INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids **31**: 309–323 (1999)

A UNIFIED-GRID FINITE VOLUME FORMULATION FOR COMPUTATIONAL FLUID DYNAMICS

SUKUMAR CHAKRAVARTHY*

Metacomp Technologies, Inc., 650 Hampshire Road, Suite 200, Westlake Village, CA 91361-2510, USA

SUMMARY

A new extremely flexible finite volume framework is presented. It is based on updating cell average values of the dependent variables. It is embedded in a unified-grid framework that unifies the treatment of structured and unstructured grids, single and multi-block grids, patched-aligned, patched-non-aligned and overset grids, and various cell shapes, including hexahedra, triangular prisms and tetrahedra in three dimensions, quadrilaterals and triangles in two dimensions and the linear element in one dimension. A novel discretization approach has been developed to deal with unified-grid topologies. It includes a piecewise-linear multi-dimensional non-oscillatory reconstruction procedure that is based on synergistic utilization of polynomials at cell vertices. Least-squares reconstruction is used when necessary. The concept of generalized neighborhoods is introduced to account for cell neighborhoods that include cells that are not logically connected through common nodes, faces, etc. This helps in the automatic treatment of all types of multi-block meshes. To go with such a general discretization procedure, new implicit relaxation procedures are introduced that help achieve fast convergence to steady state solutions. The framework has been implemented in a new CFD code (CFD++) using a finite volume formulation. The paper presents the methodology with the help of several annotated examples taken from various compressible and incompressible flows. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS: unified-grid finite volume formulation; CFD++ framework; cell average values

1. THE CFD++ FRAMEWORK

The numerical framework of CFD++ is based on the following general elements:

1. Unsteady compressible and incompressible Navier–Stokes equations with turbulence modeling, including specialization to low-speed flows. Various forms of the Navier–Stokes equations and even other equation sets are also easily added. The examples presented in this paper are for compressible flow.
2. Unification of structured curvilinear, and unstructured grids, including hybrids. Multi-block structured grids can easily be imported into the flow solver. Similarly, existing unstructured grids can be used equally easily.
3. Unification of treatment of various cell shapes, including hexahedral, tetrahedral and triangular prism cells (three-dimensional), quadrilateral and triangular cells (two-dimensional) and linear elements (one-dimensional). Other special cells for self-similar flows and on-surface problems are also available.

* Correspondence to: Metacomp Technologies, Inc., 650 Hampshire Road, Suite 200, Westlake Village, CA 91361-2510, USA.

CCC 0271–2091/99/170309–15$17.50

4. Treatment of multi-block patched-aligned (nodally connected), patched-non-aligned and overset grids. Interblock connectivity and blanking is automatically determined. No separate pre-processing module is necessary.
5. Total variation diminishing (TVD) discretization based on a new multi-dimensional interpolation framework. This results in an extremely versatile discretization framework that can handle the above mentioned cell and grid topologies seamlessly.
6. Riemann solvers to provide proper signal propagation physics. A new modification of Roe's Riemann solver permits great savings in memory. The total memory required for CFD++ is similar to that for a structured grid flow solver, even though the bookkeeping in CFD++ is very general. Various Riemann solvers and their replacement modules can be added and used.
7. Consistent and accurate discretization of viscous terms use the same multi-dimensional polynomial framework and memory saving implementation. Non-TVD derivatives computed for inviscid terms are reused to compute viscous terms and turbulence model discretization. This provides for additional computational efficiency.
8. Various pointwise turbulence models that do not require knowledge of distance to walls (wall-distance-free (WDF)). These are the ideal models for multi-block (patched and overset), unstructured grid and massively parallel approaches. They show very little sensitivity to grid skewness. These models are also eminently suitable for separated flows and free shear layers.
9. Versatile boundary condition implementation includes a rich variety of integrated boundary condition types.
10. All Mach numbers can be simulated. Improved convergence to steady state solutions for low-speed and incompressible flow cases is achieved by using various pre-conditioners.
11. Various explicit and implicit time stepping schemes are provided. The implicit schemes for arbitrary mesh topologies are based on relaxation methods for unfactored upwind schemes.

2. SYNOPSIS OF THE GOVERNING EQUATIONS AND THE NUMERICAL METHOD

The conservation form of the governing differential equations currently supported by CFD++ can be written as

$$\frac{\partial q}{\partial t}+\frac{\partial (f_{\mathrm{i}}-f_{\mathrm{v}})}{\partial x}+\frac{\partial (g_{\mathrm{i}}-g_{\mathrm{v}})}{\partial y}+\frac{\partial (h_{\mathrm{i}}-h_{\mathrm{v}})}{\partial z}=S,$$

where the vector q represents the dependent conservation variables; f, g and h represent the fluxes in the three spatial directions; and S represents the source terms. The subscripts i and v are for the inviscid and viscous terms respectively.

