of the domain, with some user intervention still needed to provide guidance upon cell size and possibly stretching directions. Although the volume grid generation can be relatively automated, the surface discretization of complex geometries is still a non-trivial task. There are presently two separate camps of unstructured volume grid generation: tetrahedral and Cartesian based. Tetrahedral based mesh generation approaches currently being investigated can be grouped into advancing-front [1], advancing-layer [2], and point insertion [3] methods. Cartesian mesh generation is a relatively newer approach, which uses a recursive subdivision of convex, unit-aspect ratio Cartesian cells, and creates (possibly) non-convex polyhedra near boundaries [4, 5, 6].

The use of tetrahedral elements can provide efficient cell-centered and vertex-based schemes. For a cell-centered approach, where the conservation volumes are the tetrahedra themselves, the fixed number of faces and vertices of the control volume results in a simpler flow solver. For vertex-based schemes, where the control volume is the dual mesh, elegant formulations can result using a conservative, finite-element framework. By using linear finite-elements and exploiting certain geometric properties of the tetrahedra, efficient edge-based schemes can be formulated. The use of tetrahedral grids does, though, have its drawbacks. One stems from requiring that the surface discretization match faces in the volume grid exactly, which makes the surface discretization a controlling part of the quality of the volume grid generation. In addition, the volume grids generated are irregular in the sense that the orientation of the faces of the volumes do not typically follow a preferred direction.

Cartesian based approaches attempt to overcome these two problems by filling the volume with regularly oriented, nearly isotropic cells, that become general polyhedra near the boundaries, where these boundary cells have been cut from the Cartesian/boundary intersections. This has essentially sacrificed grid smoothness at the boundary for grid smoothness over the larger portion of the volume. Other benefits of the Cartesian approach can be traced to taking advantage of the geometric regularity of the un-cut cells, and other implementation specific benefits resulting from the hierarchy of the grid from the grid generation process. The lack of grid smoothness along the boundaries can cause problems for both inviscid and viscous calculations, and the resulting solvers are slightly more complicated than those based upon tetrahedral cells. These drawbacks aside, the Cartesian approach is proving to be a very successful method for computing inviscid flows about complex geometries.

*Aerospace Engineer, coirier@lerc.nasa.gov

** Aerospace Engineer, Member AIAA

Both tetrahedral and Cartesian strategies are lacking when computing viscous flows. The current viscous flux formulations dictate that smoothly stretched, nearly orthogonal grids are needed to provide robust and accurate predictions of viscous flows. The requirement of grid smoothness rises to extreme importance, since non-smoothness has a direct effect upon needed derivative quantities at walls, such as skin friction and heat transfer, which are typically the quantities desired from such an analysis. Grid smoothness also has a direct effect upon convergence behavior, since the typical flux functions in use today will produce non-positive stencils if certain geometric qualities of the mesh are not met [3,4]. To predict skin friction and heat transfer properly in turbulent flows, high resolution is needed normal to the wall dictating large numbers of cells. In addition, from an efficiency standpoint, grid stretching is typically needed in only a single direction, normal to the stream surface, and is not needed along it. By construction, Cartesian based methods do not allow for anisotropy of the mesh, while the efficiency of using highly stretched tetrahedral cells is suspect.

A means that is proposed to alleviate these deficiencies is currently being called a prismatic grid approach. In this case, bounding surfaces are triangulated, and this bounding triangulation is extruded away from the surface, creating layers of cells that are smoothly stretched in a surface normal direction. Within the layers, the desired smoothness and near-orthogonality is retained. These prismatic cells are typically grown out a distance from the surface, then a volume mesh generation strategy is used to fill the void. Examples of this approach are shown by Melton et al. [7] for the Euler equations, where a Cartesian grid was used to fill the void, and a hyperbolic-like approach was used to generate the prismatic layers. Karman [6] used a similar Cartesian/prismatic approach for the Euler and Navier-Stokes equations, where a more algebraic approach was used for the prisms. Connell et al. [8,9] used a surface discretization coupled with a CAD-based surface description, from which an algebraic approach extruded the prisms, and an advancing front mesh generator filled the void with tetrahedra. Kallinderis et al. [10] used a similar approach, but did not create the surface description from a CAD basis. By exploiting the semi-structured nature of the prismatic portions of the grid, Parthasarathy et al. [11] have proposed an efficient strategy to solve the Euler and Navier-Stokes equations. Some obvious drawbacks of the prismatic approach, in general, still require some work to resolve. For instance, the boundary surface discretization will control the smoothness of the grid near the wall, and care must be taken to ensure smoothness at the prism/volume grid interface. Regardless, current examples of this approach show tremendous potential, where it is hoped to alleviate many of the problems unstructured grid approaches encounter for computing high Reynolds number, turbulent flows.

A common thread to computing flows upon these classes of grids is that the flow solver must handle both tetrahedral, pentahedral (prismatic and pyramid) and hexahedral cells. Additional capability to handle adaptive mesh refinement, "hanging nodes", Cartesian generated grids with their cut cells, the extrusion of quadrilateral cells into hexahedra, or, perhaps, extrusion of other surface polygons would also be desired. In general, this type of solver must be able to solve the conservation laws upon general, non-simplicial conservation volumes.

The use of edge-based data structures have been proposed to solve the Euler/Navier-Stokes equations on mixed-element meshes by Mavriplis et al. [12]. In this case, a convincing argument is made for the use of mixed-element meshes, and computations using differing element types for the same meshes are performed, rather impressively. In [12] the edge-based formulation is also used for the discretization of the viscous terms, which analysis shows to be inaccurate on non-simplicial meshes. An argument is made that relates this discretization to a thin-layer like formulation, so for certain flows, the results might be adequate, but in general, a different formulation for the viscous terms might be desirable. This will undoubtedly not be solely edge-based, but a careful implementation should not detract too much from the approach.

The approach presented here solves the Euler and Navier-Stokes equations using a cell-centered, finite-volume scheme upon control volumes of nearly arbitrary polyhedra constructed from triangular and quadrilateral faces. The four basic cell types of tetrahedra, prisms, pyramids and hexahedra are a subset of this, plus the approach can compute flows upon Cartesian generated grids, and grids where cell refinement has introduced "hanging nodes". It is certain that by restricting the mesh to be comprised of simpler polyhedra, a simpler flow solver results. The approach here is based upon the premise that by placing less restriction upon the topology of the mesh, an overall faster turnaround time will result. The additional computational complexities associated with this approach are tractable with well thought out data structures and algorithms. One noted difference from this approach and standard cell-centered methods is that both cell-averaged data and data at the vertices of the control volumes are used.

The outline of this paper is as follows. The basic data structures used for the approach are explained, then the approaches used to solve the conservation laws are shown. The issues regarding vectorization are addressed, namely the coloring of the different operations upon the basic data types. The accuracy of the approach is assessed using an analytic solution to the Euler equations for the four basic cell types mentioned above, and Cartesian generated grids. The laminar flow over a selection of problems are then made, and comparison is made to theory (flat plate) and experiment (developing duct flow). To demonstrate the capability of computing upon prismatic/tetrahedral meshes the approach is used to compute the laminar flow about a right circular cylinder upon a flat plate using a grid generated by Connell et al. [8].