2.1. Finite volume framework

The equations are discretized using a finite volume implementation. Assume that a computational cell with a volume, V, is known. The differential form of the equations are integrated over this volume in the following way:

$$\iiint_V \left(\frac{\partial q}{\partial t}+\frac{\partial f}{\partial x}+\frac{\partial g}{\partial y}+\frac{\partial h}{\partial z}-S\right) \mathrm{d}x\ \mathrm{d}y\ \mathrm{d}z=0,$$

where f, g and h are now the full fluxes (viscous and inviscid taken together). The spatial terms can be rewritten in the following way:

$$\frac{\partial f}{\partial x}+\frac{\partial g}{\partial y}+\frac{\partial h}{\partial z}=\nabla\cdot\vec{F}, \quad \text{where } F=f\hat{j}+g\hat{k}+h\hat{l}.$$

Next Gauss' and Leibnitz's rules are applied. Gauss's rule allows the volume integral to be written as a surface integral and Leibnitz's rule allows the time derivative to be taken outside the integral with the addition of the proper term within the integral. Therefore, the equation now can be written as

$$\frac{\partial}{\partial t}(\bar{q}V)+\oiint_A(\vec{F}\cdot\hat{n}+\hat{n}_t q)\,\mathrm{d}A=\iiint_V S\,\mathrm{d}V,$$

where $\bar{q}$ is the cell average of the dependent variables, $\hat{n}$ is the outward pointing unit normal and $\hat{n}_t$ is normal movement of the control volume. These quantities are defined as:

$$\bar{q}=\frac{1}{V}\iiint_V q\,\mathrm{d}V,$$

$$\hat{n}=\hat{n}_x\hat{j}+\hat{n}_y\hat{k}+\hat{n}_z\hat{l},$$

$$\hat{n}_t=-\dot{x}\hat{n}_x-\dot{y}\hat{n}_y-\dot{z}\hat{n}_z.$$

2.2. Spatial discretization using vertex-oriented multi-dimensional TVD polynomials

The integral term, $\oiint_A(\vec{F}\cdot\hat{n}+\hat{n}_t q)\,\mathrm{d}A$, in the equations is evaluated by a mid-point rule formula:

$$\oiint_A(\vec{F}\cdot\hat{n}+\hat{n}_t q)\,\mathrm{d}A=\sum_{\text{Faces}}\iint_A(\vec{F}\cdot\hat{n}+\hat{n}_t q)\,\mathrm{d}A\approx\sum_{\text{Faces}}(\vec{F}_i\cdot\hat{n}_i+\hat{n}_{t_i}q_i)A_i,$$

where i denotes the centroidal location of each face. This yields a second-order approximation to the integral of the fluxes over the cell faces. In the formulation, the cell averages of the dependent variables (conservation variables) are updated; therefore, in order to find pointwise values at the cell face centroids, polynomials are constructed from the cell average values. If the cell average values are used as approximations for the cell face centroidal values, then the scheme becomes a first-order scheme. For a second-order approximation, a linear multi-dimensional polynomial can be constructed in the following way:

$$B(x,y,z)=b_0+b_1(x-x_c)+b_2(y-y_c)+b_3(z-z_c),$$

where (x_c, y_c, z_c) is the centroid of the cell for which the polynomial is being constructed. One approach taken in CFD++ is to evaluate a polynomial for each vertex of a given cell, i.e. a vertex-oriented polynomial. For example, a tetrahedral element has four vertices and thus four polynomial constructions. For the elements considered in CFD++, each vertex is part of three faces in the three-dimensional case. The above polynomial has four coefficients that need to be solved for; therefore, for each vertex, the three neighboring cells (of the three faces) are used along with the cell itself to come up with the polynomial coefficients. Thus, a linear fit of the four data points is achieved. For vertices where any of the faces are boundaries, a polynomial is not initially constructed. At boundaries, after an extrapolation is done and boundary conditions are applied, the boundary values can then be included in polynomial evaluations. Thus, at the boundaries, two modes are used, i.e. excluding or including the boundary points in the evaluation of the polynomials. CFD++ has the ability to construct

least-squares polynomial for each vertex. This polynomial will fit cell average values of the dependent variables for all cells that share the vertex node. Least-squares polynomials can also be used to fit cell average values of the dependent variables for cells from various blocks that contain the physical co-ordinates of a given vertex. This approach greatly facilitates the treatment of various types of multi-block grids, including patched-non-aligned and overset meshes.

It is well known that polynomial evaluations can introduce new minima and maxima in a given data field. In order to avoid this phenomenon, which can cause oscillations near discontinuities, limiters are introduced that limit the slopes by comparing them with their counterparts from other vertex polynomials within that cell. More details of such slope-limiting approaches can be found in [1].

The steps of the overall spatial discretization scheme can be summarized as follows:

1. Compute all vertex polynomials for a given cell.
2. Compute in-face and out-of-face Dq's for each face.
3. For each face compare the in-face and out-of-face quantities slope limiters.
4. Once the TVD Dq's are calculated for each cell face, a weighted average of these quantities is as follows:

$$\Delta q_{\mathrm{TVD}} = \frac{1+\phi}{2}(\text{in-face})_{\mathrm{TVD}} + \frac{1-\phi}{2}(\text{out-of-face})_{\mathrm{TVD}}, \quad -1 \le \phi \le 1.$$

For evaluation of the viscous fluxes a simple average of all the vertex polynomials (non-TVD) of a face are used. The derivatives of the temperature and three velocity components in the viscous fluxes are constructed from polynomial representations of the conservation variables. In the above discussion, a full three-dimensional case was explained. Obvious simplifications exist for the two- and one-dimensional cases and are coded in CFD++ to achieve computational efficiency for lower dimensional problems.

Once the unlimited and limited polynomials are computed, the goal is to find the values of the fluxes at each integration point. For each integration point, there are two possible values from the polynomial evaluations: a value from the left (from that cell's polynomial) and a value from the right (from a neighboring cell). For the viscous fluxes, a simple average of the two values of dependent variables and their derivatives is taken; however, for the inviscid fluxes, a Riemann solver is invoked in order to insure a correct signal propagation of the hyperbolic (inviscid) terms. More details about the traditional and modified Riemann solvers in CFD++ are given in [1]. The above vertex-oriented approach results in a compact discretization of derivatives at cell faces and avoids, by construction, decoupling effects that arise in other approaches based on cell-centered multi-dimensional polynomial reconstruction.

2.3. Proximity-based neighborhoods

In evolving the CFD++ framework to meet the needs of computing with multi-block meshes, a new and very versatile interconnection capability and the accompanying discretization have been identified. The CFD++ framework originally included the first implementation of a very general multi-block capability that could handle nodally aligned (patched-aligned), nodally non-aligned (patched-non-aligned) and overset grid capability. However, the interconnection mechanism for this implementation was based on finding connecting cells for cell face centroids. An improved approach based on connecting nodes with nearby or overlying cells has been identified, implemented and tested. Since grid nodes are the fundamental means of defining the grid, the boundaries, etc., this seemed more natural. This

new approach also permits a very natural and versatile treatment of grid subdivisions. It also is a perfect fit for the cell vertex-based multi-dimensional interpolation procedure that is used in CFD++.

2.4. Vertex-oriented polynomials and hierarchical meshes

Vertex-oriented polynomials are easily adaptable to deal with hierarchically subdivided meshes. In such meshes, adjacent mesh cells can be of different sizes. For example, in the two-dimensional case, one mesh element face of a bigger cell can lie adjacent to two mesh faces. In the three-dimensional case, one face may neighbor four faces. The vertex-oriented polynomials can be used in a least-squares mode as described above. At each node, a collection of neighboring cells (elements) is identified. These can include all cells that logically (in the bookkeeping framework) contain that node. It can also include cells that are 'proximal' (near) that node. Once proximity-based neighborhoods are included, it is very easy to deal with hierarchy-induced cell neighborhoods. The question of formal conservation across such intercell boundaries (faces of different sizes next to each other) is important, but the numerical examples demonstrate the correctness of this approach. Formal consistency can be achieved by construction, but details will be presented elsewhere because this aspect is not a primary focus of this paper.

2.5. Vertex-oriented polynomials and intersecting meshes

The proximity-based neighborhood alluded to in the previous subsection helps deal with intersecting (overlapped) meshes as well as patched-aligned and non-aligned meshes. A given node is part of at least one cell in the bookkeeping data structure. For a given node, the shortest link to any other node of cell(s) that include this node as a vertex can be found. When proximity-based search finds a connection to another cell or cells, it can be because (a) the other cell(s) contain the physical co-ordinates of the given node; (b) the other cells do not contain the physical co-ordinates but a projection to the nearest cell face is found. In the latter situation, the connection is currently deemed acceptable if this projected distance is less than the shortest link distance defined earlier. This approach helps to find the connections across gaps.

2.6. Implicit relaxation with pre-conditioning

An efficient scheme for solving the discretized system of the equation has been developed. It combines three basic ideas: implicit local time stepping, relaxation and pre-conditioning. Pre-conditioning the equations ideally equalizes the eigenvalues of the inviscid flux Jacobians and removes the stiffness arising from large discrepancies between the flow and sound velocities at low speeds. Use of an implicit scheme circumvents the stringent stability limits suffered by their explicit counterparts, and successive relaxation allows update of cells as information becomes available and thus aids convergence.

2.7. Relaxation sweeps using cell ordering

The order in which the cells are 'swept' in the relaxation scheme can also play a significant role in its convergence properties. Several options have been considered:

(a) Sweep order according to cell numbers.
(b) 'Boundaries-on-in' ordering of cells.

(c) Ordering starting from a specific boundary.
(d) Checkerboard (or colored) scheme numbering.
(e) Cell ordering based on eigenvalue.

3. ILLUSTRATIVE EXAMPLES

3.1. Overset multi-block mesh

In Figure 1, a four-block mesh is presented. The four blocks are roughly associated with the four quadrants of a rectangular region. All the blocks overlap each other. In particular, in the neighborhood of the center of the outer rectangle, all blocks overlap each other. It is an interesting challenge to devise a method that can transparently compute through such a mesh. The CFD++ capability is able to solve the conservation laws (inviscid flow in this example) through such meshes. The interblock mesh boundaries are identified as slightly darker lines. The flow field solved corresponds to an oblique shock generated at the lower left corner of the mesh for a Mach 2 free stream and a 10° turn. This shock wave is reflected by the upper boundary (the particular choice of the boundary condition on the upper boundary is not significant for present purposes). Figure 1 also shows the pressure contour lines. The results confirm that proper shock-capturing is achieved even through such a multiply-overlapped mesh.

3.2. Patched and overset multi-block mesh

The same problem is now presented using a slightly different mesh topology. The four blocks are still to be found but not all the blocks overlay each other (Figure 2). For example, the lower left block and upper left block are patched and not overset with respect to each other. One can also notice that these two blocks are not nodally aligned. There is nodal alignment between the lower left block and the lower right block. The two blocks on the right overlap each other. The upper left block is also found to overlay the upper right block. Pressure contour lines are also shown in Figure 2. The CFD++ implementation computes seamlessly through such meshes. This example serves as a prototype for testing the code's capability to handle all types of interblock connectivities.