II Data and Data Structures

The compressible Euler and Navier-Stokes equations are solved in three-dimensions in a cell-centered, finite-volume format upon a mesh of polyhedra where each polyhedron is created from an arbitrary number of triangular and quadrilateral face elements. This particular finite-volume approach uses data at both cell centers and mesh vertices, which implicitly uses all nearest neighbor cell data, without having to store its connectivity. Since the governing equations are being solved in conservation law form using a cell-centered scheme, the data structures used in the code are designed for such an approach. Three separate data entities are identified that make up the mesh and are needed to perform the calculations: vertices, faces, and cells. These form a hierarchical-like relationship with each other, since faces are constructed from vertices and cells are constructed from faces. Edges can be obtained from faces/cells, but since they are not used directly in this cell-centered scheme, a separate data entity for them is not maintained. The separate operations needed in the solution of the conservation laws are cast as operations upon these data types. A collection of vertices/faces/cells which make up a portion (either complete or by some geometric decomposition) of the entire domain are further grouped together into data entities called grids. Due to the hierarchy of the data entities, much information can be obtained directly, such as cell-face-cell connectivity. To clarify these data entities, the following briefly describes each entity, the data it contains, and how it is stored and maintained.

II.a Grids

Grids are composed of lists of cells/faces/vertices upon which the computations are performed, as well as lists of boundary faces and boundary vertices. All loop coloring information is also stored here. It is intended to eventually perform multigrid acceleration, where each grid in the sequence will be constructed via an agglomeration approach. This approach lends well to a hierarchy of grids, corresponding to the agglomeration of the parent grid and the construction of further grids in the sequence via solution adaptive mesh refinement. This will be represented hierarchically in the grid data structure also, so that each grid will have pointers to its parent and children. Although multigrid acceleration is not employed at this stage of the solver development, the use of the grid data entity should aid in its implementation. The grid data type might also be useful for parallel computations, by holding the spatially decomposed data.

II.b Cells

Cells define the conservation volumes upon which the conservation laws are solved and provide a place to store cell-averaged data, flux balances and other cell-based quantities. Each cell is comprised of an arbitrary number of faces, where a list of pointers for each face is maintained in the cell. Cells are accessed by a list of pointers to the cell data structures. For vectorization of cell-to-vertex scatter operations, cells are grouped into like vertex number groups and colored (see Section IV).

II.c Faces

Fluxes are integrated across faces, and the result of the integrations are scattered to the cells which are logically left and right of the face where the logical orientation is determined by the face normal vector. Faces contain pointers to the cells that are logically left and right of the faces. Each face contains a list of pointers to globally unique vertices defining the geometry of the face. For practicality, the faces must be either triangular or quadrilateral. The needed geometric data for the flux integration (Gauss points, area vector) are stored in the face data structure, but may also be computed from the face vertex data. Cell faces are constructed from the positively (outward pointing normal) ordered vertex lists of the input cell definition. To create a global list of unique faces, an octree-based, fixed bucket size searching procedure is used to match already created faces. Faces are sorted according to Gauss point location in space, and are matched by their ordered vertex lists. The tree automatically sizes itself during the creation phase, and prunes itself to zero length when it is not needed. For a multi-block grid or a grid with different grid types in each domain, there is no need to supply inter-block or inter-grid connectivity data, since this face matching procedure will automatically ensure the proper face matching across inter-block or cell-refinement boundaries. Interior and boundary faces are maintained and processed in separate lists. The face data is not only used for flux evaluation, but are also used in the upwind, inviscid reconstruction procedure explained in Section III.b. For vectorization purposes, the face loops are also colored (Section IV). For boundary condition application, ghost cells are created for all boundary faces, and the data used in the cell-to-vertex scatters. Boundary condition faces of the same boundary condition type are grouped together, and these are also colored.

II.d Vertices

The vertex data structures hold the spatial coordinates of the vertices of the mesh, and provides a list of pointers to cells which have edges of faces that are subtended by the vertex. This provides a means of obtaining the solution at the vertices from the cell-centered data, which is needed to compute the upwind, inviscid reconstruction, to form the viscous fluxes, and to plot the solution. Vertex data is obtained in a nominally linearity-preserving manner, as shown in Section III.a. As in the face data, upon input, a self-expanding, bucket searching octree procedure is used to provide unique vertex data, where sorting and matching is made according to the spatial location of the vertex. This makes multi-block and multi-grid type data definition easier, since the burden of vertex matching is taken by the octree, and not the grid generation.

II.e Input Data Types

The flexibility of the conservation volume construction is evident by the various standard grid data types that can be read in by the code. The solver is presently configured to read in 5 types of grid data: PLOT3D data [13], VGRID data (see [2] and others), an input format corresponding to the prismatic/tetrahedral grid generation system described in [8,9,14], and a format corresponding to the tetrahedral generator of the FELISA system[15]. Another grid type is also available, termed here as the MVG (Mixed Volume Grid) type, which defines each cell as being constructed of an arbitrary number of triangular and quadrilateral faces. Cartesian generated grids are input as the MVG data type, since the cut cells generated by the Cartesian grid generator produces polyhedra that are not of the four types listed above. All of the 5 specific data types can be translated into the MVG format.

III Solution of the Conservation Laws

The Euler and Navier-Stokes equations cast in conservation law form are solved in a cell-centered, finite-volume format upon the polyhedral control volumes. Both upwind and central-difference approximations of the convective fluxes are used, and a directed gradients procedure is used for the viscous fluxes. A simple three-stage explicit scheme with a spatially varying time step, based upon both hyperbolic and parabolic time step constraints, is used to update the solution. Contrary to most cell-centered approaches, the solution procedure here relies upon both vertex and cell-averaged data. The use of both vertex and cell-averaged data is also used in the USM3D series of unstructured mesh solvers developed by Frink [16], and has inspired some of the generalizations to mixed-volume grids, shown here.

III.a Vertex Data Interpolation

The data at the vertices of the mesh is obtained from the cell-averaged data by a linearity-preserving interpolation procedure. This interpolation procedure is similar to that presented by Holmes and Connell in [17], where it is termed a linearity-preserving Laplacian weighting. By considering an interpolation formula that interpolates the solution at the j-th vertex from the n cells surrounding it as

$$f_j = \sum_n \varpi_n f_n \tag{1}$$

where the ϖ_n are found from the weights of a pseudo-Laplacian operator,

$$L(f) = \sum_n \omega_n (f_n - f_j) \tag{2}$$

as

$$\varpi_n = \frac{\omega_n}{\sum_n \omega_n} . \tag{3}$$

Linearity preservation is guaranteed by requiring (2) satisfy

$$L(x) = L(y) = L(z) = 0 . \tag{4}$$

By expanding the weights about unity in terms of linear basis functions, as

$$\omega_n = 1 + \lambda_x (\bar{x}_n - x_j) + \lambda_y (\bar{y}_n - y_j) + \lambda_z (\bar{z}_n - z_j) \tag{5}$$

a 3x3 linear system is found for the λ_i which is inverted beforehand. This process is purely geometric, and therefore is precomputed for a given mesh, and only requires a simple cell-to-vertex scattering procedure. By providing higher-order constraints in (4) and expanding with higher-order basis functions in (5), quadratic-preserving reconstructions can also be found [4] but are not used here, since only linearity-preserving cell- and face-based reconstructions are used. In practice, the weights (5) are restricted $0 < \omega_n < 2$.

III.b Upwind Flux Construction

The upwind scheme follows standard practice used for unstructured grids: A reconstruction procedure is used to reconstruct the solution locally within each cell, which is then followed by an upwind flux construction at the cell faces. The reconstruction within each cell is used to provide states to an approximate Riemann solver, which is used to compute the fluxes. The fluxes are then scattered to the logical left and right cells of the face.