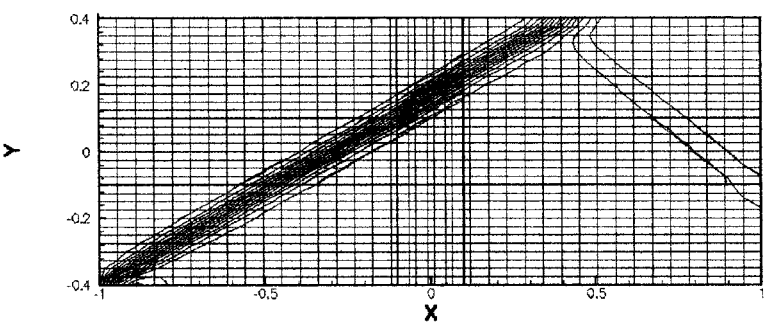

Figure 1. Overset mesh and pressure contours for oblique shock problem.

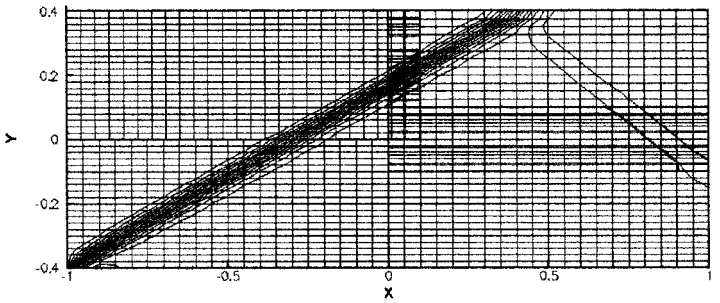

Figure 2. Patched and overset mesh and pressure contours for oblique shock problem.

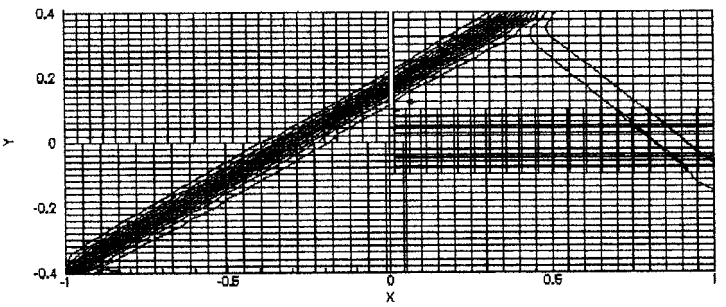

Figure 3. Multi-block mesh with interblock gap and pressure contours for oblique shock problem.

3.3. Patched, overset multi-block mesh with interblock gap

The next example is included partly for amusement and partly to illustrate a very important aspect of the CFD++ capability. Four blocks are still used, but now we introduce a gap between the upper left and upper right blocks. Small gaps could arise in putting grids together with different resolution levels or even different offsets from a reference location. For example, if one considers two different meshes on a circular cylinder (or any curved surface) that are constructed so that all mesh points on the lower grid boundary are directly located on the circular cylinder, one will find that sometimes one of the meshes will have lines outside the computational domain as defined by the other mesh. While special issues arise in such situations when one must interpolate information between such meshes, in principle, CFD++ can avoid such issues by being able to make use of appropriate information in both meshes simultaneously. In any case, back to the situation at hand, the zonal interconnection procedure can properly connect points on the right boundary of the upper left block with cells on the left boundary of the upper right block, etc., based on proximity considerations. Figure 3 displays the mesh and pressure contour lines for this case.

3.4. Laminar flow over flat plate

Now results are included showing the effect of such overlapped mesh treatment through boundary layer regions of an example problem that involves laminar supersonic flow ($M = 2$) over a flat plate. A close-up of the overlapping mesh in the vicinity of the boundary layer is shown in Figure 4. Axial velocity contours are displayed in Figure 5. Complete continuity in the contour lines across zonal boundaries is seen. The velocity profile at $x = 0.1$, obtained

from the numerical solution, is compared with the exact solution in Figure 6. The quality of the agreement is evidence of the quality of the numerical procedure. It is observed here that conventional overset mesh approaches deal with the situation as an information transfer problem from one mesh to another. This simulation deals with overset meshes at the discretization level directly.

Figure 7 shows the convergence history using a point-implicit (CFL 50), the relaxation scheme using CFL 50 and two forward LHS sweeps in the streamwise direction, the other relaxation scheme using eigenvalue-based cell ordering and two forward–backward LHS sweeps. Roughly six orders of magnitude drop in the residual was achieved in 400 steps by the eigenvalue-based cell ordering, even in this highly stretched grid.

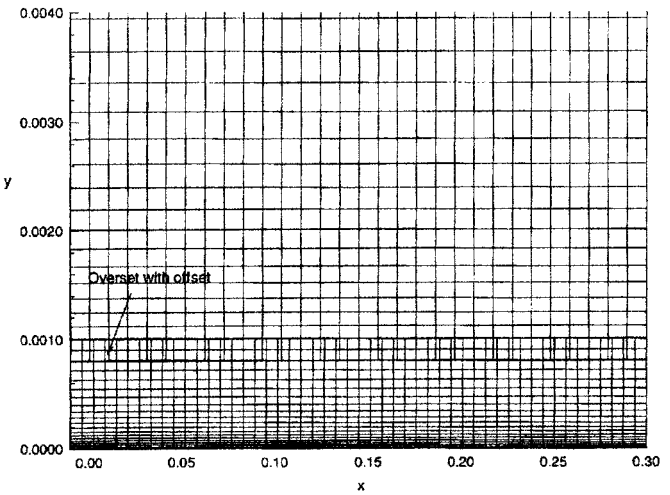

Figure 4. Overset grid with overlap used for laminar flat plate calculations ($M = 2.0$).

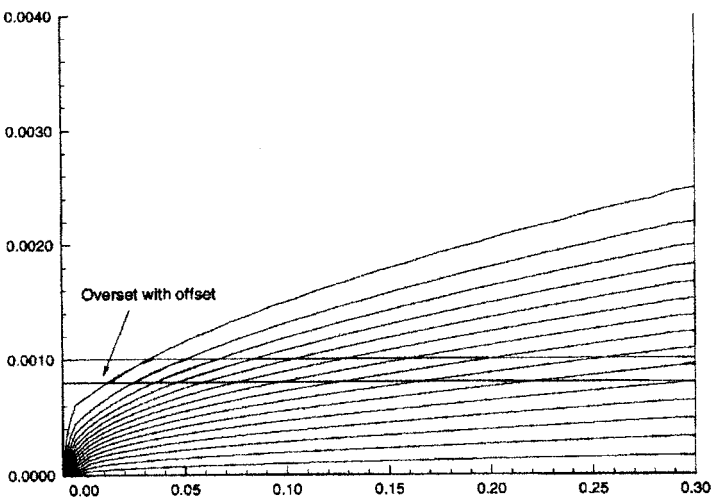

Figure 5. Contours of streamwise velocity for $M = 2.0$ laminar flat plate.

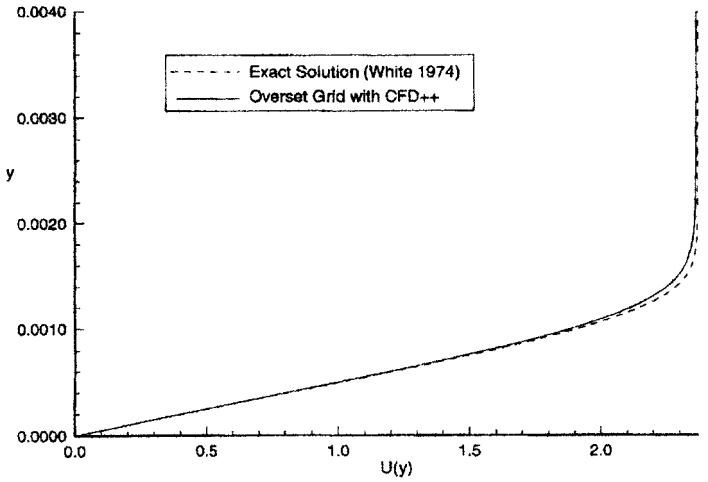

Figure 6. Comparison between velocity profiles at $x = 0.1$ (CFD++ and exact).

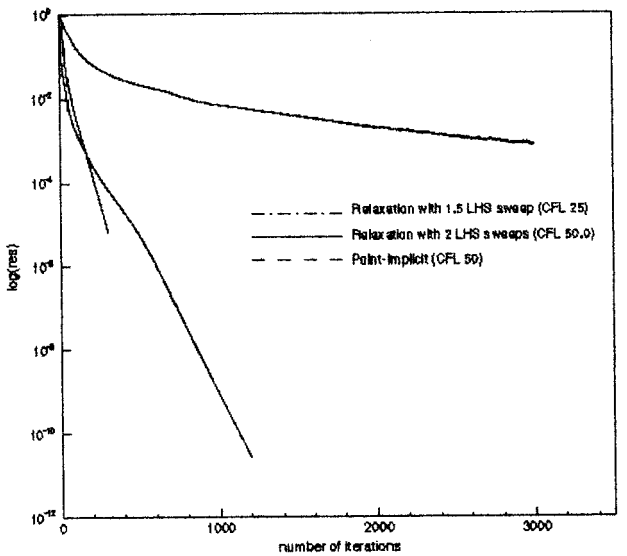

Figure 7. Convergence histories for Mach 2.0 laminar flow over flat plate.

3.5. Inviscid flow over cylinder

A Mach 0.4 inviscid flow over a cylinder is calculated using a variety of mesh topologies. The goal here is to demonstrate the ease with which the relaxation scheme operates within the unified-grid framework. Four grids were chosen for this analysis: the first was a 26×26 quadrilateral grid clustered near the surface of the cylinder, the second was a finer version of the same grid (51×51), the third a triangular mesh grid and the fourth a hybrid grid that has a hierarchically subdivided section near the top of the cylinder. Figures 8(a)–(c) show these mesh topologies. In all cases, the relaxation scheme was set up so that it operated at a local CFL of 50 with two forward–backward boundaries-on-in LHS sweeps. Figure 9 shows the

convergence characteristics on the four mesh topologies compared with a second-order Runge–Kutta and first-order point-implicit schemes. Nearly four orders of magnitude drop in the residual is realized within 150 steps. One should note that the leveling off seen in the residual of the hierarchical mesh is due to the truncation error mismatch on the boundaries of the two grids.

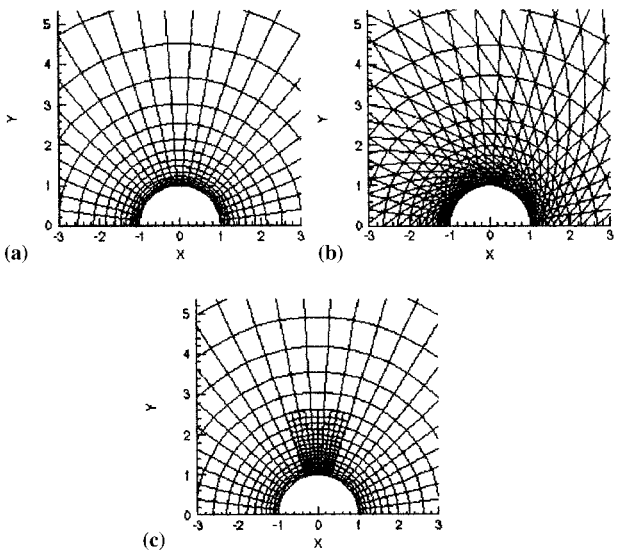

Figure 8. (a) Quadrilateral mesh around circular cylinder, (b) triangular mesh, (c) hierarchical mesh near cylinder.