IIIb.1 Reconstruction

The reconstruction of the solution within each conservation volume is found by performing a surface integral over the conservation volume itself. Since the data at the vertices of the conservation volume are found in a linearity preserving manner, a second-order, Gaussian quadrature guarantees that the reconstructed solution is also linearity preserving. For a general control volume, consider finding the gradient in the reconstruction

$$u = \bar{u} + \frac{\partial u}{\partial x_i} \cdot (x_i - \bar{x}_i) \tag{6}$$

by applying a divergence integral over the volume

$$\int_V \frac{\partial u}{\partial x_i} dV = \oint_{\partial V} u \hat{N}_i d(\partial V) \tag{7}$$

By replacing the surface integral in (7) with a numerical quadrature, the gradient can be found by a face based scatter operation. The surface integral is replaced by a single point quadrature so that the gradient is found to be

$$\frac{\partial u}{\partial x_i} = \frac{1}{V} \sum_{faces} u_G N_i \tag{8}$$

where N_i is the i-th Cartesian component of the outward facing (non-unit) normal, and u_G is found by evaluating a linear or bi-linear expansion in the face at the face Gauss point for three- and four-vertexed faces, respectively. The cell volumes and centroids are found beforehand by using a third-order Gaussian quadrature of another application of the divergence integral. For the general flux G_i, application of the divergence theorem replaces the volume quadrature with a surface integral. That is

$$\int_V \frac{\partial G_i}{\partial x_i} dV = \oint_{\partial V} G_i \hat{N}_i d(\partial V) \tag{9}$$

To obtain the cell volume and centroid locations the flux on the right hand side of (9) is taken to be

$$G = \begin{bmatrix} (x, 0, 0) & \text{for V} \\ \left(\frac{x^2}{2}, 0, 0\right) & \text{for } \bar{x}V \\ (xy, 0, 0) & \text{for } \bar{y}V \\ (xz, 0, 0) & \text{for } \bar{z}V \end{bmatrix} \tag{10}$$

The reconstruction (8) preserves linear data, and for a general mesh implicitly includes all local cell data in the construction of the gradient through the vertex data interpolation. Since the faces are colored for vectorization of the integration of the inviscid and viscous fluxes, this procedure follows closely the overall face-based solution method.

It is instructive to note that when applied to a mesh of tetrahedra, this procedure results in the same formula found by Frink [16], also pointed out by Mitchell [18].

IIIb.2 Upwind Flux Construction

Three popular upwind flux functions are available: Roe's FDS [19], Van Leer's FVS [20] and Liou's AUSM+ [21]. The reconstruction provides the left and right input states at the faces, which then use the approximate Riemann solvers noted above to compute the flux. The upwind fluxes are then scattered to the cells.

III.c Central Differencing with Explicit, Scalar Dissipation

A conservative formulation corresponding to centrally differenced fluxes with blended second- and fourth-order dissipation is also available in the solver. Following [22], at a given face, the flux is formulated as

$$F = \frac{1}{2}(F_L + F_R) - d \tag{11}$$

where the dissipative flux, d, is constructed from a blending of a first-difference and pseudo third-difference. When integrated over the faces of the control volume, this yields a blended Laplacian and Biharmonic operator which is used to dissipate spurious oscillations. This scalar dissipative flux is formulated as

$$\begin{aligned} d &= d_1 - d_3 \\ d_1 &= \tilde{\sigma}\varepsilon^{(2)} (W_R - W_L) \\ d_3 &= \tilde{\sigma}\varepsilon^{(4)} [D_2(W_R) - D_2(W_L)] \end{aligned} \tag{12}$$

The pseudo-second differences formed for each cell,

D_2, represent undivided approximations to Laplacians using only differences about faces. That is, for an arbitrary function, f

$$D_2(f) = \sum_{faces} (f_R - f_L) \qquad (13)$$

where the R and L refer to the logically oriented cells sharing the face.

The coefficients $\varepsilon^{(2)}$ and $\varepsilon^{(4)}$ are used to scale the first and pseudo-third differences with the cell size, and to turn off the fourth-order dissipation across shocks. The coefficient of the first-difference in (12) is computed as a normalized second-difference of pressure, acting as a shock sensor

$$v = \frac{D_2(P)}{\sum_{faces} (P_R + P_L)} \qquad (14)$$

This gives the coefficients

$$\begin{aligned} \varepsilon^{(2)} &= \kappa_2 \max(v_R, v_L) \\ \varepsilon^{(4)} &= \max\left[0, \left(\kappa_4 - \varepsilon^{(2)}\right)\right] \end{aligned} \qquad (15)$$

Standard values of the dissipation coefficients are used as $\kappa_2 = 1/2$ and $\kappa_4 = 1/32$. The factor $\bar{\sigma}$ scales the dissipation according to the maximum eigenvalue of the flux jacobian as

$$\begin{aligned} \bar{\sigma} &= \frac{(\sigma_L + \sigma_R)}{2} \\ \sigma_{L/R} &= \left(\left|u\hat{n}_x + v\hat{n}_y + w\hat{n}_z\right| + a\right)\Big|_{L/R} \end{aligned} \qquad (16)$$

On a uniform mesh, this dissipation results in a blended Laplacian/biharmonic operator.

III.d Treatment of Viscous Fluxes

The viscous fluxes are computed following a directed gradients procedure suggested by Mitchell [23]. This procedure is linearly K-exact, and produces the same gradient computed using a divergence-based reconstruction that preserves linear data, as in [24] and [4], commonly called a diamond-path reconstruction. The idea is based upon taking the inner product of the gradient with three vectors joining locations where the data has been obtained in an at least linearity-preserving manner. That is, for the three vectors $\delta_1, \delta_2, \delta_3$

$$\begin{aligned} \Delta u_1 &= (\nabla u) \cdot \delta_1 \\ \Delta u_2 &= (\nabla u) \cdot \delta_2 \\ \Delta u_3 &= (\nabla u) \cdot \delta_3 \end{aligned} \qquad (17)$$

This results in a 3x3 linear system for the unknown gradient, as

$$\begin{bmatrix} \Delta x_1 & \Delta y_1 & \Delta z_1 \\ \Delta x_2 & \Delta y_2 & \Delta z_2 \\ \Delta x_3 & \Delta y_3 & \Delta z_3 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ \Delta u_3 \end{bmatrix} \qquad (18)$$

The choice of the base vectors, δ_i is taken so that δ_1 joins the two centroids of the cells that share the face, and the two others lie in the plane of the face. When a linearly-exact procedure is used to produce the data at the vertices, this procedure preserves linear gradients. The viscous flux construction, evaluation and scattering to the cell residuals is done on a face basis, which for vectorization, depends upon loop coloring, as is explained in the next section.

IV Loop Coloring for Vectorization

Vectorization of the solver is pursued for the use of CRAY class vector machines. If the solver is used upon non-vector machines, such as workstations, or is made parallel, the basic structure and loop indexing used for parallelization will result in a negligible penalty. The preprocessing time for the loop colorings is marginal.

There are primarily two different types of scatter operations that must use loop coloring to obtain vectorization. Loops over faces scattering to cells are performed for the inviscid reconstruction, time-step computation and for the viscous and inviscid flux integrals. Loops over cells, scattering to vertices are performed to interpolate the data at the vertices from the cells. All loops which are to be vectorized are done so by compiler directives to ignore vector dependencies. The vectorized code ports to non-vector machines with no changes.

The face loop coloring is performed in a heuristic fashion as indicated by the following pseudo-code.

```
N_colors=0
/* loop over all faces */
for(all faces)
    added_color=false
    /* loop over all available colors */
    for(all colors && !added_color)
        L_clrd=is_left_colored_color(color)
        R_clrd=is_right_colored_color(color)
        /* if left and right cell are not color
```

```
            color them color */
         if(!L_clrd && !R_clrd)
            face_color=color
            added_color=true
         endif
      endfor
      /* if not colored, color it color */
      if(!added_color)
         face_color=N_colors
         N_colors+=1
      endif
   endfor
```

This results in roughly even-length loops of faces, except for the last one or two colors, which typically have a smaller numbers of faces. The balance of the colors is dependent upon the order that the above algorithm visits the faces, but using the input ordering appears to be sufficient.