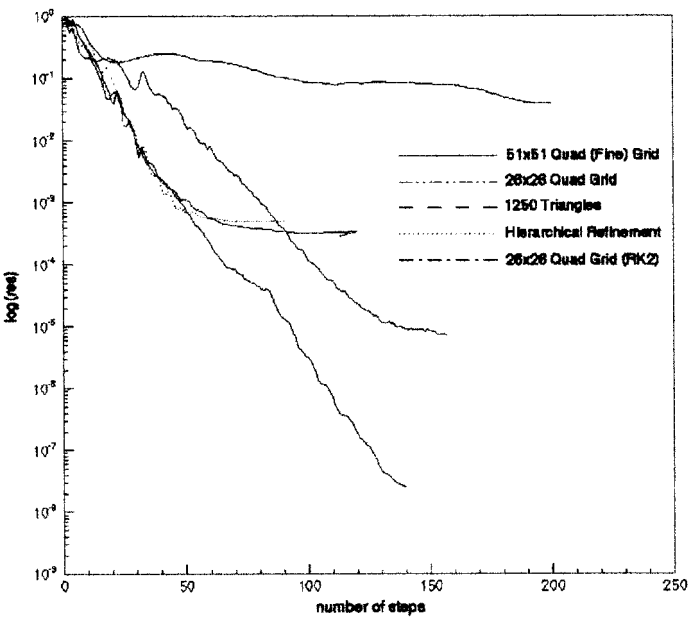

Figure 9. Convergence histories for Mach 0.4 inviscid flow over circular cylinder.

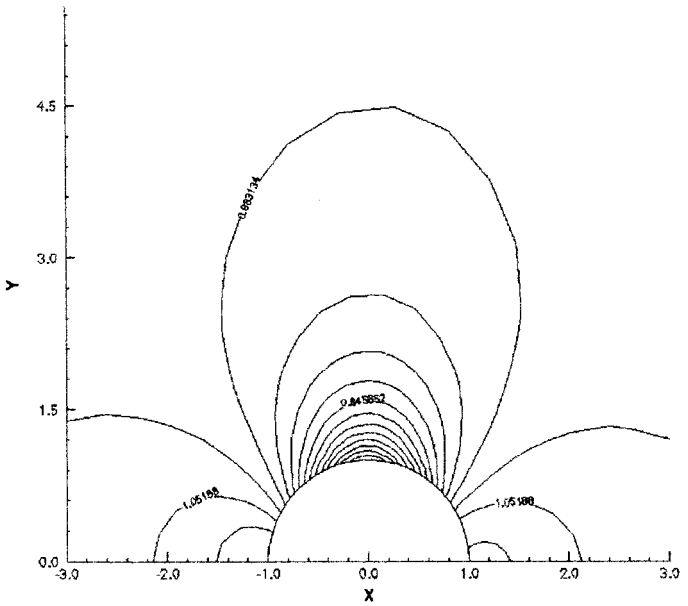

Figure 10. Pressure contour lines on circular cylinder mesh with local refinement.

In the case of the hierarchically refined mesh, at the junction of the fine-grid and coarse-grid regions, a mesh cell face from the coarse part of the grid will be juxtaposed with two mesh faces from the finer part. The usual mechanisms that automatically connect mesh faces are geared towards finding exact matches between cells that share identically defined cell faces. All cell faces for which a matching cell is not found (sharing the same nodes of the face) are markable as boundary faces. A feature that automatically marks such boundary faces as interzonal boundary faces when the user does not supply physical boundary conditions for them has been included in CFD++. At node points that define such faces, proximity-based connectivities are automatically performed in CFD++ using a node-oriented connection scheme. Once again, a seamless discretization procedure results. Figure 10 displays pressure contour lines. The result corresponds to inviscid flow with free stream Mach number of 0.4. The fore–aft symmetry of the results and the continuous transfer of information across coarse/fine boundaries are evident.

3.6. Circular cylinder with near-surface and Cartesian background meshes

Next illustrated is the ability to deal with more complex overset grid mesh topologies that require the identification of nodes outside the computational domain. Figure 11 displays a composite mesh made up of a near surface mesh around a circular cylinder and a Cartesian background mesh. The physical domain is assumed to be outside the circular cylinder and inside the outer boundaries of the Cartesian mesh. Many nodes of the background mesh are inside the circular cylinder and therefore must be marked for exclusion (or blanking) along with the cells that contain such nodes. This is achieved by a unique distance function based approach.

Figure 11(b) shows the pressure contours obtained on this composite mesh. Note that in this simple example, the background mesh has not been locally adapted in the vicinity of the edge of the near-surface mesh. Additionally, the results only represent an inviscid calculation at an

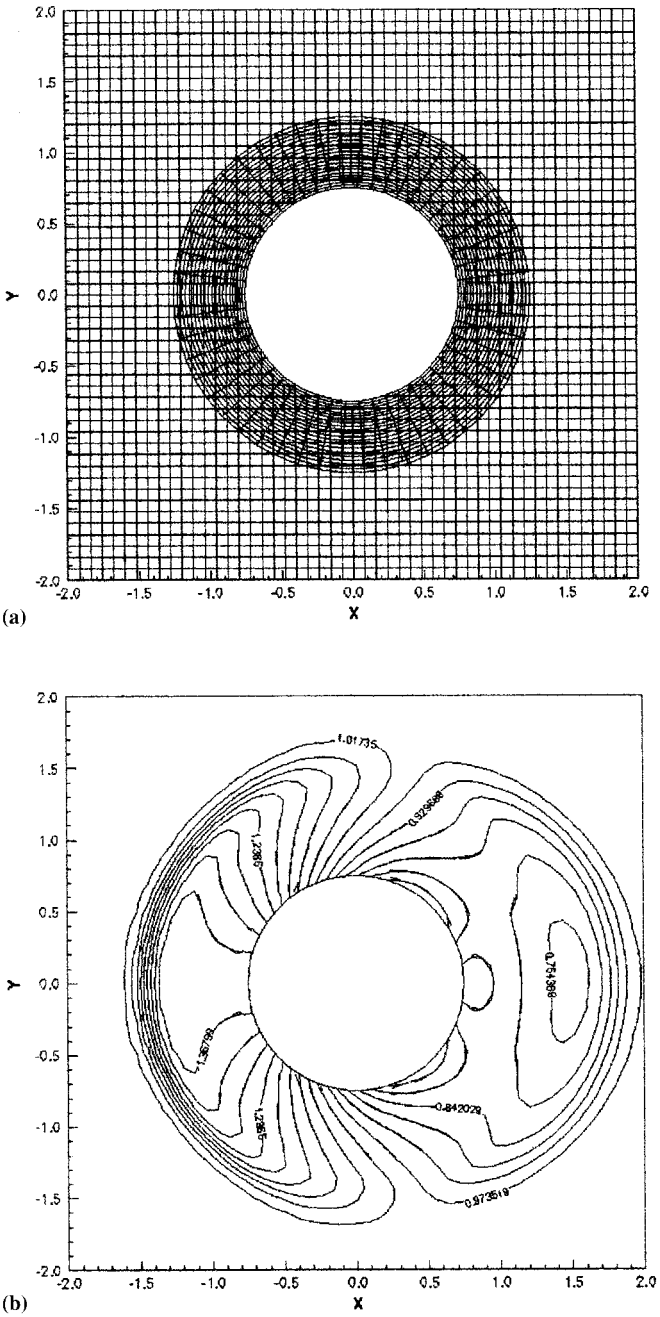

Figure 11. (a) Near-surface and Cartesian background meshes, (b) pressure contour lines for complex overset mesh around cylinder.

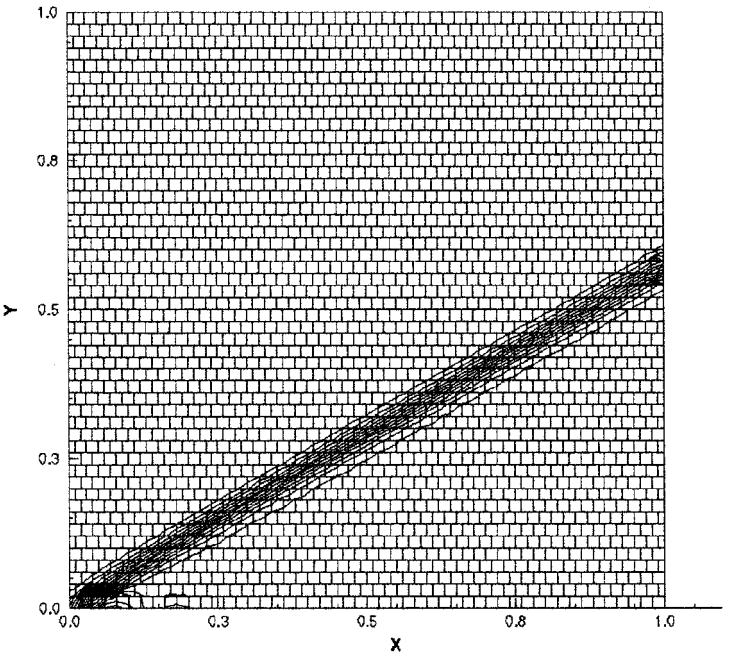

Figure 12. Brick wall grid and pressure contours for the oblique shock problem.

early stage of a transient that occurs when a free stream subsonic flow encounters the presence of the circular cylinder through the imposition of boundary conditions.

3.7. 'Brick-wall' mesh

Several examples have already been presented that show how effectively CFD++ can handle various types of interblock and intercell connectivities. An extreme example is provided here in which there exists no nodal connectivity between any cell in the vertical direction (except at boundaries). Such a grid is shown in Figure 12. The flow field corresponds to an oblique shock emanating from the lower left corner of the mesh for a free stream Mach number of 2 and a 10° turn. The pressure contours lines obtained are also shown in Figure 12. The shock angle and pressure rise are correctly predicted. CFD++ automatically detects the proper neighborhood of cells at each cell vertex and appropriately uses them for evaluating the reconstruction polynomials. This example shows that CFD++ can determine connectivity by proximity fully automatically.

3.8. Double wedge airfoil

The next example has been chosen to illustrate how background meshes can be used as 'glue' to bridge gaps between other near-surface meshes. Figure 13 shows the surface geometry as well as a single layer of near-surface cells next to it for a double wedge airfoil configuration. The forward and aft sections of the double wedge were specified as two different sections with no logical nodal connectivity between them (at the maximum thickness point). A close-up of this region is shown in Figure 14. The single layer of near-surface cells is used merely to have uncluttered plots in Figures 13 and 14. The actual near-surface mesh comprised a 10-cell width layer.