Vectorization of the cell-to-vertex scattering operation is complicated by the fact that the cells in the mesh may have a variable number of vertices. This is overcome by grouping cells into groups of like vertex number, termed here as verticity. Within cells of the same verticity, cells are colored to avoid scattering to the same vertex in the same loop. A binary-encoded coloring history of each vertex is used to group the cells into colors. The following pseudo-code illustrates the approach.

```
/* zero history for all vertices */
for(all vertices)
   hist=0
endfor

/* loop over like-verticity cells */
nColors=0
for(all cells)
   /* construct a single history for this
   cells' vertices using bitwise or */
   v_C=0
   for(all vertices in cell)
      v_C=v_C | hist
   endfor
   f_a_c=first_avail_color(v_C)

   num=1
   for(all colors < f_a_c)
      num*=2
   endfor
   for(all vertices in cell)
      hist=hist | num
   endfor
endfor
```

The routine `first_avail_color` performs bitwise shifts upon the cell-encoded vertex history and finds the first available color for the vertices in the cell.

```
int first_aval_color(vC_hist)
   pos = 1 << 0
   color=0
   while( (vC_hist & pos) != 0)
      pos=pos << 1
      color++
   endwhile
return color
```

This colors the loops within a verticity-group of roughly even length loops, with the last one or two loops of smaller length. Representative performance values of the vectorized code are given in the following sections.

V Validation and Demonstration

The accuracy of the solution procedure is first assessed by performing a grid refinement study upon a sequence of related meshes of hexahedra, prisms, tetrahedra and those produced by a Cartesian grid generator. A developing laminar boundary layer and a developing laminar duct flow are shown. Also shown is the laminar flow computed about a flat plate/cylinder intersection, upon a prismatic/tetrahedral grid generated by Connell [8].

V.a Grid Refinement Study: Supersonic Vortex

An analytical solution to the Euler equations is used to assess the accuracy of the convective flux discretization schemes of the solver. This flow field has been used by Aftosmis et al. [25] to investigate the effects of gradient limiting and by Luo, et al. [26] to assess an improved reconstruction scheme for triangular meshes. The solution is a relatively simple function, and also lends itself well to a grid refinement study, since although the flow is simple, there are considerable gradients in pressure and density across the domain. By assuming an isentropic flow described by a line vortex situated at the origin, parallel to the z-axis, the solution is

$$\begin{aligned}\frac{T}{T_i} &= 1+\frac{(\gamma-1)}{2}M_i^2\left(1-\frac{1}{\xi^2}\right)\\ \frac{\rho}{\rho_i} &= \left(\frac{T}{T_i}\right)^{1/(\gamma-1)}\\ \frac{P}{P_i} &= \left(\frac{T}{T_i}\right)^{\gamma/(\gamma-1)}\\ u &= U\cos\theta\\ v &= U\sin\theta\\ w &= 0\end{aligned} \tag{19}$$

where the i-subscript refers to conditions along a reference (inner) radius r_i, $\xi = r/r_i$ the polar angle in the z=constant plane is $\theta = \operatorname{atan}(y/x)$, and the flow speed non-dimensionalized by the inner radius speed is

$U = 1/\xi$. The domain is taken to be for $\xi \in [1, 1.384]$, $z \in [0, 1]$ and $\theta \in [0, \pi/2]$, and the Mach number along the inner radius is taken to be $M_i = 2.25$. This is a two-dimensional problem, so during the refinement sequence, the discretization of the grid in the z-direction is kept the same for all of the grids. Figure 1 shows Mach number contours for this flow.

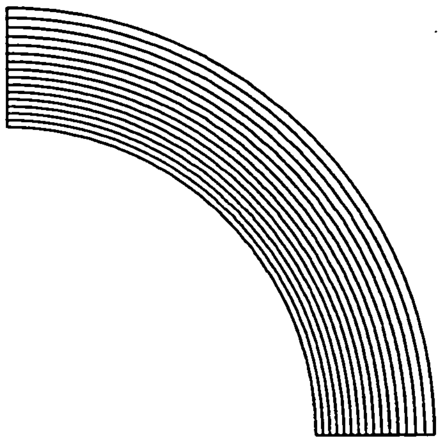

Figure 1. Mach contours, compressible vortex flow.

A sequence of related grids are generated based upon a sequence of structured hexahedral grids with 5x5x5, 10x10x5, 20x20x5 and 40x40x5 cells. Tetrahedral grids are created from the hexahedral meshes by creating six tetrahedra from each hexahedra. Three families of prismatic grids are generated from the base hexahedral meshes by creating prisms from the hexahedra whose orientation of the normal vector of the triangular faces in the prisms lie along the three different computation coordinate axes. A sequence of Cartesian grids are also generated, which are also used to assess the accuracy, but are not directly related to the five grid sequences above. Figure 2 shows the relationships between the hex-related meshes, while Figure 3 shows the intersection of a z=constant plane through a representative Cartesian mesh. Note that for plotting purposes, cut-Cartesian cells that cannot be represented as either hexahedra, pyramids, prisms or tetrahedra are split into a collection of tetrahedra and pyramids. For these cut cells, a fictitious point located at the cell centroid is introduced, which is used to create, corresponding to each triangular or quadrilateral face of the cell, a tetrahedron or a pyramid, respectively. This procedure is only needed for plotting purposes, and is not used in the flow solver.

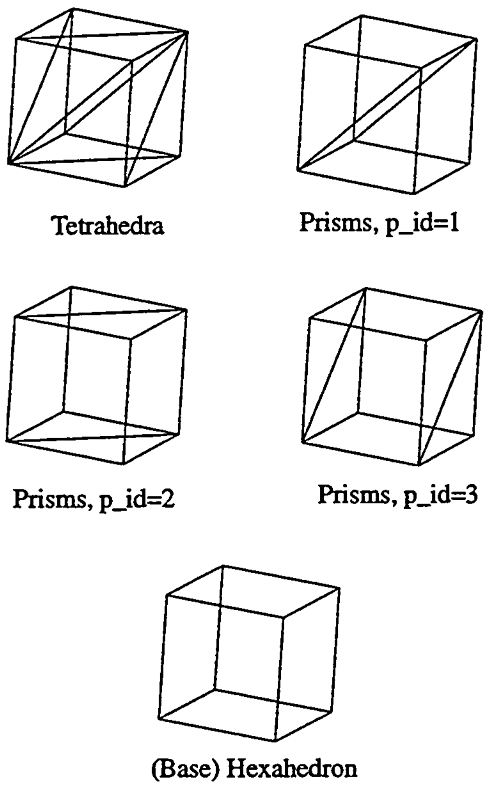

Figure 2. The five, hexahedra-derived grids types used in the mesh convergence study.

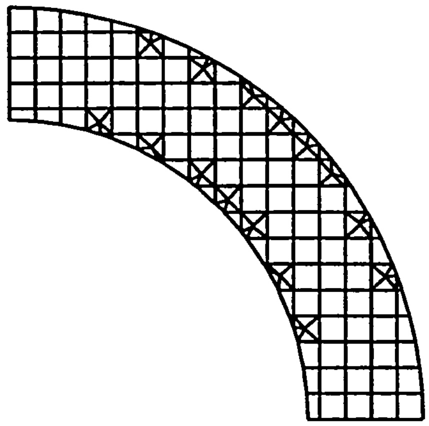

Figure 3. z=constant cut through a Cartesian mesh

A plot of the L_1 norm of the density error $e_\rho = |\bar{\rho} - \rho_{exact}|$ against a representative two-dimensional length scale found on each mesh is shown in Figure 4. This length scale is introduced merely for the estimation of the truncation error order, and is not representative of a computational cost or efficiency norm. The intent of this study is only to gauge the accuracy and correctness of the inviscid flux constructions and is not an attempt to assess whether one mesh topology is superior to another. That assessment would require a careful comparison of cost as well. This representative length scale is calculated as

$$l_{2d} = \sqrt{\frac{V_{ave}}{\Delta z}} \qquad V_{ave} = \frac{\sum_{nCells} V_n}{nCells} \tag{20}$$

All calculations were performed using the upwind formulation III.b, with the FDS scheme without gradient limiting.