Figure 15 displays a close up view of a composite of both the near-surface meshes and a background hierarchical mesh that was geometrically adapted to every boundary node of the near surface meshes.

Figure 16 shows the composite mesh (full view) after the background mesh has been further subdivided to produce a more refined mesh away from the double wedge surfaces. Figure 17 shows the pressure contour lines obtained on such a mesh demonstrating the 'gluing' ability of background meshes of the type used here.

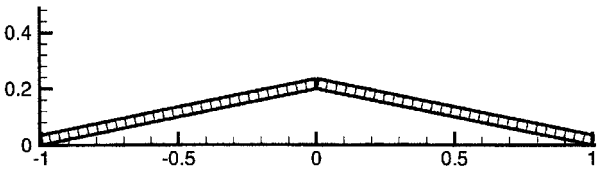

Figure 13. Single layer near-surface mesh on double wedge.

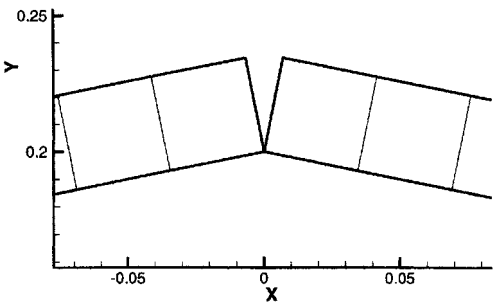

Figure 14. Close-up of double wedge near-surface mesh showing gap.

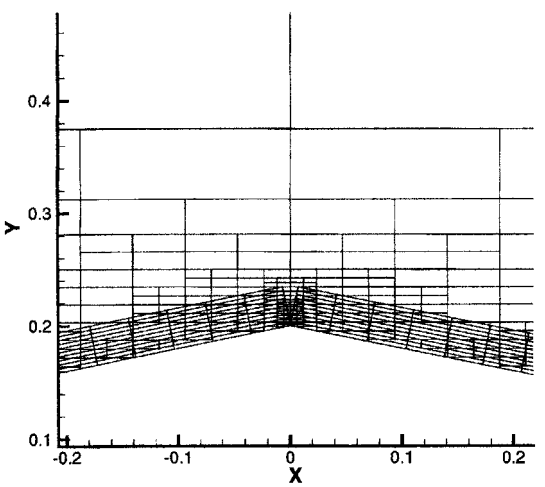

Figure 15. Close-up of composite mesh.

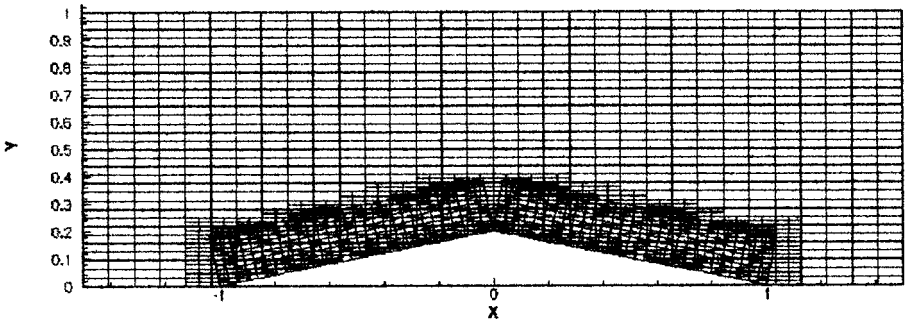

Figure 16. Composite mesh with more refined far-field for diamond wedge airfoil.

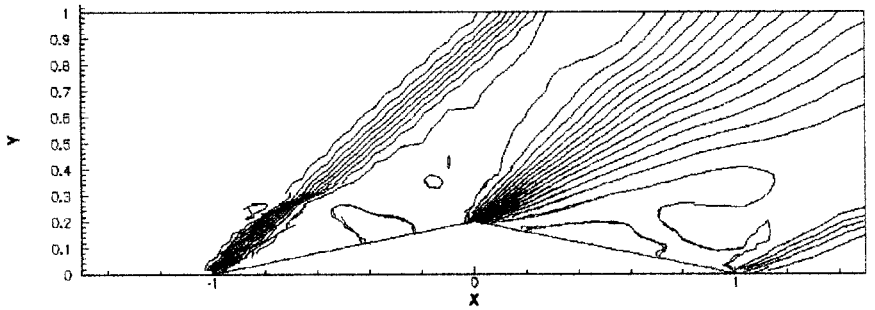

Figure 17. Pressure contour lines on composite mesh for diamond wedge airfoil.

4. CONCLUDING REMARKS

The unified-grid methodology has been introduced along with several illustrative examples. The main enabling feature is a multi-dimensional interpolation framework that helps deal with all the different multi-block connectivities. More details about CFD++ can be found in [1–3].

REFERENCES

1. O. Peroomian and S. Chakravarthy, 'A 'grid-transparent' methodology for CFD', *AIAA Paper No. 97-0724*, January 1997.
2. S.R. Chakravarthy, U.C. Goldberg, O. Peroomian and B. Sekar, 'Some algorithmic issues in viscous flows explored using a unified-grid CFD methodology', *AIAA Paper No. 97-1944*, 1997.
3. O. Peroomian, S. Chakravarthy and U. Goldberg, 'Convergence acceleration for unified-grid formulation using preconditioned implicit relaxation', *AIAA Paper No. 98-0116*, January 1998.