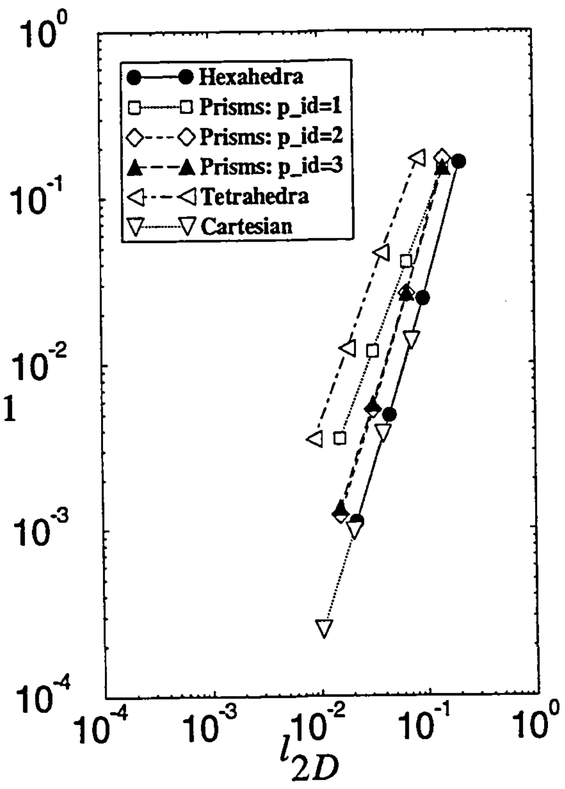

Figure 4. L_1 -norms for upwind scheme

The orders of the discrete truncation error, found by computing the slope on a logarithmic plot of the final two meshes in each sequence, for the six different mesh sequences, are shown in Table I, in the Appendix. All schemes were globally second-order accurate, while the tetrahedral and plane_id=1 meshes had a first-order max-norm.

When viewing Figure 4, it is important to keep in mind that the hexahedral, tetrahedral and prismatic meshes are all interrelated, since they are derived from the same mesh, while the Cartesian is not. All meshes are made with a constant spacing in the z-direction, resulting in a stack of four cells in the z-direction. The Cartesian mesh and the hexahedral mesh are shown to have nearly the same truncation error for the same length scale, which is attributable to the similarity in the reconstruction/flux construction schemes that they use. In [27] an analytic solution of the Euler equations, Ringleb's flow, was used to compare the error computed by the Cartesian approach and a structured mesh approach. There, a structured mesh, which used an upwind coordinate-by-coordinate reconstruction, was shown to be slightly more accurate than the Cartesian approach, which used a multi-dimensional reconstruction. The results shown here compare the truncation error between a sequence of hexahedral and Cartesian meshes, where the multi-dimensional reconstruction shown in Section IIIb.1 is used.

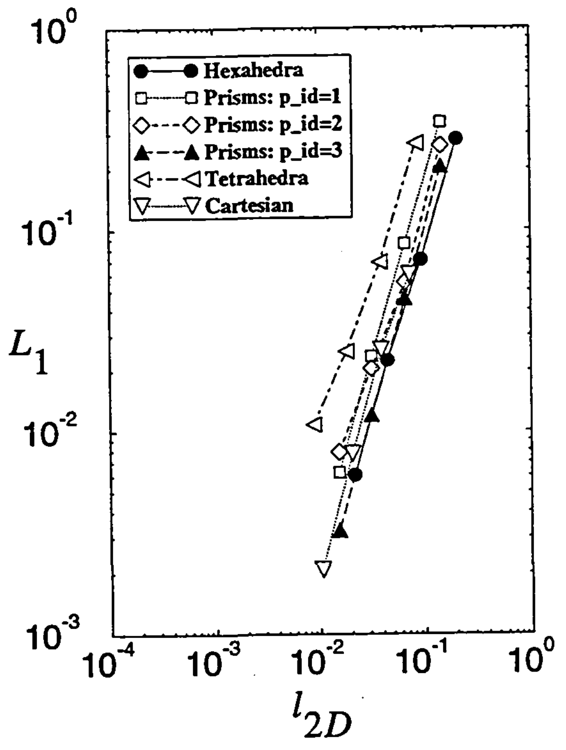

Figure 5. L_1 -norms for central scheme.

A series of calculations were also made using the central-difference flux scheme with added dissipation on the exact same meshes. Figure 5 shows the computed L_1 norms against the two-dimensional length scale and Table II shows the asymptotic orders of accuracy.

As is indicated by the order of the L_∞ norms, the particular implementation of the dissipative flux constructions is not uniformly second-order accurate. It is surmised that the construction of the dissipative fluxes near the boundaries has reduced the accuracy of the scheme. The results shown here construct the third-difference operator at the boundaries using ghost-cell information. A comparison to a presumably less accurate formulation which does not use ghost cell data indicated no appreciable improvement.

Since the discrete errors for both the upwind and central difference schemes are available for the same meshes, one can also compare these two schemes. Figure 6 shows the L_1 norms for the hexahedral, tetrahedral and Cartesian meshes.

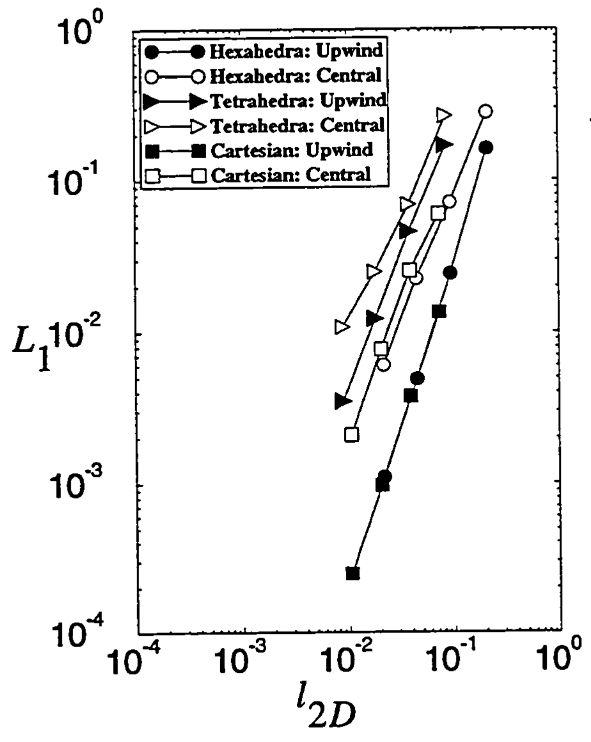

Figure 6. Comparison of L_1-norms for the upwind and central schemes.

As is seen in the plot, there is an appreciable difference between the truncation error of the central and upwind schemes on the Cartesian mesh, where the central scheme exhibits nearly ten times the error on the finer mesh than the upwind scheme. This difference in error is not at as severe on the tetrahedral and hexahedral meshes, where the central scheme error is approximately four and six times more than the upwind formulation, respectively.

The grid convergence study was performed on IBM RS6000, model 590 workstations, using the xlc compiler with standard optimizations. The processing rates for both the upwind and central schemes are shown in Table III and Table IV for the final meshes in each sequence. Also shown in are the rates for the same meshes and algorithms, but on the CRAY C-90, eagle, using the Cray standard C compiler, Version 4.0.2.7.

It is important to consider the particular way the different schemes are compared to one another in Figure 4, Figure 5 and Figure 6. The length scale has been constructed to deduce the order of the schemes. A comparison that would critically gauge the different approaches upon the different mesh topologies must use some measure of computational efficiency, perhaps measured by a memory-time integral. This efficiency will be greatly effected by the tailoring of the solver to a particular convective flux discretization (i.e. central or upwind) and for a particular mesh topology. The intent of this convergence study is only to verify the accuracy of the particular flux constructions and to validate the mixed volume grid approach, and not to promote one mesh topology over another.

V.b Developing Laminar Flow over a Flat Plate

The flow developing over a flat plate which is aligned with the free stream is computed next. A mesh of hexahedra is generated with 31, 21 and 5 points in the streamwise, plate normal, and span-wise directions, respectively. The mesh is stretched in both the streamwise and normal directions. Flow conditions correspond to a freestream Mach number of $M_\infty = 0.25$ and a Reynolds number based on plate length of $Re = 50,000$. The code is run using the upwinding formulation and the FDS scheme. As in the previous case, a hexahedral mesh has been constructed, and from this mesh, other derived mesh types are found. A tetrahedral mesh and a prismatic mesh with the p_id= 2 are constructed from the base mesh, and the prismatic mesh is shown in Figure 7.

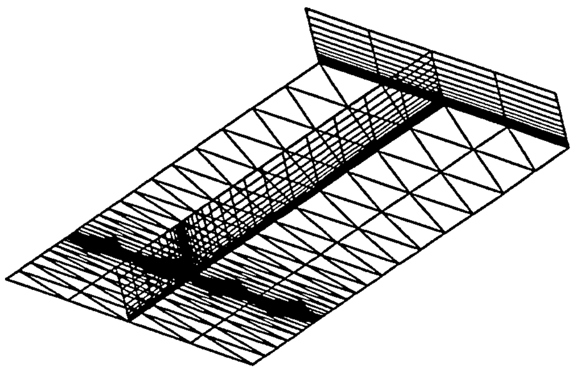

Figure 7. Prism (p_id=2) mesh derived from hexahedral mesh.

Figure 8 and Figure 9 show the computed u- and v-velocity profiles plotted against the similarity coordinate η, for the three meshes at a location on the mid-plane of the plate, with a Reynolds number based on distance from the leading edge of $Re_x = 40,000$. Each mesh produces good results.

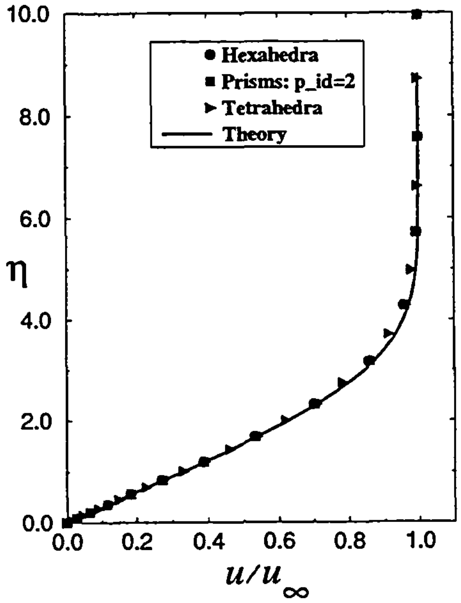

Figure 8. u-velocity profiles at $Re_x = 40,000$.

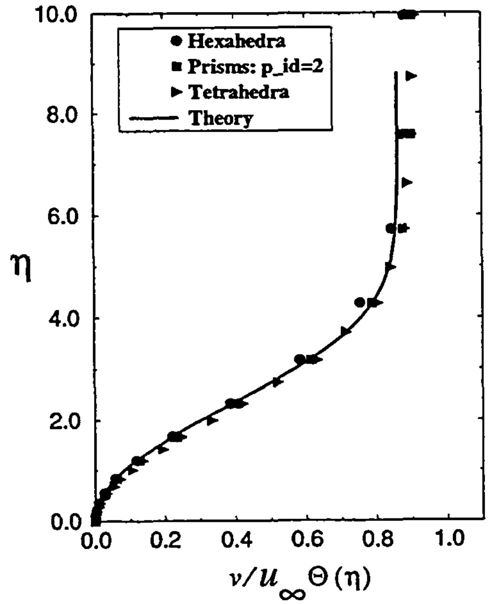

Figure 9. v-velocity profiles at $Re_x = 40,000$

V.c Developing Laminar Flow in a Rectangular Duct

The developing laminar flow in a five-to-one aspect ratio duct is computed next, and compared to the experimental data of Sparrow et al. [28]. The computations are performed for a Reynolds number based on inlet mass flow rate and hydraulic diameter of 500 for a duct length of 31.25 inches. The 5:1 aspect ratio duct has major and minor widths of 3.125 and 0.625 inches. The exit pressure is specified according to the experimental data from the experimentally measured pressure drop correlation

$$K_p = \frac{P_{t,\infty} - P(z)}{\frac{1}{2}\rho\bar{w}^2} = 1.89 + 76.3\left(\frac{z/D}{Re}\right) \qquad (21)$$

where D is the hydraulic diameter. The inlet total conditions are set so that the Mach number based on area-averaged axial velocity is $M_\infty = 0.1$. A 15x15x21 hexahedral mesh is constructed with stretching in all directions. A tetrahedral mesh is also constructed, as before, from the base hexahedral mesh. These calculations use the upwinding formulation with the FDS flux function. Figure 10 and Figure 11 compare the computed centerline pressure drop and streamwise velocities from the hexahedral and tetrahedral meshes to the experimental data of [28]. Agreement with experiment is considered good.

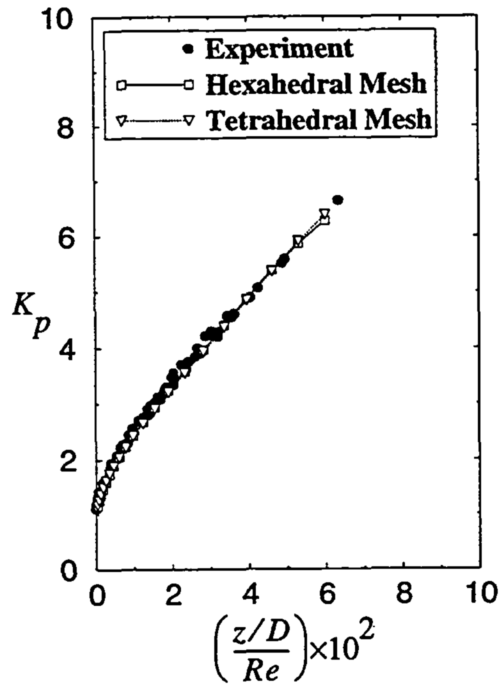

Figure 10. Computed and experimental axial-pressure loss.

\

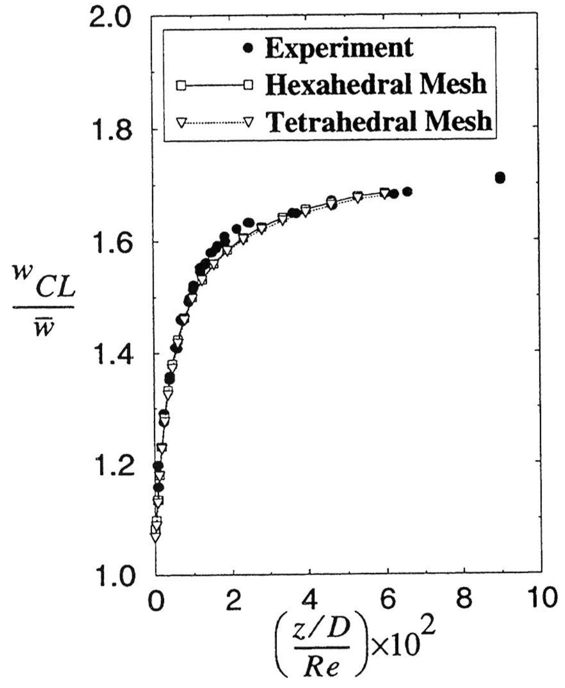

Figure 11. Computed and experimental centerline, axial velocity.

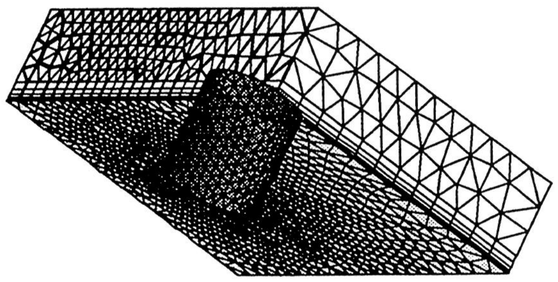

Figure 12. Perspective view of boundaries of prismatic/tetrahedral mesh for wall/cylinder.

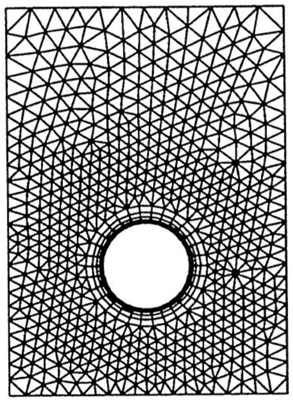

Figure 13. Upper boundary of prismatic/ tetrahedral mesh

V.d Wall/Cylinder Flow: Prismatic/ Tetrahedral Mesh

The flow about a wall/cylinder combination is computed using the mixed volume grid approach. The mesh was generated by the prismatic/tetrahedral mesh generator developed by Connell et al. [8], for which the solver provides an input data type. Flow conditions correspond to a Reynolds number based upon cylinder diameter of $Re = 50$ and a freestream Mach number of $M_{\infty} = 0.25$. Figure 12 shows a perspective view of the grid, while Figure 13 shows a slice through the mesh at the mid-plane, showing the prismatic mesh about the cylinder. The mesh has 9810 prisms and 19243 tetrahedra. The calculation converged approximately four orders of magnitude in 10,000 iterations, using 5011 seconds of cpu time and approximately 4.8 Mwords of storage on the Cray C-90, eagle. Figure 14 shows a perspective view of speed contours of the computed solution at various planes slicing the volume mesh, showing the boundary layer growth along the wall, the upstream influence of the cylinder and the momentum deficit in the wake. The predicted solution appears to be symmetric.

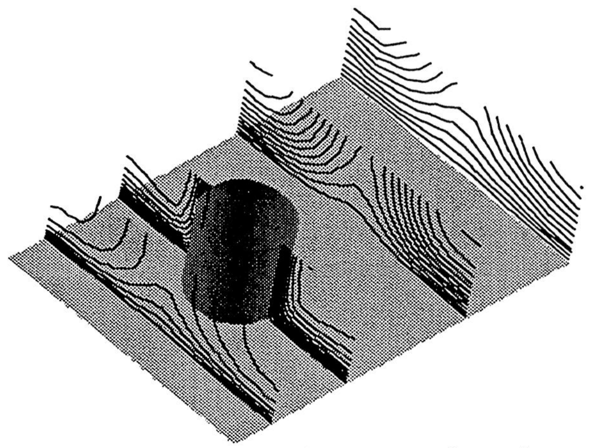

Figure 14. Speed contours through volume

VI Concluding Remarks and Future Efforts

An approach which solves the compressible Euler and Navier-Stokes equations upon control volumes of nearly arbitrary polyhedra has been presented. The conservation laws are solved using a cell-centered, finite-volume formulation where the control volumes are created from an arbitrary number of triangular and quadrilateral faces. This flexibility allows the unified treatment of structured, unstructured, prismatic, and Cartesian-cell based grids with a single flow solver. The basics behind the organization and interrogation of the data structures has been explained.

The mixed volume grid approach uses both cell-averaged and vertex data. The convective terms of the governing equations have been treated with both an upwind and central differencing approach. The reconstruction method used for the upwind flux discretization uses an application of the divergence theorem upon the actual control volume itself. The viscous fluxes have been constructed using a directed gradients procedure.

An analytical solution to the Euler equations, a supersonic vortex, has been used to assess the accuracy of the approach, and has been used to compare results amongst the different grid types and between the upwind and central difference approximations. A refinement sequence of four hexahedral meshes was used to generate 3 families of prismatic meshes and a sequence of tetrahedral meshes by subdividing each hexahedra into the respective volume type. In addition, a Cartesian generated mesh was used and compared to the others. Here, the bulk of the Cartesian generated volume is comprised of axes-aligned hexahedra with arbitrary polyhedra along the boundaries created from the geometric intersection of the boundaries and the Cartesian cells. The grid refinement study showed that the hexahedral based mesh had the lowest error, while the tetrahedral mesh had the highest. In addition, the global error norms indicated that for the same mesh the upwind formulation was more accurate than the central difference approach. The Cartesian grids were shown to give an error comparable to the hexahedral meshes. The upwind formulation was shown to be globally second-order accurate upon all the meshes, and locally second-order accurate on some of the element types. The central scheme was shown to be globally, marginally second-order accurate, and first-order accurate locally. This reduction in the order for the central scheme was surmised to be caused by the construction of the boundary dissipative fluxes.

Laminar calculations have been made for the developing flow over a flat plate for hexahedral, prismatic, and tetrahedral meshes and for the developing flow in a 5:1 aspect ratio duct. Favorable comparisons to theory and experiment were shown for these cases. To demonstrate a prismatic mesh type calculation, a prismatic/tetrahedral mesh, supplied by Connell et al. [8], was used to compute the flow about a cylinder/wall configuration.

Acknowledgments

The authors gratefully acknowledge Dr. Stuart Connell and Dr. Mark Braaten at the General Electric Corporate Research and Development center for supplying the mesh and input data for the wall/cylinder prismatic/tetrahedral mesh and Mr. R. Jeffrey Benko of The University of Michigan, Department of Aerospace Engineering for the Cartesian meshes used in the mesh convergence study. Special thanks go to Dr. N. T. Frink of NASA Langley Research Center for his helpful discussions, comments and supplied references. Trong Bui of NASA Lewis Research Center graciously supplied the experimental duct flow experimental data.

Appendix A: Mesh Convergence Study: Asymptotic Slopes

TABLE I Asymptotic Slopes of Error Norms: Upwind Formulation

Grid Type	L_1 Norm	L_∞ Norm
Hexahedral	2.08	1.94
Tetrahedral	1.77	0.93
Prismatic: plane_id=1	1.68	0.88
Prismatic: plane_id=2	2.05	1.63
Prismatic: plane_id=3	2.02	1.78
Cartesian: $\Delta z = 1/4$	2.03	1.73

TABLE II Asymptotic Slopes of Error Norms: Central Differencing Formulation

Grid Type	L_1 Norm	L_∞ Norm
Hexahedral	1.82	1.18
Tetrahedral	1.17	0.78
Prismatic: plane_id=1	1.85	0.78
Prismatic: plane_id=2	1.33	1.02
Prismatic: plane_id=3	1.82	1.02
Cartesian: $\Delta z = 1/4$	1.96	1.06

Appendix B: Processing Rates on Final Meshes used in Convergence Study

TABLE III Cell processing rates (seconds/iteration/cell) on final meshes, upwind scheme

Grid Type	RS6000 Model 590	CRAY C-90, eagle
Hexahedral	3.41e-4	2.51e-5
Tetrahedral	2.27e-4	1.50e-5
Prismatic: plane_id=1	2.88e-4	2.04e-5
Prismatic: plane_id=2	2.89e-4	2.02e-5
Prismatic: plane_id=3	2.79e-4	2.05e-5
Cartesian: $\Delta z = 1/4$	3.47e-4	2.20e-5

TABLE IV Cell processing rates (seconds/iteration/cell) on final meshes, central scheme

Grid Type	RS6000 Model 590	CRAY C-90, eagle
Hexahedral	1.84e-4	1.94e-5
Tetrahedral	1.26e-4	1.14e-5
Prismatic: plane_id=1	1.55e-4	1.57e-5
Prismatic: plane_id=2	1.53e-4	1.54e-5
Prismatic: plane_id=3	1.50e-4	1.55e-5
Cartesian: $\Delta z = 1/4$	1.82e-4	1.60e-5

References

[1] K. Morgan, J. Peraire, and J. Peiro. Unstructured Grid Methods for Compressible Flows. In *Computational Fluid Dynamics*. Von Kármán Institute for Fluid Dynamics, Lecture Series 1990-04, 1990.

[2] S. Pirzadeh. Unstructured Viscous Grid Generation by Advancing-Layers Method. AIAA-93-3453, 1993.

[3] T. Barth. On Unstructured Grids and Solvers. In *Computational Fluid Dynamics*. Von Kármán Institute for Fluid Dynamics, Lecture Series 1990-04, 1990.

[4] W. Coirier. *An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*. PhD thesis, The University of Michigan, Department Aerospace Engineering. Also published as NASA TM 106754, 1994.

[5] J. Melton, M. Berger, M. Aftosmis, and M. Wong. 3D Applications of a Cartesian Grid Euler Method. AIAA 33rd Aerospace Science Meeting, AIAA Paper 95-0853, 1995.

[6] S. Karman. SPLITFLOW: A 3D Unstructured Cartesian/Prismatic Grid CFD Code for Complex Geometries. AIAA-95-0343, 1995.

[7] J. Melton, S. Pandya, and J. Steger. 3D Euler Flow Solutions using Unstructured Cartesian and Prismatic Grids. AIAA paper AIAA-93-0331, 1993.

[8] S. Connell and M. Braaten. Semi-Structured Mesh Generation for 3D Navier-Stokes Calculations. GE Report 94CRD154, 1994.

[9] S. Connell, J. Sober, and S. Lamson. Grid Generation and Surface Modeling for CFD. To Appear in the 1995 NASA Surface Modeling and Grid Generation Conference Procedings, 1995.

[10] Y. Kallinderis, A. Khawaja, and H. McMorris. Hybrid Prismatic/Tetrahedral Grid Generation for Complex Geometries. AIAA paper AIAA-93-0669, 1993.

[11] V. Parthasarathy and Y. Kallinderis. Hybrid Adaptation Method and Directional Viscous Multigrid with Prismatic-Tetrahedral Meshes. AIAA paper AIAA-95-0670, 1995.

[12] D. Mavriplis and V. Venkatakrishnan. A Unified Multigrid Solver for the Navier-Stokes Equations on Mixed Element Meshes. AIAA paper AIAA-95-1666, 1995.

[13] P. Walatka, P. Buning, L. Pierce, and P. Elson. PLOT3D User's Manual, 1990.

[14] M. Braaten and S. Connell. A 3-D Unstructured Adaptive Multigrid Scheme for the Navier-Stokes Equations. GE Report 94CRD146, 1994.

[15] J. Peiro, J. Peraire, and K. Morgan. FELISA SYSTEM Reference Manuals I and II. User's Manuals, January, 1995.

[16] N. Frink. Recent Progress Toward a Three-Dimensional Unstructured Navier-Stokes Solver. AIAA paper AIAA-94-0061, 1994.

[17] D. Holmes and S. Connell. Solution of the 2D Navier-Stokes Equations on Unstructured Adaptive Grids. AIAA Paper 89-1932-CP, 1989.

[18] C. Mitchell. Improved Reconstruction Schemes for the Navier-Stokes Equations on Unstructured Meshes. AIAA paper AIAA-94-0642, 1994.

[19] P. L. Roe. Approximate Riemann Solvers, Parameter Vectors and Difference Schemes. *Journal of Computational Physics*, 43, 1981.

[20] B. van Leer. Flux-vector Splitting for the Euler Equations. *Lecture Notes in Physics*, 170, 1982.

[21] M.-S. Liou. A Continuing Search for a Near-Perfect Numerical Flux Scheme: Part I: AUSM+, 1994.

[22] E. Turkel and V. Vatsa. Effect of Artificial Viscosity on Three-Dimensional Flow Solutions. *AIAA Journal*, 32(1), 1994.

[23] C. Mitchell. Private Communication, 1994.

[24] D. Knight. A Fully Implicit Navier-Stokes Algorithm Using an Unstructured Grid and Flux Difference Splitting. AIAA Paper 93-0875, 1993.

[25] M. Aftosmis, D. Gaitonde, and T. S. Tavares. On the Accuracy, Stability and Monotonicity of Various Reconstruction Algorithms for Unstructured Meshes. AIAA Paper AIAA-94-0415, 1994.

[26] H. Luo, J. Baum, and R. Lohner. An Improved Finite Volume Scheme for Compressible Flows on Unstructured Grids. AIAA-95-0348, 1995.

[27] W. Coirier and K. Powell. An Accuracy Assessment of Cartesian-Mesh Approaches for the Euler Equations. *Journal of Computational Physics*, 117:121–131, 1995.

[28] E. Sparrow, C. Hixon, and G. Shavit. Experiments on Laminar Flow Development in Rectangular Ducts. *Journal of Basic Engineering, Transactions of the ASME*, pages 116–124, 1994.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE January 1996	3. REPORT TYPE AND DATES COVERED Technical Memorandum

4. TITLE AND SUBTITLE A Mixed Volume Grid Approach for the Euler and Navier-Stokes Equation	5. FUNDING NUMBERS WU-505-62-52
6. AUTHOR(S) William J. Coirier and Philip C.E. Jorgenson	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191	8. PERFORMING ORGANIZATION REPORT NUMBER E-10065
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, D.C. 20546-0001	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA TM-107135 AIAA-96-0762

11. SUPPLEMENTARY NOTES
Prepared for the 34th Aerospace Sciences Meeting and Exhibit sponsored by the American Institute of Aeronautics and Astronautics, Reno, Nevada, January 15-18, 1996. Responsible person, William J. Coirier, organization code 2610, (216) 433-5764.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category 02 This publication is available from the NASA Center for Aerospace Information, (301) 621-0390.	12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

An approach for solving the compressible Euler and Navier-Stokes equations upon meshes composed of nearly arbitrary polyhedra is described. Each polyhedron is constructed from an arbitrary number of triangular and quadrilateral face elements, allowing the unified treatment of tetrahedral, prismatic, pyramidal and hexahedral cells, as well the general cut cells produced by Cartesian mesh approaches. The basics behind the numerical approach and the resulting data structures are described. The accuracy of the mixed volume grid approach is assessed by performing a grid refinement study upon a series of hexahedral, tetrahedral, prismatic and Cartesian meshes for an analytic inviscid problem. A series of laminar validation cases are made, comparing the results upon differing grid topologies to each other, to theory and experimental data. A computation upon a prismatic/tetrahedral mesh is made simulating the laminar flow over a wall/cylinder combination.

14. SUBJECT TERMS Euler/Navier-Stokes; Prismatic grids; Polyhedral control volumes; Cartesian grids	15. NUMBER OF PAGES 17
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT

National Aeronautics and
Space Administration

Lewis Research Center
21000 Brookpark Rd.
Cleveland, OH 44135-3191

Official Business
Penalty for Private Use $300

POSTMASTER: If Undeliverable — Do Not Return